Transportation Research Part C 140 (2022) 103711

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Synchromodal transport planning with flexible services: Mathematical model and heuristic algorithm

Yimeng Zhang [a], Wenjing Guo [b,*], Rudy R. Negenborn [a], Bilge Atasoy [a]

[a] Department of Maritime and Transport Technology, Delft University of Technology, 2628 CD Delft, The Netherlands
[b] School of Transportation and Logistics Engineering, Wuhan University of Technology, 430063, Wuhan, China

ARTICLE INFO

Keywords:
Synchromodal transport planning
Flexibility
Mixed integer linear programming
Adaptive large neighborhood search

ABSTRACT

As a critical feature of synchromodal transport (ST), service flexibility plays an important role in improving the utilization of resources to reduce costs, emissions, congestions, and delays. However, none of the existing studies considered flexible services under the framework of synchromodality. This paper develops a Mixed Integer Linear Programming (MILP) model to formulate service flexibility in ST planning. In the MILP model, vehicles with flexible services as well as fixed services are both considered, and vehicle routes and request routes are planned simultaneously. Due to the computational complexity, an Adaptive Large Neighborhood Search heuristic is designed to solve the problem. Several customized operators are designed based on the characteristics of the studied problem. The proposed model is compared with the models developed in a highly-cited paper and a newly published paper that do not consider service flexibility. Case studies on small instances verified that the proposed model with flexibility performs better on all scenarios, including scenarios with different weights for the individual objectives, scenarios under congestion,and dynamic optimization scenarios. On large instances (up to 1600 shipment requests), the proposed model with flexibility reduces the cost by 14% on average compared with the existing models in the literature.

1. Introduction

International container transportation is facing the challenges of improving efficiency and reliability and reducing cost, delay, congestion, and emissions due to the pressures from both huge volume of global trade and higher requirements of stakeholders (Ambra et al., 2019). Global containerized trade reached 143 million twenty-foot equivalent units (TEUs) in 2019, while the expectations on transportation are increasing (UNCTAD, 2020). At the same time, greenhouse emissions from transportation reached 1103 million tons in Europe in 2019 (EEA, 2020). To achieve sustainable freight transportation, European Commission proposes to cut carbon emissions in transport by 60% by 2050 and shift 50% freight transport from road to rail and waterborne transport (Kallas, 2011). China also has the "Carbon Peak and Carbon Neutrality" Policy, which measures to achieve a peak in carbon emissions by 2030 and carbon neutrality by 2060 (Dong et al., 2021). Intermodal transport is a promising solution on the above issues and the modal shift towards sustainable modes. As an advanced version of intermodal transport, Synchromodal Transport (ST) is proposed to address these issues further (Zhang and Pel, 2016; Behdani et al., 2014). Synchromodality has been recognized as an innovative solution for achieving socio-economic and environmental sustainability by utilization of existing capacities and assets (Giusti et al., 2019; Ambra et al., 2019). As stated by Mes and Iacob (2016), on the corridor from the Netherlands to Italy, synchromodal planning could

* Corresponding author.
E-mail addresses: Yimeng.Zhang@tudelft.nl (Y. Zhang), wenjingguo@whut.edu.cn (W. Guo), R.R.Negenborn@tudelft.nl (R.R. Negenborn), B.Atasoy@tudelft.nl (B. Atasoy).

https://doi.org/10.1016/j.trc.2022.103711
Received 20 June 2021; Received in revised form 20 April 2022; Accepted 3 May 2022
Available online 24 May 2022

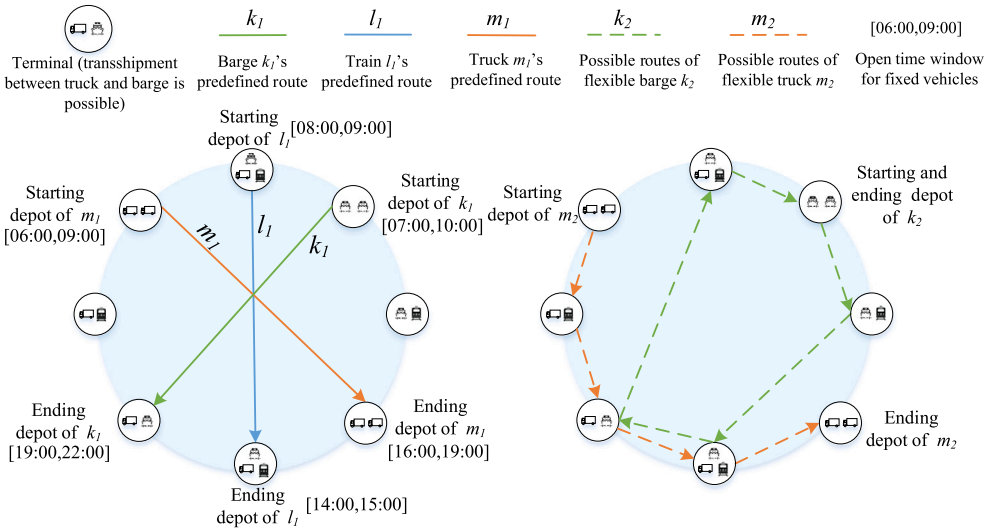

Fig. 1. Routing of vehicles with fixed and flexible services in synchromodal transport.

achieve an average cost reduction of 10.1% and a CO_2 reduction of 14.2%. A Rotterdam hinterland container transport case study shows that ST could increase service line occupancies (by 10%), reduce delivery times (by 8%), facilitate a modal shift from road transport towards rail transport (by 16%), and reduce CO_2 emissions (by 28%) compared with intermodal transport (Zhang and Pel, 2016).

As a distinct feature of synchromodality, service flexibility plays a key role in improving the utilization of resources (Behdani et al., 2014; Van Riessen et al., 2015; Zhang and Pel, 2016; Giusti et al., 2019; Delbart et al., 2021). The service flexibility means that the decision-maker can change vehicles' routes and schedules based on demand and available resources. In other words, flexible services enable the decision-maker to achieve the objective better by exploiting the benefits of each modal choice in ST. Flexibility has different functions under different scenarios. Specifically, when the objective is to minimize cost, flexible services can reduce more costs than predefined services by avoiding empty miles, improving loading factors of low-cost vehicles, minimizing storage time, etc. Besides, flexible services can provide more alternatives to alleviate impacts of congestions or other unexpected events than fixed services. Flexibility is also vital to handle transport demand in the most efficient and sustainable way by using different modes and routes in an integrated network (Giusti et al., 2019; Delbart et al., 2021).

However, the majority of the existing models in ST, e.g., Demir et al. (2016) and Guo et al. (2020), only consider services with fixed routes and schedules. The flexible services are not considered mainly due to the following reasons: (a) providing flexible services needs the development of various technologies, such as digital platform, information and communication technologies, and physical internet (Ambra et al., 2019; Giusti et al., 2019); (b) achieving flexible services needs to consider transshipment and synchronization of operations (Giusti et al., 2019); (c) tackling the optimization problem with flexible services needs customized, sophisticated and efficient algorithms due to computational complexity (Wolfinger, 2021). Therefore, existing studies usually assume that the routes and schedules of services are predefined, which loses the flexibility of trucks and ships. In reality, routes of trucks and ships can be changed according to demands and weather, and transport operators and shippers are flexible in their negotiations depending on the circumstances, such as transportation volume and disturbances (Van Riessen et al., 2013). The requirement of flexibility and the development of modern technologies are driving the transformation from fixed to flexible services in ST.

Fig. 1 illustrates the routing of vehicles with fixed versus flexible services. In the following, vehicles with fixed services and vehicles with flexible services are abbreviated as fixed vehicles and flexible vehicles, respectively. In Fig. 1, the nodes are ST terminals, which could be ports or truck/train stations, and these nodes are connected by roads, railways, or inland waterways. When the vehicle is fixed, it can only run between predefined terminals. In contrast, when it is a flexible vehicle, its transport network is expanded, and it can go to any terminal if there are suitable routes for it. Assuming the routes are fixed may cause empty miles and low load factors, which increases the transport cost. Fixed vehicles' departure and arrival times need to fit in the predefined open time windows at terminals. Besides, the schedules of different requests are the same when using the same fixed vehicles. In contrast, the schedules are adjustable when using flexible vehicles because the transport operator can customize schedules for different requests to avoid unnecessary storage and delay. Although flexible vehicles could bring benefits for ST, fixed vehicles are still necessary in the current ST. For instance, the schedules of freight trains are usually predefined due to the higher priority accorded to passenger trains (Wolfinger et al., 2019). Therefore, the mix of fixed and flexible vehicles needs to be considered in synchromodal transport panning.

Flexible routing and scheduling need a vehicle routing component, which is often addressed by coarse approximations in the existing models that cannot be applied to ST with flexible services (Drexl, 2012). For example, the links or paths are used

to "transport" containers in the literature (Van Riessen et al., 2013; Demir et al., 2016; Guo et al., 2020). When considering flexible routing and scheduling at the operational level, the transport operator needs to take the capacity and speed of each vehicle into account and decide which vehicle will be used to serve requests. Moreover, the schedules of flexible vehicles, such as arrival/departure time and waiting time, need to be decided by the model. It is more convenient to calculate these times by adding a vehicle routing component as in the case of Vehicle Routing Problems (VRPs). The request routing is also required due to the possible transshipment between vehicles. Therefore, request routing and vehicle routing need to be modeled simultaneously. Furthermore, the transshipments of requests and interdependency between vehicles complicate both routing and scheduling in synchromodal transport planning (Drexl, 2014; Rais et al., 2014; Zhang and Pel, 2016).

In order to address the above-mentioned modeling requirements for synchromodality, we define the optimization problem as Synchromodal Transport Planning Problem with Flexible Services (STPP-FS). In STPP-FS, vehicles and requests are planned simultaneously, which allows the model to keep flexible during operations. The objective is to minimize the total cost, including transit cost, transfer cost, storage cost, carbon tax, waiting cost, and delay penalty. Besides typical constraints in the routing optimization, such as time windows and capacity constraints, the special constraints for ST are considered in STPP-FS, including constraints on transshipments, different modes, fixed and flexible vehicles, and complex schedules. An Adaptive Large Neighborhood Search (ALNS) heuristic is developed to solve the proposed problem efficiently. The proposed model allows flexible planning based on transport demands, which improves the utilization of available resources and reduces costs and emissions. To the best of our knowledge, this is the first paper that formulates the STPP-FS and develops a customized ALNS to solve it.

The remainder of this paper is organized as follows: Section 2 presents a brief literature review about articles related to synchromodal transport planning. Section 3 defines the problem in detail. The optimization problem is formulated by Mixed Integer Linear Programming in Section 4. Section 5 presents the solution methodology, i.e., a customized ALNS with a series of operators and performance improvement methods. In Section 6, experimental settings and results are provided. Section 7 concludes and gives future research directions.

2. Literature review

This section presents a review of the literature on the optimization models in synchromodal transport planning and the studies in the freight transportation domain that considers flexible vehicles and transshipments.

2.1. Optimization models in synchromodal transport planning

In the literature, containers in ST are moved by vehicles with fixed schedules (Guo et al., 2020; Agamez-Arias and Moyano-Fuentes, 2017; SteadieSeifi et al., 2014). These models can be divided into two groups: Minimum Cost Network Flow model (MCNF) and Path-based Network Design model (PBND) (Van Riessen et al., 2013). In MCNF, containers are transported over various links in the network and each link has capacity constraints. One branch of MCNF is the shipment matching problem, which defines links as services and matches these services and requests (Guo et al., 2020; Demir et al., 2016). For PBND models, the possible paths, i.e., subsequent links, are predetermined. Compared with MCNF, PBND reduces the number of decision variables and is more efficient. However, PBND loses some potential solutions and may need a higher cost than MCNF. Both MCNF and PBND assume that the services (links or paths) are predefined and obey fixed time schedules, which loses flexibility due to the following reasons:

1. These services routes (links or paths) are predefined depending on historical information, such as transport volume and decision makers' experience. Because it is not practical to keep all possible service options, some potential routes are neglected due to historical low demand when designing services, although they can serve current requests in a better way. Therefore, research on synchromodal routing is mostly limited to commodity flow formulations based on predefined services, and vehicle routing is usually ignored (Wolfinger et al., 2019).
2. The time schedules are fixed and vehicles' departure/arrival time may not fit the pickup/delivery time windows of requests, which may cause unnecessary waiting cost, transshipment cost, storage cost, and delay penalties. Moreover, strictly complying with predefined time schedules is not realistic because there are uncertainties and disturbances (Van Riessen et al., 2013).

Some scholars allow some flexibility in the model but only allow the flexible due/departure times (Demir et al., 2016; Van Riessen et al., 2013). They regard the problem as a Service Network Design Problem (SNDP) and uses MCNF, PBND, or a combination of MCNF and PBND to model it. Some papers allow flexible due times and charge delay penalties (Van Riessen et al., 2013; Ghane-Ezabadi and Vergara, 2016; Guo et al., 2020). Some other papers assume vehicles' departure times can be flexible in a defined time window (Moccia et al., 2011; Demir et al., 2016; Hrušovskỳ et al., 2018). Wolfinger et al. (2019) introduce a Multimodal Long Haul Routing Problem (MMLHRP) and they assume that the routes of trucks are flexible. Trucks are only used in the first- and last-mile and each request may use at most one long haul vehicle (ship or train) in the MMLHRP. Wolfinger et al. (2019) compare the multimodal and unimodal transport, and the benefits of flexible trucks are not evaluated. Qu et al. (2019) propose a re-planning model for ST and integrate shipment rerouting and service rescheduling. The re-planning model in Qu et al. (2019) benefits from two types of flexibility, i.e., split of shipments and buffer times on the departure of services. However, their model does not consider changing pre-planned service routes.

In practice, transport operators do not strictly follow the schedules because there are always new requests and delays, which makes some vehicles unavailable and needs other unplanned vehicles to serve the requests (Zhang and Pel, 2016). Moreover, there

are not always enough requests that make full use of the capacities of vehicles. Therefore, an optimization model with flexible services for ST is needed. To achieve flexible services, this study adds a vehicle routing component and these vehicles are highly dependent on each other due to flexibility, which leads to phenomena such as chain reactions that do not occur in MCNF and PBND. Furthermore, flexibility brings in computational complexity as the number of feasible services increases, and then a powerful and customized heuristic is needed. Therefore, the distinctions of our study compared with MCNF/PBND lie in having a mixed fleet of fixed and flexible vehicles in the research problem, a vehicle routing component in modeling, and a customized heuristic in the solution methodology.

2.2. Optimization models with flexible vehicles and transshipments in freight transport

Optimization models that consider flexible vehicles and transshipments in freight transportation are usually regarded as Pickup and Delivery Problem with Transshipment (PDPT). The PDPT is a variant of the Pickup and Delivery Problem (PDP), where requests can change vehicles at transshipment points during their trips (Shang and Cuff, 1996; Mitrović-Minić and Laporte, 2006; Masson et al., 2013). Qu and Bard (2012) use a Greedy Randomized Adaptive Search Procedure (GRASP) to generate the initial solution of PDPT for transport of aircraft and then use ALNS to improve the initial solution. Rais et al. (2014) propose several variants for PDPT, including cases with and without time windows, a heterogeneous fleet of vehicles, variable size fleet, split loads, and a limited number of transfer nodes visited by a vehicle. A small instance with seven requests is solved using Gurobi optimization software in their paper. Ghilas et al. (2016) integrate freight flows with scheduled public transportation services in short-haul transport, and the packages can be transferred between trucks and scheduled lines. Danloup et al. (2018) use both Large Neighborhood Search (LNS) and Genetic Algorithm (GA) to solve PDPT, where transport duration limitation is considered for requests and pickup/delivery time windows are ignored, therefore there is no time synchronization in their paper. Moreover, a request can be transshipped at most once, i.e., it cannot be served by more than two vehicles in their model.

Wolfinger and Salazar-González (2021) propose a branch-and-cut algorithm for solving PDP with split loads and transshipments (PDPSLT), however, the time window is not considered and the time synchronization method is not proposed. In another paper of Wolfinger (2021), the time window is considered and LNS is used to solve PDPSLT, and some insights regarding the benefits of combining split loads and transshipments are provided. In Wolfinger (2021)'s model, transshipment is allowed at dedicated transshipment locations and not allowed at customer locations.

The key feature of PDPT is the synchronization of activities among different vehicles. These synchronization requirements make routes interdependent (Drexl, 2013; Hojabri et al., 2018). For example, if a special request is inserted into a route of vehicle k and delayed request r served by vehicles k and l, all later requests in the route of vehicle l will also be delayed. Then these already scheduled requests need to be re-planned due to interconnections between routes.

2.3. Summary and contributions

This section compares the model developed in this paper with the existing studies in the literature in Table 1. In Table 1, all models are divided into three groups, i.e., models in ST (upper part), models in freight transport (lower part), and the proposed model (the last row). Almost all models consider transshipment but only part of them considers synchronizations among vehicles. The models in ST include multiple modes and fixed vehicles, while models in freight transport consider more flexibilities. In comparison, the model developed in this paper has synchronization requirements, flexible routes, and flexible schedules, which include flexible due time, flexible waiting time, flexible storage time, and flexible departure time.

Regarding the studies in freight transport, Ghilas et al. (2016)'s study seems similar to us. However, we establish models for different fields, and the sizes of transport networks are also different. They consider trucks and scheduled lines in urban transport. However, in our paper, services with different modes, including barges, trucks, and trains, are allowed to be used to transport containers in hinterland transport. Besides, the objectives of our mathematical models are different, which will influence the solutions significantly. While Ghilas et al. (2016) design more origin and destination nodes but a few transshipment nodes, all the nodes in our model can be transshipment nodes. These differences make the routes of vehicles in synchromodal transport more dependent on each other and it also makes the problem in ST more difficult to solve because it causes complicated chain reactions and heavy burdens on the computation time (see detailed explanations in Section 5.4). Compared with PDPT in the literature (Wolfinger, 2021; Danloup et al., 2018), there are some new characteristics in STPP-FS, such as more than two modes, a mix of fixed and flexible vehicles, and complex schedules. Moreover, these characteristics influence each other, which makes the transshipment and synchronization in STPP-FS more complex than PDPT.

In contrast, both Guo et al. (2020) and Demir et al. (2016) solve Synchromodal Transport Planning Problem (STPP) and consider the same modes with our study, including waterway, railway, and road. Guo et al. (2020) design the same objective function and Demir et al. (2016) only do not consider the storage cost compared with ours. In their studies, some links between two terminals are defined as services, and each service has a specific capacity, travel and service times, costs, and CO_2 emissions. The requests need to be picked up/delivered within specific time windows and can be transferred between services. Therefore, the most related articles are the studies of Guo et al. (2020) and Demir et al. (2016). However, there are still significant differences between the studies. The services in Guo et al. (2020) and Demir et al. (2016) are fixed and vehicle routing is not considered. In this paper, the possibility of using transshipments increases a lot due to flexible services, therefore the complexity of STPP-FS grows exponentially. Moreover, the vehicle routing and request routing need to be considered simultaneously in STPP-FS, which makes the modeling more complicated. Therefore, the modeling approach for STPP-FS is different from both models in synchromodal and freight transport.

Table 1
Comparison between the proposed model and existing models in the literature.

Article	Problem	Mode	Service	Objective	Heuristic	T	S	F	FR	FDue	FW	FS	FDep
Synchromodal transport													
Moccia et al. (2011)	SNDP	Railway, road	Link&path	c	BC	✓	✓	✓					✓
Van Riessen et al. (2013)	SNDP	Waterway, railway, road	Link&Path	c, t, d	–	✓		✓	✓	✓			
Ghane-Ezabadi and Vergara (2016)	SNDP	–	Path	c	DS	✓		✓		✓			
Demir et al. (2016)	STPP	Waterway, railway, road	Link	c, t, e, w, d	–	✓	✓	✓					✓
Hrušovskỳ et al. (2018)	SNDP	Waterway, railway, road	Link	c, t, e, w, d	–	✓	✓	✓					✓
Wolfinger et al. (2019)	MMLHRP	Waterway, railway, road	Vehicle	c	ILS	✓	✓	✓	✓		✓		✓
Qu et al. (2019)	SNDP	Waterway, railway, road	Link	c, t, s, d	–	✓	✓	✓					✓
Guo et al. (2020)	STPP	Waterway, railway, road	Link	c, t, s, e, w, d	PGFM	✓	✓	✓		✓		✓	
Freight transport													
Qu and Bard (2012)	PDPT	Road	Vehicle	c	GRASP&ALNS	✓	✓		✓		✓		✓
Rais et al. (2014)	PDPT	Road	Vehicle	c	–	✓	✓		✓	✓	✓		✓
Ghilas et al. (2016)	PDPTWSL	Metro, road	Vehicle	c, t	ALNS	✓	✓	✓	✓		✓	✓	✓
Danloup et al. (2018)	PDPT	Road	Vehicle	n, dis	LNS&GA	✓			✓				✓
Wolfinger and Salazar-González (2021)	PDPSLT	Road	Vehicle	c, t	–	✓			✓				
Wolfinger (2021)	PDPSLT	Road	Vehicle	c, t	LNS	✓	✓		✓		✓		✓
Our paper	STPP-FS	Waterway, railway, road	Vehicle	c, t, s, e, w, d	ALNS	✓	✓	✓	✓	✓✓✓	✓	✓	✓

–: not considered in the related paper.
T: Transshipment operations; S: Synchronization of operations; M: Multiple modes; F: Fixed vehicles; FR: Flexible routing; FDue: Flexible due time; FW: Flexible waiting time; FS: Flexible storage time; FDep: Flexible departure time; c, t, s, e, w, d, n, dis: transit cost, transfer cost, storage cost, carbon tax, waiting cost, delay penalty, number of used vehicles, distance; SNDP: Service Network Design Problem; MMLHRP: Multimodal Long Haul Routing Problem; PDPT: PDP with Transshipment; PDPTWSL: PDP with Time Windows and Scheduled Lines; PDPSLT: PDP with Split Load and Transshipment; STPP: Synchromodal Transport Planning Problem; STPP-FS: STPP with Flexible Services; BC: Branch-and-cut algorithm; DS: Decomposition-based Search; ILS: Iterated Local Search; PGFM: preprocessing heuristics of Path Generation and Feasible Matches; GRASP: Greedy Randomized Adaptive Search Procedure; LNS: Large Neighborhood Search; ALNS: Adaptive LNS; GA: Genetic Algorithm.

The main contributions of this paper are briefly summarized as follows. Firstly, we propose a mathematical model to provide a formulation of the STPP-FS. Fixed vehicles are restricted by predefined routes and time windows at terminals. On the contrary, flexible vehicles have flexible routes and schedules. Transshipment and synchronization of both fixed and flexible vehicles are considered. Only fixed, only flexible, or hybrid fleets can be handled by the proposed model. The proposed model with flexibilities necessitates an efficient solution algorithm as the solution space is large. To address this need, we develop a customized Adaptive Large Neighborhood Search (ALNS) heuristic algorithm. Therefore, the second contribution is the ALNS with specific adaptations and improvements, which include customized operators for ST, feasibility checking methods, and performance improvement approaches. Finally, we provide insights about the added value of flexibility in ST through computational experiments that compare the proposed approach to different benchmarks. In a nutshell, we design and validate a model that optimizes routes and schedules for fixed and flexible vehicles simultaneously in ST and can be used by freight forwarders and carriers for more economic and sustainable transport operations.

3. Problem description

We consider a setting with multiple shippers and a transport operator. The transport operator can be the freight forwarder, carrier, or transport platform in reality, and makes decisions on the routing and scheduling of vehicles (Li et al., 2015). The shippers provide the request information, including pickup and delivery terminals, number of containers, and time windows; and the transport operator provides transport network information, including terminal information, distances among terminals, vehicle information, and cost. The transport operator wants to optimize the transport operations and provide low-cost services to shippers (Di Febbraro et al., 2016). Moreover, the transport operator is assumed to be able to keep flexible schedules by collaborating with terminal operators. All travel times between terminals are known beforehand, except trucks' travel times which are influenced by traffic congestion. All costs are in Euros and the unit of containers is Twenty-foot Equivalent Unit (TEU).

The characteristics of the proposed STPP-FS include multiple modes, transshipment, the mix of fixed and flexible vehicles, complex schedules, and synchronization, as shown in Fig. 2.

1. Multiple modes. In ST, trucks, barges, and trains are used to serve requests, and one request can be served using any combination of these modes. Different modes have different parameters on capacity, speed, costs, and emissions. Barges usually have the lowest emissions and costs but have the slowest travel speed. Trains have a moderate speed and cost. We assume that a truck service is a truck fleet and each truck can serve requests in the fastest way when containers arrive. Trucks are the fastest vehicles but the transportation cost is higher than trains and barges.
2. Transshipment. A request can be transferred between vehicles at transshipment terminals, which can provide transshipment equipment and a yard for the temporary storage of containers. Typically a transshipment terminal has functions of regular terminals, therefore it can also be pickup/delivery terminals. Different transshipment terminals provide different types of services, such as transshipments between barges and trucks. Transshipments between vehicles with the same mode but

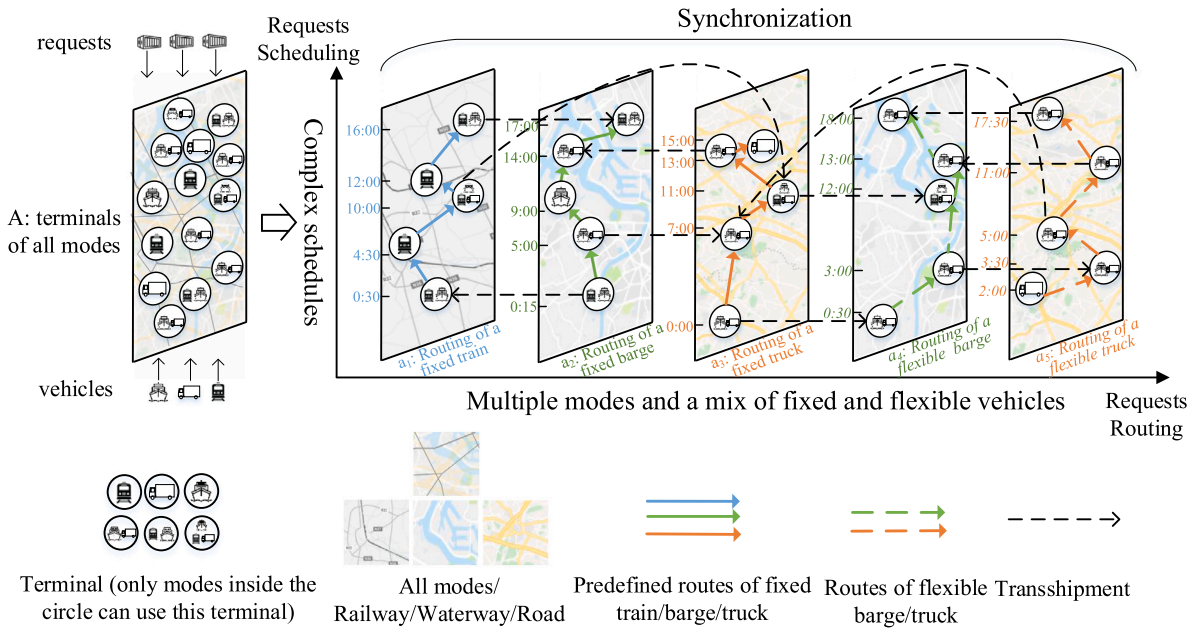

Fig. 2. Characteristics of the STPP-FS. To illustrate the problem clearly, the real-life transport network (layer A) is decomposed to five layers (a_1, a_2, a_3, a_4, a_5), which are routing and scheduling of five types of vehicles. The main horizontal and vertical axes are routing and scheduling of requests, respectively. The vehicles and requests are planned simultaneously in this study.

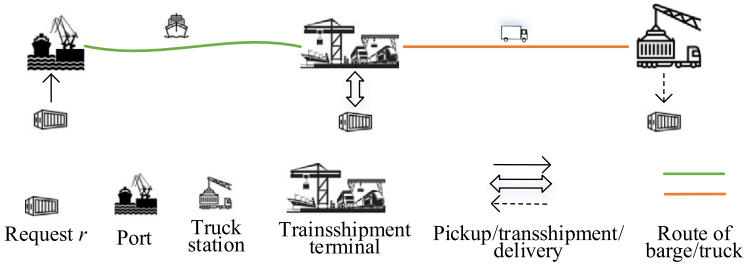

Fig. 3. Transshipment.

different time schedules are also possible. Compared with ST with only fixed vehicles, the flexible services increase the possibility of transshipments significantly. In Fig. 3, request r is transported by a barge firstly and then transferred to a truck fleet. At the transshipment terminal, the barge should arrive earlier than trucks. When trucks arrive at the transshipment terminal before the barge completes the unloading, trucks will wait for the barge.

3. The mix of fixed and flexible vehicles. In ST, trucks and barges may be flexible while trains are fixed. Fixed vehicles can only run between predefined terminals, and the departure time and arrival time are also predefined. Flexible vehicles can go to any terminal (on available routes, such as waterways for barges) and have no predefined schedules. Therefore, there are five types of vehicles, i.e., fixed barges, trains, and trucks, and flexible barges and trucks, as shown in Fig. 4. In Fig. 4, there are three terminals, i.e., terminal A, B, and C, and two requests, i.e., requests r_1 and r_2, which are transported in different ways by five types of vehicles. Request r_1's pickup terminal and transshipment terminal are terminals A and B. Request r_2's transshipment terminal and delivery terminal are terminals B and C. When requests are transported by the fixed barge, two barges are needed, i.e. barge k_1 from terminal A to B for request r_1 and barge k_2 from terminal B to C for request r_2. When the barge is flexible, only one barge k_1 is needed and k_1 starts at A and goes through B to C. Another flexible barge k_2 could go to other terminals and transport other requests. The case with fixed trains l_1 and l_2 is similar to fixed barges. Regarding truck fleet, one request is usually served by multiple trucks, and each truck transports one container. The difference between fixed and flexible truck fleets is similar to barges.
4. Complex schedules. In the scheduling of ST, the waiting time, storage and delay need to be considered. If a vehicle arrives before containers at the pickup terminal or transshipment terminal, it can wait until containers arrive. If containers arrive before vehicles, they can be temporarily stored in the terminal with a storage fee until the vehicle arrives. If a vehicle is delayed at the delivery terminal, it will be charged with a delay penalty. The transshipment makes the waiting time and delay

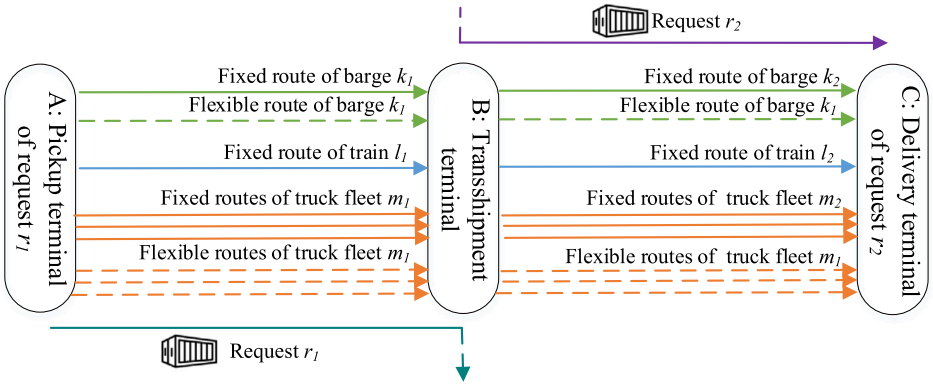

Fig. 4. Five types of vehicles.

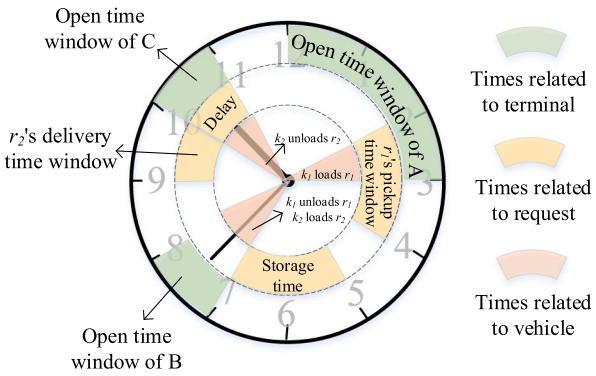

Fig. 5. Complex schedules.

of one vehicle influenced by another vehicle, and the waiting and storage time could be changed to synchronize vehicles. Moreover, the mix of fixed and flexible vehicles and transshipments between different modes also make the scheduling more complicated. Fig. 5 shows the schedules of fixed barges k_1 and k_2 in Fig. 4. Containers must be loaded/unloaded in the open time windows for vehicles at each terminal. At terminal A, the open time window for vehicle k_1 is $[1,3]$, and request r_1's pickup time window is $[2,4]$. Therefore, request r_1 will be picked up at time 2. Request r_2 arrives at terminal B at time 5. However, vehicle k_1 has not arrived and request r_2 needs to wait for vehicle k_1 at terminal B. Therefore, request r_2 is stored at terminal B until time 7. During the open time window ($[7,8]$) at terminal B, k_1 unloads r_1 and k_2 loads r_2. At terminal C, k_2 arrives later than the delivery time window of request r_2, which causes one hour's delay. If requests are transported by flexible vehicles, these unnecessary storage and delay could be avoided, but the overall schedule will be more complex because the schedule of one vehicle will influence the schedule of another vehicle.

5. Synchronization. Because of the transshipment, changes in a vehicle's route may affect another vehicle's route. Such influences might trigger a chain reaction in all routes, which may make the original plan infeasible. When vehicles influence each other, synchronization between vehicles is required in ST. Specifically, the synchronization coordinates vehicles and minimizes the changes to the original plan. The complex schedules also complicate the time synchronization between vehicles. As shown in Fig. 6, there are three requests (r_2, r_3, and r_4) that are served by a flexible barge, a flexible truck fleet, and a train with fixed services. The requests are transferred between these three services at two transshipment terminals and a vehicle may transport more than one request at the same time. Request r_2 is transferred twice, i.e., from barge to truck and then to train. When a new request r_1 is inserted into the barge's route, not only the barge's schedule is influenced, but also the train's schedule is influenced due to the transshipment of request r_2. Moreover, the barge and train's schedules are also influenced by the changes in the truck's schedule due to the transshipment of requests r_2 and r_4 at another transshipment terminal.

4. Mathematical model

This section presents the mathematical model to formulate the STPP-FS. The notation used in the mathematical model is provided in Table 2. There are multiple modes $w \in W$ in a transport network. The transport network is defined as a directed graph $G = (N, A)$, where N represents the set of terminals (ports and train/truck stations) and $A = \{(i,j)|i,j \in N, i \neq j\}$ represents the set of arcs

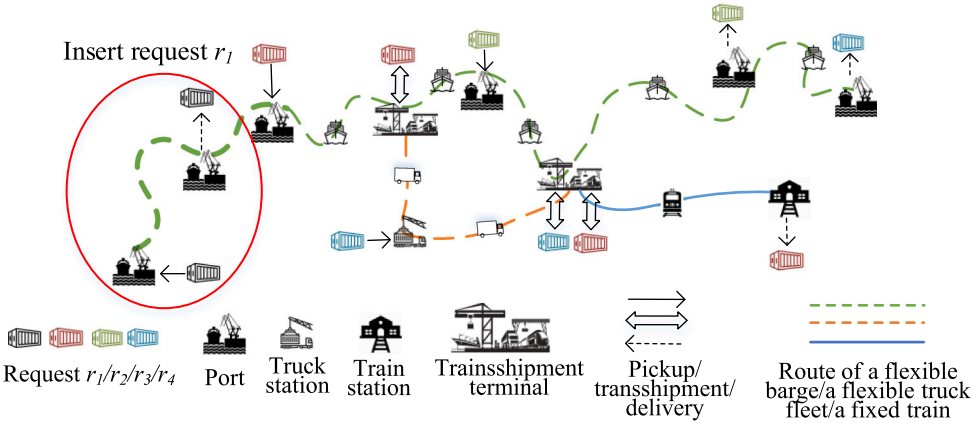

Fig. 6. Synchronization.

(roads, railways, and inland waterways). $P, D, T, O \subseteq N$ are sets of pickup terminals, delivery terminals, transshipment terminals, and depots of vehicles. In ST, the same terminal could belong to all P, D, T, O sets and be accessed by all transport modes. The nonnegative travel time τ_{ij}^k equals distance between i and j divided by speed v_k of vehicle k. Note that distances are different for different modes because different modes use different routes between i and j. Moreover, the travel time τ'^{kr}_{ij} of trucks is considered time-dependent, which means travel time at peak periods will be longer than non-peak periods due to traffic congestion (Guo et al., 2020).

The unit of capacity of vehicles is TEU. Barges and trains have fixed capacities and a truck's capacity is 1 TEU. We assume that each truck fleet has an unlimited number of trucks. The pickup and delivery terminals of request $r \in R$ are designated by $p(r)$ and $d(r)$. Let $o(k)$ and $o'(k)$ represent the starting and the ending depot of vehicle $k \in K$. Some depots may be the same terminals with pickup/delivery terminals, which makes some constraints of pickup/delivery terminals, such as time window constraints, will also work on these depots when the related vehicle does not serve the request. Dummy depots $\overline{o}(k)$ and $\overline{o'}(k)$ are therefore created. Fixed vehicles can only go to terminals in predefined routes, and there is an open time window at each terminal $[a_i^k, b_i^k]$, in which fixed vehicles can load/unload containers.

A solution of the STPP-FS is a set of $|K|$ routes that serve all requests and route k starts and ends at (dummy) depots. At any moment, the number of containers carried simultaneously by vehicle k cannot exceed capacity u_k. For every request r, terminals $p(r)$ and $d(r)$ can be served by the same vehicle k, which means $d(r)$ being served after $p(r)$. Terminals $p(r)$ and $d(r)$ can also be served by distinct vehicles $k_1 \in K$ and $k_2 \in K$, and r is transferred from k_1 to k_2, which means vehicle k_2 must start its service at the transshipment terminal after vehicle k_1 which unloads the containers. Request r needs to be picked up in time window $[a_{p(r)}, b_{p(r)}]$ and delivered in time window $[a_{d(r)}, b_{d(r)}]$, but the delivery time can exceed $b_{d(r)}$ with a delay penalty. Vehicle k is allowed to wait for containers at terminal i and request r is allowed to be stored at terminal i when the vehicle has not arrived.

The objective of the proposed STPP-FS is minimizing cost (Euros), which consists of transit cost (F_1), transfer cost (F_2), storage cost (F_3), carbon tax (F_4), waiting cost (F_5), and delay penalty (F_6), as shown in Eqs. (1)–(7) (Guo et al., 2020). The emissions are calculated using an activity-based method by Demir et al. (2016) and the amount of emissions is related to vehicle type, distance, and amount of containers.

$$\min \ F = F_1 + F_2 + F_3 + F_4 + F_5 + F_6 \tag{1}$$

$$F_1 = \sum_{k\in K}\sum_{(i,j)\in A}\sum_{r\in R}(c_k^1\tau_{ij}^k + c_k^{1'}d_{ij}^k)q_r y_{ij}^{kr} \tag{2}$$

$$F_2 = \sum_{k,l\in K,k\neq l}\sum_{r\in R}\sum_{i\in T}(c_k^2 + c_l^2)q_r s_{ir}^{kl} + \sum_{k\in K}\sum_{(i,j)\in A_p}\sum_{r\in R}c_k^2 q_r y_{ij}^{kr} + \sum_{k\in K}\sum_{(i,j)\in A_d}\sum_{r\in R}c_k^2 q_r y_{ij}^{kr} \tag{3}$$

$$F_3 = \sum_{k,l\in K,k\neq l}\sum_{r\in R}\sum_{i\in T}c_k^3 q_r s_{ir}^{kl}(t'^{lr}_i - \overline{t}_i^{kr}) + \sum_{k\in K}\sum_{(i,j)\in A_p}\sum_{r\in R}c_k^3 q_r y_{ij}^{kr}(t'^{kr}_i - a_{p(r)}) \tag{4}$$

$$F_4 = \sum_{k\in K}\sum_{(i,j)\in A}\sum_{r\in R}c_k^4 e_k q_r d_{ij}^k y_{ij}^{kr} \tag{5}$$

$$F_5 = \sum_{k\in K_{\mathrm{b\&t}}}\sum_{i\in N}c_k^5 t_{ki}^{\mathrm{wait}} \tag{6}$$

$$F_6 = \sum_{r\in R}c_r^{\mathrm{delay}} q_r t_r^{\mathrm{delay}} \tag{7}$$

Constraints (8)–(26) are the spatial constraints and Constraints (27)–(47) are the time-related constraints.

Table 2
Notation.

Sets:	
W	Set of modes indexed by w.
R	Set of requests indexed by r.
N	Set of terminals indexed by i and j. $O/\overline{O} \subseteq N$, set of depots/dummy depots. $P/D/T \subseteq N$, set of pickup/delivery/transshipment terminals. $T_{w_1}^{w_2}$, set of terminals allowing transshipments between mode w_1 and mode w_2.
K	Set of vehicles indexed by k and l. $K_{\text{b\&t}} \subseteq K$, set of barges and trains. $K_{\text{truck}} \subseteq K$, set of truck fleets. $K_w \subseteq K$, set of vehicles of mode w. $K_{\text{fix}} \subseteq K$, set of fixed vehicles.
A	Set of arcs. For $i, j \in N$, the arc from i to j is denoted by $(i,j) \in A$. $A_p/A_d \subseteq A$ represents the set of pickup/delivery arcs. For $(i,j) \in A_p$, $i \in P$. For $(i,j) \in A_d$, $j \in D$. $A_w \subseteq A$ represents the set of arcs for mode w. $A_{\text{fix}}^k \subseteq A$ represents the set of arcs for a fixed vehicle $k \in K_{\text{fix}}$.
Parameters:	
u_k	Capacity (TEU) of vehicle k.
q_r	Quantity (TEU) of request r.
τ_{ij}^k	The travel time (in hours) on arc (i,j) for vehicle k.
$[a_{p(r)}, b_{p(r)}]$	The pickup time window for request r.
$[a_{d(r)}, b_{d(r)}]$	The delivery time window for request r.
$[a_i^k, b_i^k]$	The open time window for fixed vehicle k at terminal i.
${t''}_i^k$	The loading (or unloading) time (in hours) for vehicle k at terminal i.
v_k	Speed (km/h) of vehicle k.
d_{ij}^k	Distance (km) between terminals i and j for vehicle k.
e_k	The CO_2 emissions (kg) per container per km of vehicle $k \in K$.
c_k^n	Unit cost of different terms, $n \in \{1, 1', 2, 3, 4, 5, 6\}$. All costs are in Euros. $c_k^1/c_k^{1'}$ are transport cost per hour/km per container using vehicle $k \in K$. c_k^2 is the loading (or unloading) cost per container. c_k^3 is the storage cost per container per hour. c_k^4 is the carbon tax coefficient per ton. c_k^5 is the cost per hour of waiting time.
c_r^{delay}	The delay penalty per container per hour of request r.
t_b	The b_{th} breakpoint of time-dependent travel time functions of trucks, $b \in \{1, 2, ..., B\}$, B is the number of breakpoints.
T_m	The m_{th} time period within a day, $T_m = [t_m, t_{m+1}]$, $m \in \{1, 2, ..., B-1\}$.
θ_m	The slope of the travel time function for time period T_m.
η_m	The intersection of the travel time function for time period T_m.
M	A large enough positive number.
Variables:	
x_{ij}^k	Binary variable; 1 if vehicle k uses the arc (i,j), 0 otherwise.
y_{ij}^{kr}	Binary variable; 1 if request r transported by vehicle k uses arc (i,j), 0 otherwise.
z_{ij}^k	Binary variable; 1 if terminal i precedes (not necessarily immediately) terminal j in the route of vehicle k, 0 otherwise.
s_{ir}^{kl}	Binary variable; 1 if request r is transferred from vehicle k to vehicle $l \neq k$ at transshipment terminal i, 0 otherwise.
$t_i^{kr}/{t'}_i^{kr}/\bar{t}_i^{kr}$	The arrival time/service start time/service finish time of request r served by vehicle k at terminal i.
$t_i^k/{t'}_i^k/\bar{t}_i^k$	The arrival time/last service start time/departure time of vehicle k at terminal i.
t_{ki}^{wait}	The waiting time of vehicle k at terminal i.
t_r^{delay}	The delay time of request r at delivery terminal.
$\tilde{t}_i^{kr}$	Normalized departure time of truck $k \in K_{\text{truck}}$ with request r at terminal i, $0 \leqslant \tilde{t}_i^{kr} \leqslant 24$.
${\tau'}_{ij}^{kr}$	The time-dependent travel time (in hours) on arc (i,j) for truck $k \in K_{\text{truck}}$ with request r.
n_i^{kr}	An integer variable used for normalizing departure time of truck $k \in K_{\text{truck}}$ with request r at terminal i.
ζ_{irk}^b	A continuous variable used for linearizing the time-dependent travel time function of truck $k \in K_{\text{truck}}$, $0 \leqslant \zeta_{irk}^b \leqslant 1$, $r \in R$, $i \in N$, and b means the b_{th} breakpoint of time-dependent travel time function.
ξ_{irk}^m	A binary variable used for linearizing the time-dependent travel time function of truck $k \in K_{\text{truck}}$, $r \in R$, $i \in N$, and m means the m_{th} time period within a day.

Constraints (8)–(15) are typical constraints in PDP. Constraints (8) and (9) ensure that a vehicle begins and ends at its begin and end depot, respectively. Constraints (8)–(9) are modified from Rais et al. (2014), and these constraints only limit the routes of barges and trains because each truck service is considered as a fleet of trucks which might have different routes. Constraints (10)–(12) are the subtour elimination constraints and provide tight bounds among several polynomial-size versions of subtour elimination constraints (Öncan et al., 2009). Constraints (13) and (14) ensure that containers for each request must be picked and delivered at its pickup and delivery terminal, respectively. Constraints (15) are the capacity constraints.

$$\sum_{j \in N} x_{o(k)j}^k \leqslant 1 \quad \forall k \in K_{\text{b\&t}} \tag{8}$$

$$\sum_{j \in N} x_{o(k)j}^k = \sum_{j \in N} x_{j\overline{o'}(k)}^k \quad \forall k \in K_{\text{b\&t}} \tag{9}$$

$$x_{ij}^k \leqslant z_{ij}^k \quad \forall i, j \in N, \ \forall k \in K_{\text{b\&t}} \tag{10}$$

$$z_{ij}^k + z_{ji}^k = 1 \quad \forall i,j \in N,\ \forall k \in K_{\text{b\&t}} \tag{11}$$

$$z_{ij}^k + z_{jp}^k + z_{pi}^k \leqslant 2 \quad \forall i,j,p \in N,\ \forall k \in K_{\text{b\&t}} \tag{12}$$

$$\sum_{k\in K}\sum_{j\in N} y_{p(r)j}^{kr} = 1 \quad \forall r \in R \tag{13}$$

$$\sum_{k\in K}\sum_{i\in N} y_{id(r)}^{kr} = 1 \quad \forall r \in R \tag{14}$$

$$\sum_{r\in R} q_r y_{ij}^{kr} \leqslant u_k x_{ij}^k \quad \forall (i,j) \in A,\ \forall k \in K \tag{15}$$

Constraints (16) and (17) facilitate transshipment. Constraints (16) ensure that the transshipment occurs only once per transshipment terminal. Constraints (17) forbid transshipment between the same vehicle k.

$$\sum_{j\in N} y_{ji}^{kr} + \sum_{j\in N} y_{ij}^{lr} \leqslant s_{ir}^{kl} + 1 \quad \forall r \in R,\ \forall i \in T,\ \forall k,l \in K \tag{16}$$

$$s_{ir}^{kk} = 0 \quad \forall r \in R,\ \forall i \in T,\ \forall k \in K \tag{17}$$

Flow conservation constraints of both vehicles and requests are handled by Constraints (18)–(23). Constraints (18) represent flow conservation for vehicle flow and (19)–(22) represent flow conservation for request flow. Constraints (19) are for regular terminals and Constraints (20) are for transshipment terminals. Constraints (21) and (22) ensure the flow conservation of requests when vehicle k passes the transshipment terminal but no transfer happens. Constraints (21) and (22) consider a special case in STPP-FS, where request r is not transferred at terminal $i \in T$ but vehicle k passes terminal i due to operations for other requests. Constraints (23) link y_{ij}^{kr} and x_{ij}^k variables to guarantee that for a request to be transported by a vehicle, that vehicle needs to traverse the associated arc.

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{ji}^k = 0 \quad \forall k \in K_{\text{b\&t}},\ \forall i \in N \setminus \overline{o}(k), \overline{o'}(k) \tag{18}$$

$$\sum_{j\in N} y_{ij}^{kr} - \sum_{j\in N} y_{ji}^{kr} = 0 \quad \forall k \in K,\ \forall r \in R,\ \forall i \in N \setminus T, p(r), d(r) \tag{19}$$

$$\sum_{k\in K}\sum_{j\in N} y_{ij}^{kr} - \sum_{k\in K}\sum_{j\in N} y_{ji}^{kr} = 0 \quad \forall k \in K,\ \forall r \in R,\ \forall i \in T \setminus p(r), d(r) \tag{20}$$

$$\sum_{j\in N} y_{ij}^{kr} - \sum_{j\in N} y_{ji}^{kr} \leqslant \sum_{l\in K} s_{ir}^{lk} \quad \forall k \in K,\ \forall r \in R,\ \forall i \in T \setminus p(r), d(r) \tag{21}$$

$$\sum_{j\in N} y_{ji}^{kr} - \sum_{j\in N} y_{ij}^{kr} \leqslant \sum_{l\in K} s_{ir}^{kl} \quad \forall k \in K,\ \forall r \in R,\ \forall i \in T \setminus p(r), d(r) \tag{22}$$

$$y_{ij}^{kr} \leqslant x_{ij}^k \quad \forall (i,j) \in A,\ \forall k \in K,\ \forall r \in R \tag{23}$$

Characteristics of ST are considered in Constraints (24)–(26). Constraints (24) avoid vehicles running on unsuitable routes, for example, the truck cannot run on inland waterways. Constraints (25) take care of predefined routes for certain vehicles. Constraints (26) ensure the transshipment occurs in the right transshipment terminal because some transshipment terminals only allow the transshipment between two specific modes. When the containers need to be transferred from barges to trucks, terminals that only allow transshipment between barges and trains will not be considered. Constraints (24)–(26) are unique to this model because they consider the characteristics of vehicle routing in ST.

$$x_{ij}^k = 0 \quad \forall k \in K_w,\ \forall (i,j) \in A \setminus A_w,\ \forall w \in W \tag{24}$$

$$x_{ij}^k = 0 \quad \forall k \in K_{\text{fix}},\ \forall (i,j) \in A \setminus A_{\text{fix}}^k \tag{25}$$

$$s_{ir}^{kl} = 0 \quad \forall k \in K_{w_1},\ \forall l \in K_{w_2},\ \forall i \in T \setminus T_{w_1}^{w_2},\ \forall r \in R,\ \forall w_1,w_2 \in W \tag{26}$$

Constraints (27)–(31) are time constraints related to services, which are necessary for both fixed and flexible services. Constraints (27) guarantee that service start time is later than the arrival time of containers. Constraints (28) ensure that the service finish time equals service start time plus service time. Constraints (29) maintain that the departures happen only after all services are completed. Constraints (30) ensure that the request's arrival time cannot be earlier than the vehicle's arrival time. Constraints (31) define the vehicle's last service start time.

$$t_i^{kr} \leqslant t'^{kr}_i \quad \forall i \in N,\ \forall k \in K,\ \forall r \in R \tag{27}$$

$$t'^{kr}_i + t''^{kr}_i \sum_{j\in N} y_{ij}^{kr} \leqslant \bar{t}_i^{kr} \quad \forall i \in N,\ \forall k \in K_{\text{b\&t}},\ \forall r \in R \tag{28}$$

$$\bar{t}_i^k \geqslant \bar{t}_i^{kr} \quad \forall i \in N,\ \forall k \in K_{\text{b\&t}},\ \forall r \in R \tag{29}$$

$$t_i^k \leqslant t_i^{kr} \quad \forall i \in N,\ \forall k \in K_{\text{b\&t}},\ \forall r \in R \tag{30}$$

$$t'^k_i \geqslant t'^{kr}_i \quad \forall i \in N,\ \forall k \in K_{\text{b\&t}},\ \forall r \in R \tag{31}$$

Constraints (32) and (33) ensure that the time on the route of barges or trains is consistent with the distance traveled and speed. When waiting time is not considered, Constraints (32) are enough to take care of arrival time t_j^k. Since vehicles should wait at terminals in this study, Constraints (33) need to be added to restrict travel time tightly and avoid wrongly adding waiting times to t_j^k ($t_j^k > \bar{t}_i^k + \tau_{ij}^k$). Constraints (34) and (35) take care of the time windows for pickup terminals and fixed terminals, respectively.

$$\bar{t}_i^k + \tau_{ij}^k - t_j^k \leqslant M(1 - x_{ij}^k) \quad \forall (i,j) \in A,\ \forall k \in K_{\text{b\&t}} \tag{32}$$

$$\bar{t}_i^k + \tau_{ij}^k - t_j^k \geqslant -M(1 - x_{ij}^k) \quad \forall (i,j) \in A,\ \forall k \in K_{\text{b\&t}} \tag{33}$$

$$t'^{kr}_{p(r)} \geqslant a_{p(r)} y_{ij}^{kr},\ \bar{t}^{kr}_{p(r)} \leqslant b_{p(r)}(y_{ij}^{kr} + M(1 - y_{ij}^{kr})) \quad \forall (i,j) \in A, \forall r \in R,\ \forall k \in K \tag{34}$$

$$t_i^{kr} \geqslant a_i^k y_{ij}^{kr},\ \bar{t}_i^{kr} \leqslant b_i^k (y_{ij}^{kr} + M(1 - y_{ij}^{kr})) \quad \forall (i,j) \in A,\ \forall r \in R,\ \forall k \in K_{\text{fix}} \tag{35}$$

Constraints (36) are time constraints for transshipment. If there is a transshipment from vehicle k to vehicle l, but vehicle l arrives before vehicle k departs, vehicle l can wait until vehicle k completes its unloading. Constraints (37) and (38) calculate waiting time and delay time, respectively, and these constraints are used to reduce waiting times and delay costs.

$$\bar{t}_i^{kr} - t'^{lr}_i \leqslant M(1 - s_{ir}^{kl}) \quad \forall r \in R,\ \forall i \in T,\ \forall k,l \in K,\ k \neq l \tag{36}$$

$$t_{ki}^{\text{wait}} \geqslant t'^{k}_i - t_i^k \quad \forall i \in N,\ \forall k \in K_{\text{b\&t}} \tag{37}$$

$$t_r^{\text{delay}} \geqslant (\bar{t}_{d(r)}^{kr} - b_{d(r)}) \sum_{i \in N} y_{id(r)}^{kr} \quad \forall r \in R,\ \forall k \in K \tag{38}$$

Constraints (39) to (46) are imposed to linearize the time-dependent travel time functions of trucks and Constraints (47) take care of the arrival time of trucks (Lin et al., 2013; Guo et al., 2020).

$$\tilde{t}_i^{kr} = \bar{t}_i^{kr} - 24 n_i^{kr} \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R \tag{39}$$

$$\tilde{t}_i^{kr} = \sum_{b \in \{1,2...,B\}} \zeta_{irk}^b t_b \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R \tag{40}$$

$$\sum_{b \in \{1,2...,B\}} \zeta_{irk}^b = 1 \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R \tag{41}$$

$$\sum_{m \in \{1,2...,B-1\}} \xi_{irk}^m = 1 \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R \tag{42}$$

$$\zeta_{irk}^1 \leqslant \xi_{irk}^1 \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R \tag{43}$$

$$\zeta_{irk}^B \leqslant \xi_{irk}^{B-1} \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R \tag{44}$$

$$\zeta_{irk}^b \leqslant \xi_{irk}^{b-1} + \xi_{irk}^b \quad \forall k \in K_{\text{truck}},\ \forall i \in N,\ \forall r \in R,\ \forall b \in \{2,3...,B-1\} \tag{45}$$

$$\tau'^{kr}_{ij} = \zeta_{irk}^1(\theta_1 t_1 + \eta_1) + \sum_{b \in \{2...,B\}} \zeta_{irk}^b(\theta_{b-1} t_b + \eta_{b-1}) \quad \forall k \in K_{\text{truck}},\ \forall r \in R,\ \forall (i,j) \in A \tag{46}$$

$$(t_j^{kr} - \bar{t}_i^{kr}) y_{ij}^{kr} = \tau'^{kr}_{ij} \quad \forall (i,j) \in A,\ \forall k \in K_{\text{truck}},\ \forall r \in R \tag{47}$$

Compared with studies that model services as links and paths and ignore vehicle routing (Moccia et al., 2011; Van Riessen et al., 2013; Ghane-Ezabadi and Vergara, 2016; Demir et al., 2016; Hrušovskỳ et al., 2018; Guo et al., 2020), the vehicles and requests in this study are planned simultaneously by the vehicle routing component (constraints related to x_{ij}^k variable), requests routing component (constraints related to y_{ij}^{kr} variable), and the relations between these two components (such as constraints related to the transshipment variable s_{ir}^{kl}). These components enable the proposed model to explore routes that are not defined in advance.

As Constraints (25) and (35) do not work on flexible vehicles, the number of alternatives is significantly larger than the case of MCNF/PDND models in the literature (Van Riessen et al., 2013; Demir et al., 2016; Guo et al., 2020). It makes the feasible region of the proposed STPP-FS very large and the problem hard to solve. However, some parts of the feasible region can be cut without losing any feasible solutions by using so-called valid inequalities (Cornuéjols, 2008). We propose a novel set of valid inequalities (see Appendix A), which are divided into three categories, i.e., valid inequalities related to requests, vehicles, and transshipments.

5. Adaptive large neighborhood search

Due to the computational complexity, we develop an Adaptive Large Neighborhood Search (ALNS) heuristic to solve the proposed problem. In the literature, ALNS was proposed to solve PDP based on an extension of the LNS heuristic, which obtains the best solution by using removal and insertion operators to destroy and repair routes iteratively (Ropke and Pisinger, 2006). To solve the STPP-FS in this research, we adapt the traditional operators and design new ones in ALNS considering characteristics of ST. Compared with ALNS for PDPT in the literature (Qu and Bard, 2012; Masson et al., 2013; Wolfinger, 2021), the innovations of the proposed ALNS are (a) getting the initial solutions by multiple methods (Section 5.1), (b) customizing operators considering the characteristics of ST (Section 5.2), (c) providing feasibility check methods for fixed and flexible vehicles and synchronization methods for interdependent vehicles (Section 5.3), and (d) using several performance improvement methods (Appendix C).

The pseudocode of the designed ALNS is shown in Algorithm 1. The adaptive mechanism of ALNS is illustrated in detail in our previous paper (Zhang et al., 2020, 2022) and not repeated in this paper.

Algorithm 1: ALNS algorithm

Input: K, R, N, A, X_{current}; **Output:** X_{best} ; // $X_{\text{current}}/X_{\text{best}}$ means the current/best solution.
$[K, R, N, A] = Preprocessing(K, R, N, A)$;
define the set of unserved requests as R_{pool} ; // R_{pool} represents the request pool.
obtain initial solution X_{initial}; set $T_{\text{Temp}} > 0$ depending on X_{initial};
$X_{\text{last}} \leftarrow X_{\text{initial}}$; $X_{\text{best}} \leftarrow X_{\text{last}}$; // X_{last} means the last solution.
repeat
  refresh weights and choose operators depending on weights at the beginning of each segment;
  $X_{\text{current}} \leftarrow X_{\text{last}}$; $[X_{\text{current}}, R_{\text{pool}}] = RemovalOperator(X_{\text{current}}, R_{\text{pool}})$; $flag = False$;
  while R_{pool} *is not empty* **do**
    if $flag == True$ **then**
      $[X_{\text{current}}, R_{\text{pool}}] = RemovalOperator(X_{\text{current}}, R_{\text{pool}})$
    else
      $flag = True$
    end
    $[X_{\text{current}}, R_{\text{pool}}] = InsertionOperator(X_{\text{current}}, R_{\text{pool}})$;
    if *insertion operator is a greedy type* **then**
      $[X_{\text{current}}, R_{\text{pool}}] = BundleInsertion(X_{\text{current}}, R_{\text{pool}})$
    end
  end
  $[X_{\text{current}}, R_{\text{pool}}] = SwapOperator(X_{\text{current}}, R_{\text{pool}})$;
  if $F(X_{current}) < F(X_{last})$ **then**
    $X_{\text{last}} \leftarrow X_{\text{current}}$;
  else
    $X_{\text{last}} \leftarrow X_{\text{current}}$ with probability $p = e^{\frac{-(F(X_{\text{current}})-F(X_{\text{last}}))}{T_{\text{Temp}}}}$; // Update X_{last} based on the simulated annealing idea (Ropke and Pisinger, 2006).
  end
  if $F(X_{last}) < F(X_{best})$ **then**
    $X_{\text{best}} \leftarrow X_{\text{last}}$;
  end
  $T_{\text{Temp}} \leftarrow T_{\text{Temp}} \cdot c$; // c is the cooling rate.
until *the predefined number of iterations is reached*;

5.1. Initial solution

In the literature, the initial solution is usually obtained from insertion operators from scratch (Ropke and Pisinger, 2006; Qu and Bard, 2012; Wolfinger, 2021). However, in ST, planning from scratch may cause significant changes in the predefined schedules. The transport operator may not be able to make significant changes due to external factors, such as port schedules and reliable services required by shippers. Moreover, the more flexibility the optimization problem has, the harder it is to solve. The optimization based on fixed schedules may reduce the complexities brought by flexibility. Therefore, the predefined fixed schedules could be taken into account when designing the initial solution. We designed different methods to obtain the initial solution:

1. Method R: By the Regret Insertion operator from scratch. This method does not consider predefined fixed schedules.
2. Method S: Using the best solution of the problem with fewer flexibilities, e.g., the best solution when all vehicles are fixed is used as an initial solution for the problem with flexible trucks. This method obtains the solution of full flexibility step by step and the complexity of the problem with flexible vehicles is reduced.
3. Method M: Using the optimal solution of a predefined fixed schedule and the optimal solution is obtained by the matching model proposed by Guo et al. (2020). There are two differences between method M and method S: (a) method M always uses the solution without flexible vehicle as the initial solution for different flexibility levels, while method S uses the solution of fewer flexibilities which may have some flexible vehicles; (b) method S's initial solution is obtained by ALNS itself and it may be sub-optimal, while method M uses the optimal solution of a predefined fixed schedule.

5.2. Operators in ALNS

There are many different operators in the literature (Grangier et al., 2016; Sarasola and Doerner, 2020; Qu and Bard, 2012; Liu et al., 2019; Wolfinger, 2021). Choosing the insertion and removal operators not only needs to consider features of the studied problem but also the balance between exploitation and exploration. How to use the historical experience and predict the future reward also need to be considered. In this work, Transshipment Insertion and Node Removal operators consider the transshipments and specific modes (waterway and railway) in ST. The Greedy Insertion, Most Constrained First insertion, and Worst Removal operators are used for exploiting. Random Insertion, Random Removal, Route Removal, and Related Removal (also called Shaw Removal) operators are responsible for exploring. History Removal and Regret Insertion operators use the historical experience and

predict future situations, respectively. Besides insertion and removal operators, a novel Swap operator is proposed to make up for the disadvantages of using an insertion operator or a removal operator alone.

Some operators have been reported in our previous paper (Zhang et al., 2020, 2022), including Greedy Insertion, Transshipment Insertion, Random Insertion, Worst Removal and Random Removal. Related Removal operator is widely discussed in the literature (Ropke and Pisinger, 2006; Danloup et al., 2018). The following sections introduce customized operators in detail and the others are introduced briefly.

5.2.1. Insertion operators

All insertion operators have two basic operations, i.e., inserting one request to one route or multiple routes. When a request is inserted into one route of vehicle k, k will finish both pickup and delivery and there is no transshipment. When a request is inserted into multiple routes, firstly the request is segmented into multiple by potential transshipment terminals and then each will be served by one vehicle, and containers are transferred between vehicles.

Greedy Insertion operator tries all possible solutions using one vehicle and more than one vehicle and inserts the request into the best route(s) (Ghilas et al., 2016; Wolfinger, 2021).

Transshipment Insertion operator also inserts requests greedily, but it only tries solutions using more than one vehicle and transshipment (Masson et al., 2013; Wolfinger, 2021).

Random Insertion operator chooses vehicles and positions randomly and inserts the request once the solution is feasible (Qu and Bard, 2012; Danloup et al., 2018).

Regret Insertion operator inserts a request into the route based on regret values. This operator first tries all possibilities of inserting request r into all routes, then determines the regret value for every alternative:

$$c_r = \triangle F_r^{k_{th}} - \triangle F_r^{\text{lowest}} \tag{48}$$

where c_r is the regret value; $\triangle F_r$ is the insertion cost of r; $\triangle F_r^{k_{th}}$ is the k_{th} lowest insertion cost and $\triangle F_r^{\text{lowest}}$ is the lowest insertion cost. If now the alternative with $\triangle F_r^{\text{lowest}}$ not be chosen, then it may use a higher cost to insert r in future iterations, therefore c_r can represent a kind of look-ahead information.

In the literature, usually the r with the highest c_r is inserted in one construction step (Ropke and Pisinger, 2006; Qu and Bard, 2012), therefore n (the number of requests in the request pool) steps are needed to insert all requests. In each step, this operator tries all possible routes and positions for all requests in the requests pool. However, it will cause unnecessary computation when trying to insert requests into the other unchanged routes in the next step (experiments are provided in Section 6.6). To avoid such repetitive computation, multiple requests will be tried to be inserted into routes in one construction step. When only inserting one request in one step, at the next step the requests can choose new alternatives based on changed route(s) in this step. When inserting multiple requests, we may lose such alternatives. But these alternatives may be found using other operators and many vehicles have fixed schedules that will not be changed in ST. Therefore, we choose to insert multiple requests in each step to save the computation time.

Because these requests may use the same vehicle in the alternatives, they cannot be inserted into routes one by one depending on the sort of regret values. The possible inserted requests are divided into two groups, (a) requests which have no conflict with other requests, and (b) requests which have conflicts with other requests. Two requests have conflicts when both requests use the same vehicle(s). Notice that if the vehicle is fixed, there is no conflict because the route and schedule of the fixed vehicle will not be changed. The requests in the group (a) are inserted into routes directly. For requests in the group (b), the requests which tried to be inserted into the same vehicle k will be sorted depending on regret values. Let r_k^{regret} and r_k^{second} (if exist) represent the request with the highest and second-highest regret value among all requests which are tending to be inserted into route k. If r_k^{regret} only uses vehicle k, then it can be inserted. If r_k^{regret} uses multiple vehicles, for example k and l, the operator will check whether there are other requests that intend to use l. If only r_k^{regret} intends to use l, r_k^{regret} can be inserted. Otherwise, if the regret values of r_k^{regret} bigger than or equal to regret values of r_l^{regret}, r_k^{regret} will be inserted; if smaller, r_k^{second} (if exist) will be inserted when r_k^{second} only use k. Other requests will not be inserted in this step.

Most Constrained First Insertion operator sorts the requests depending on the following weighted function of the distance between pickup and delivery terminal when using trucks ($d^{\text{truck}}_{p(r)d(r)}$), load, and time windows:

$$C_r = \varpi_1 d^{\text{truck}}_{p(r)d(r)} + \varpi_2/(|b_{p(r)} - a_{p(r)}| + |b_{d(r)} - a_{d(r)}|) + \varpi_3 q_r \tag{49}$$

where ϖ_1, ϖ_2, and ϖ_3 are the corresponding weights (Danloup et al., 2018; Qu and Bard, 2012). Note that each component needs to be normalized by dividing the largest value of all requests. The larger the value of C_r, the harder request r fits into a route. Therefore, this operator considers the r with a larger C_r first.

5.2.2. Removal operators

All removal operators have a basic operation, i.e., removing one request. It means removing the pickup, transshipment, and delivery of this request from routes, and then recalculating the times of relevant routes. The main difference between these removal operators is that the chosen requests are different.

Worst Removal operator removes the requests with the highest cost in each route (Ghilas et al., 2016; Wolfinger, 2021).

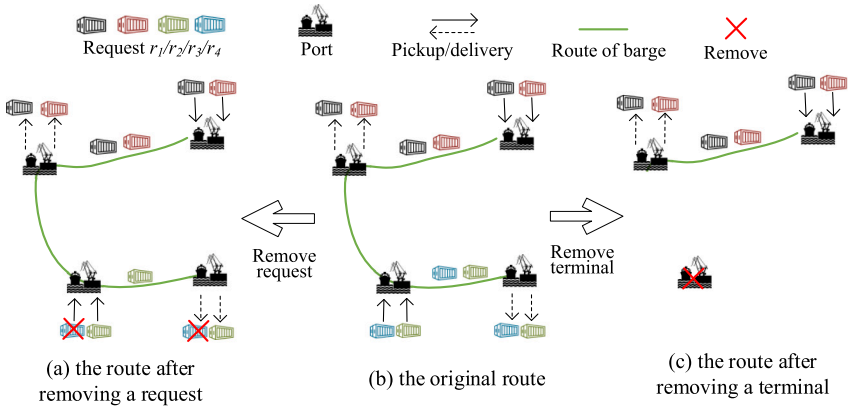

Fig. 7. Difference between removing a request and a terminal in ALNS.

Random Removal operator selects part of vehicles and removes one request from each vehicle randomly (Qu and Bard, 2012; Danloup et al., 2018).

Related Removal operator removes a request r randomly and then removes part of similar requests r' according to distance, time, load, and vehicles which can serve r and r' (Ropke and Pisinger, 2006; Danloup et al., 2018).

History Removal operator uses historical information to remove requests which may be in the wrong position and guides insertion operators to insert requests which may be inserted at a lower cost.

All insertion operators record all insertion costs, and reserve the lowest insertion cost c_r^{lowest} for each r. History Removal operator calculates the gap Δc_r between current insertion cost c_r^{current} and c_r^{lowest} and sorts the requests in descending order according to Δc_r. If there is no request whose $\Delta c_r > 0$, then the algorithm goes to next iteration directly. If there are n requests whose $\Delta c_r > 0$, this operator removes $max\{\sigma n, 1\}$ (σ is the removal proportion) requests from these n requests. Requests which are not inserted at the lowest-cost position may also make it possible for other requests to be inserted cheap, and thus allow an overall cheaper solution. Therefore a removal proportion of σ is used in this operator.

Route Removal Insertion operators may not be able to find feasible solutions based on a small number of removals in a short time. In this case, the route needs to be cleared, which means all requests in a route are removed to the request pool. Another idea behind this operator is to guide the search in the direction of minimizing the number of used vehicles and making full use of capacity.

First, this operator obtains a random number n with a given numerical distribution $[x_1, x_2, \dots, x_3]$ for $[1, 2, \dots, m]$, where m is the number of routes which served requests, $x_1 = 1/\xi$ and $x_i = x_{i-1}/\xi$ when $i > 1$. Then, it chooses n vehicles according to a probability $\psi = u_k^{\text{ava}}/(\sum_{k \in K_{\text{serve}}} u_k^{\text{ava}})$, where u_k^{ava} is available capacity and K_{serve} is a set of vehicles which have served requests. The vehicle whose capacity has not been fully made use of will have a higher probability to be cleared. In an extreme case, all routes will be cleared and all requests fill the request pool. In this case, this operator may change the search direction from the beginning and thus provide a larger neighborhood for insertion operators.

Node Removal In most times, barges and trains in ST carry multiple requests, therefore removing part of the requests may not change the routes of vehicles, as shown in Fig. 7(a). However, the cost-savings are usually obtained from minimizing distance, i.e., changing the routes of vehicles. To obtain better solutions quicker, the Node Removal operator is designed, which deletes visited terminals in the routes, as shown in Fig. 7(c). Similar to the Route Removal operator, this operator chooses n vehicles based on a distribution and probability ψ. One terminal of each route is randomly chosen and all requests which visit this terminal will be removed.

5.2.3. Swap operator

In the following cases, the requests will be wrongly placed in routes and it is difficult for the operators in Sections 5.2.1 and 5.2.2 to find the optimal solution:

1. When vehicle k is out of capacity but the served requests of vehicle k are not the most appropriate. For example, requests 1 and 2 should be served by vehicle k in the optimal solution, but vehicle k is occupied by requests 3 and 4 in the current solution. However, operators in Sections 5.2.1 and 5.2.2 cannot remove requests 3 and 4 and insert 1 and 2 precisely.
2. When vehicle k has available capacity but requests cannot be served by vehicle k due to other constraints. For example, vehicle k goes to pickup terminal A of request 1 at time 20 and arrives at the delivery terminal B at time 30. If request 2's due time is 20 at delivery terminal B, then it cannot be served by vehicle k because vehicle k needs to go to terminal A first. When there are more requests (requests 3–5) with a similar situation, the best solution should use vehicle k to serve requests 2–5 rather than only request 1 to make full use of its capacity.

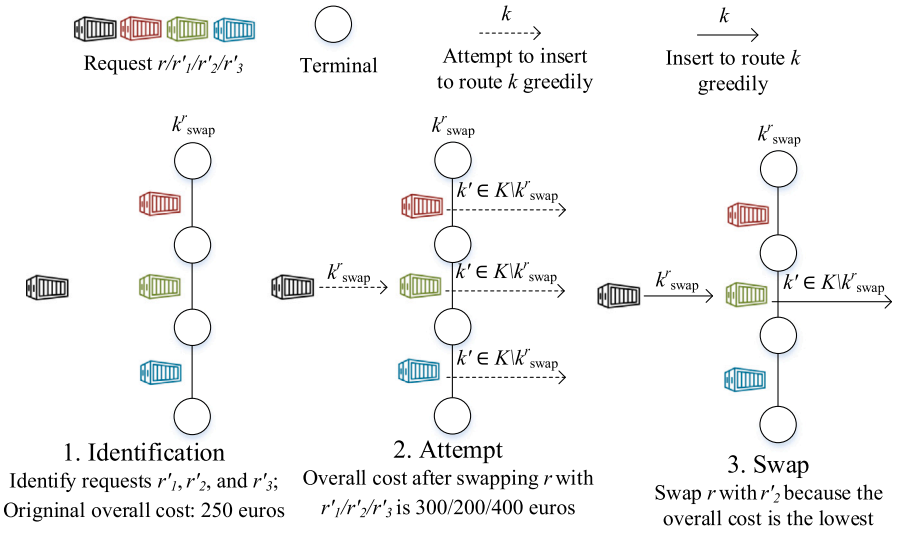

Fig. 8. Three steps in the Swap operator.

The reason behind it is that Greedy Insertion and Worst Removal operators only care about whether the solution is the best one for the individual request, rather than overall requests. Regret Insertion considers that inserting which request will let us most regret, but it still cannot find the best solution precisely. The Historical Removal operator removes the requests that are not in the historical best position, which may remove requests 1 and 2 in case 1. But it will not remove requests 3 and 4 because they are in their best position. Therefore, requests 1 and 2 still cannot be inserted into vehicle k because k's capacity is full. It is difficult to solve this problem using Random Insertion/Removal operators because they change routes randomly.

To make up for the shortcomings of existing operators, a Swap operator is designed. The Swap operator is a combination of History Removal and Greedy Insertion operators. It uses the History Removal operator to find the requests R_{swap} that are not served by the historical best vehicles K_{swap}, but only records them rather than removing them directly. As shown in Fig. 8, for all requests $r \in R_{swap}$, the following steps will be iterated:

1. Identification: The Swap operator identifies requests R'^r_{swap} that may be swapped with r. Let $K^r_{swap} \subseteq K_{swap}$ represents historical best vehicles which serve request r. For each $k^r_{swap} \in K^r_{swap}$, if it is case 1, all requests served by vehicle k^r_{swap} belong to R'^r_{swap}; if it is case 2 and vehicle k^r_{swap} is not a fixed vehicle or truck, all requests served by vehicle k^r_{swap} also belong to R'^r_{swap}. No request belongs to R'^r_{swap} when vehicle k^r_{swap} is a truck or fixed vehicle because requests served by truck fleet or using fixed schedule will not influence each other.
2. Attempt: All $r' \in R'^r_{swap}$ will be tried to be swapped with r one by one. In every attempt, firstly, the Swap operator removes request r and one possible request $r' \in R'^r_{swap}$; then, Greedy Insertion operator is used to insert request r/r' into vehicle $k^r_{swap}/k' \in K \setminus k^r_{swap}$; finally, the overall cost after the swap attempt is recorded and routes are restored as before.
3. Swap: If the lowest overall cost of all possible swaps is lower than the overall cost without swap, request r' with the lowest overall cost will be swapped with request r.

5.3. Feasibility check and synchronization

The insertion may cause infeasible solutions, e.g., the capacity constraints may be violated after an insertion. Both insertion and removal will change the times of the operated route and also relevant routes. After the insertion of each request or segment, (a) the times of the operated route will be updated, (b) vehicles that influence each other will be synchronized, and (c) the feasibility of the current solution will be checked. After the removal of each request, the (a) and (b) will be executed but (c) is not required because removal will not cause infeasible solutions. In this section, feasibility check and synchronization in ALNS are highlighted, and how to achieve flexible routing and schedule are illustrated in detail.

Same with the mathematical model, the following constraints will be checked in ALNS:

1. Subtour elimination constraints (10)–(12);
2. Capacity constraints (15);
3. Suitable routes constraints (24);
4. Time constraints (27)–(47).

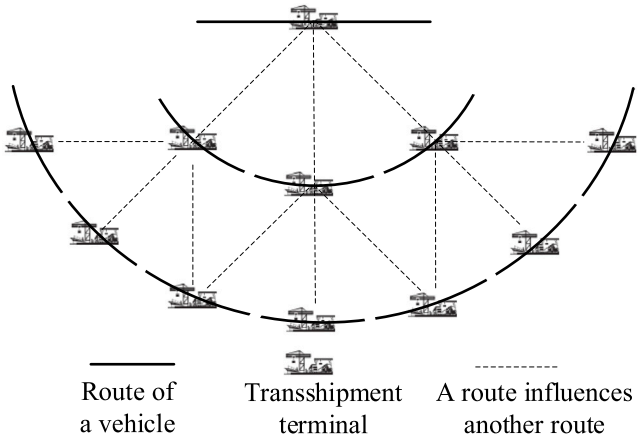

Fig. 9. Chain reaction.

Other constraints are satisfied automatically in the construction of routes, such as flow conservation (18)–(23), or preprocessing procedure, such as fixed routes constraints (25). The subtour elimination constraints can be guaranteed by checking whether there are duplicate terminals on the route. When picking up/delivering requests, the current load will increase/decrease by the quantity q_r. If the current load exceeds the capacity of the vehicle, the capacity constraints will be violated. The suitable routes constraints are ensured by checking whether the adjacent terminals in the routes are the same as unsuitable routes.

The difficulty lies in satisfying the time constraints. The times in the proposed model include time windows of requests, open time windows of terminals for fixed vehicles, loading/unloading time, waiting time, storage time, delay time, fixed travel time of barges and trains, and time-dependent travel time of trucks. Detailed feasibility checking on time constraints is shown in flow charts in Appendix B. In this section, some key points are listed, including waiting time and infeasible cases for barges and trains, time-dependent travel time for trucks, and time synchronization.

The vehicle will wait when it departs earlier than the fixed departure time ($\bar{t}_j^k < b_j^k$), arrives before pickup time window ($t'^k_j < a_{p(r)}$), and arrives earlier than the containers at transshipment terminal ($t'^k_j < Td^r_j$). If the vehicle's departure time is later than open time window ($\bar{t}_j^k > b_j^k$) or pickup time window ($\bar{t}_j^k > b_{p(r)}$), the route of vehicle k is infeasible.

At peak period, the travel time of trucks τ'^{kr}_{ij} will be longer than normal due to congestions, i.e., τ'^{kr}_{ij} is time-dependent. When the truck deliveries request at the transshipment terminal or delivery terminal, the time-dependent travel time τ'^{kr}_{ij} will be calculated depending on the departure time at the last terminal $\bar{t}_i^{kr}$ by function f_{truck}:

$$\tau'^{kr}_{ij} = \theta_m \bar{t}_i^{kr} + \eta_m \tag{50}$$

where θ_m and η_m are the slope and intersection of f_{truck} and can be calculated based on specific time period m within a day, $\bar{t}_i^{kr}$, and travel time at non-peak period τ^k_{ij} (Guo et al., 2020).

In ST with flexible services, vehicles are highly dependent on each other and synchronization is needed. The synchronization means that when a vehicle influences other vehicles, these vehicles' schedules will be re-planned and vehicles could cooperate to obtain the best solution. Such cooperation could be changing pickup/delivery time or extending/shortening the waiting or storage time. For example, at the transshipment terminal, if the pickup vehicle k arrives later than the planned time, the route plan of the delivery vehicle l needs to be synchronized to find a suitable arrival time. The route of vehicle l is called the relevant route of vehicle k. As shown in Fig. 9, a chain reaction may be caused by a small change of one route, and all relevant routes need to be synchronized. Algorithm 2 shows the synchronization on relevant routes, in which the initial input is the original changed route. To check all relevant routes shown in Fig. 9, this function is a recursion function. Only when all relevant routes meet the time constraints, the current solution is feasible, otherwise, the synchronization will stop and return "infeasible".

5.4. Comparison with (A)LNS for PDPT in the literature

A detailed comparison between the developed ALNS and the existing (A)LNS for PDPT in the literature is presented in Table 3. When an operator is used in only one paper, it is called the special operator of this paper. Although some operators proposed in the literature have similar functions to the operators designed in this paper, there are still some differences. For example, the Route Removal operator used by Danloup et al. (2018) chooses a route randomly or depending on the number of visited nodes, while in this paper, it chooses a route depending on available capacity. Besides, in this paper, we developed a Swap operator to enhance the local search, and the experiments (Section 6.6) show that the solution quality is improved by using this new operator. Several

Algorithm 2: Synchronization

Input: *relevant_routes*; **Output:** *feasibility*;
for *route* ∈ *relevant_routes* **do**
  update pickup/delivery time and extend/shorten the waiting or storage time of influenced requests;
  if *route does not satisfy time constraints* **then**
    return *infeasible*
  else
    obtain *relevant_routes* of *route*;
    Synchronization(*relevant_routes*)
  end
end
return *feasible*;

Table 3
Comparison between the proposed ALNS and existing (A)LNS for PDPT in the literature.

Article		Qu and Bard (2012)	Ghilas et al. (2016)	Danloup et al. (2018)	Wolfinger (2021)	Our paper
Removal operator	Worst	–	✓	✓	✓	✓
	Random	✓	✓	✓	✓	✓
	Route	✓	✓	✓	–	✓
	History	–	✓	–	–	✓
	Related	✓	✓	✓	✓	✓
	Special	–	Late-arrival, Worst-distance	Cluster, Many-split	–	Node
Insertion operator	Greedy	✓	✓	–	✓	✓
	Random	✓	–	✓	✓	✓
	Regret	✓	–	–	✓	✓
	Most constrained	✓	–	✓	✓	✓
	Transshipment	–	✓	–	✓	✓
	Special	–	Second best, Best λ feasible	–	–	–
Swap operator		–	–	–	–	✓
Choosing operator		Adaptive	Adaptive	Random	Random	Adaptive
Acceptance criterion		Best solution only	Simulated annealing	Fixed percentage of degradation allowed	Fixed percentage of degradation allowed	Simulated annealing
Performance improvement		Hash table	–	–	–	Preprocessing, hash table, bundle insertion
Transshipment location		Dedicated location	Dedicated location	Dedicated location	Dedicated location	Transshipment terminals
Instance size	N	–	108	100	55	10
	T	1	5	5	5	10
	R	25	100	50	100	1600
	K	3	24	Unlimited	6	116
Max. # of transshipments		Once	Allow twice	Once	Allow twice	Allow twice

–: not considered or stated in the related paper; N/T/R/K: maximum number of nodes/transshipment nodes/requests/vehicles.

performance improvement methods are also used in this paper, including preprocessing heuristics, hash table, and bundle insertion, and they are illustrated in Appendix C.

In freight transport, such as urban freight transport, all customer nodes are usually distinct, while many requests in ST share the same terminal. Therefore the maximum number of nodes in this study is fewer than others. Moreover, there are usually several dedicated transfer nodes in freight transport, which cannot be customer nodes. In comparison, almost all terminals are transshipment terminals in ST, which also have functions as pickup/delivery terminals. It increases the possibility of transshipments and the connections between vehicles. The number of requests in ST is also bigger than the number in freight transport. In Table 3, the maximum number of requests is 100 in the literature, while there are up to 1600 requests in the case studies of this paper. A large proportion of transshipment terminals together with a large number of requests make the routes of vehicles in ST highly dependent on each other. In an extreme case, a small change in one route may cause changes in several dozens of vehicles because this change will influence subsequent terminals in this route and each terminal may cause a chain reaction as in Fig. 9. Therefore, ST brings more complexity to freight transport when there are many requests. To address this issue, we propose the synchronization algorithm (Algorithm 2), swap operator, and performance improvement methods.

Table 4
The comparison between exact approach and ALNS.

T	R	L	Avg. Cost (EUR)		Avg. CPU (s)	
			Exact approach	ALNS	Exact approach	ALNS
2	1	L_0	3585	3585	343.26	0.18
2	1	L_1	3585	3585	461.56	0.18
2	1	L_2	1315	994	43 200.00[a]	0.12
2	3	L_0	8935	8935	3790.64	0.36
2	3	L_1	8935	8935	4383.69	2.68
2	3	L_2	8851	6074	43 200.00[a]	2.26
2	5	L_0	16 491	16 491	12 921.26	0.37
2	5	L_1	16 491	16 491	15 941.14	3.15
2	5	L_2	–	12 986	43 200.00[a]	2.62
5	1	L_0	2098	2098	385.46	0.13
5	1	L_1	2098	2098	431.34	0.14
5	1	L_2	3775	994	43 200.00[a]	0.06
5	3	L_0	5828	5828	5793.62	0.35
5	3	L_1	6859	5828	43 200.00[a]	1.24
5	3	L_2	11 226	5828	43 200.00[a]	1.23
5	5	L_0	13 383	13 383	11 050.79	0.59
5	5	L_1	13 738	13 383	43 200.00[a]	1.70
5	5	L_2	–	11 345	43 200.00[a]	2.37

T: number of transshipment terminals; R: number of requests; L: flexibility level.
– no solution is found due to time limitation.
[a]Time limit reached (12 h).

6. Numerical experiments

The proposed ALNS as shown in Algorithm 1 is compared with the developed MILP model and two benchmark methods from the literature that do not consider flexible services, namely Demir et al. (2016) and Guo et al. (2020). The transport network information and request data can be found in these two papers. In the comparison with the MILP model, we compared the exact approach by the commercial solver (Gurobi) and ALNS in terms of quality of solutions and computation time. In the benchmarking on small instances, firstly we compared with results in Demir et al. (2016) under different weights for the individual objectives, then we designed a scenario using transport network of Guo et al. (2020) to illustrate the function of flexible vehicles under congestion. In the benchmarking on large instances, we compared with the model in Guo et al. (2020) with up to 10 terminals, 116 services (vehicles), and 1600 requests. All instances are available at a research data website.[1] All experiments are implemented in Python 3.7 and run on Linux with 62 GB of memory and an Intel Xeon E5 CPU with a 2.40 GHz core.

We consider three levels of flexibility with an increasing degree:

1. Level 0 (L_0): all vehicles are fixed except the flexibilities considered by Demir et al. (2016) and Guo et al. (2020) when compared with them;
2. Level 1 (L_1): trucks have flexible routes and schedules, including flexible due time, waiting time, storage time, and departure time;
3. Level 2 (L_2): both trucks and barges have flexible routes and schedules.

At Level 0, the initial solution is obtained by the Regret Insertion operator. Except for Level 0, the initial solution is obtained in three different ways, as mentioned in Section 5.1. It is worth mentioning that the proposed model allows more specific flexibility levels, e.g., only part of barges can be flexible. This paper mainly shows the potential of flexibility and specific flexibility levels are not considered. The maximum number of segments of a request is also adjustable in the proposed model and it is set to three in the case studies.

6.1. Comparison with the exact approach

Table 4 shows the comparison between the exact approach (by Gurobi) and ALNS. All instances are based on the transport network with 116 vehicles published in Guo et al. (2020). There are ten terminals in the lower Rhine-Alpine corridor, and two or five of them are selected as transshipment terminals randomly. One, three, and five request(s) are randomly chosen from instances in Guo et al. (2020) and tested under different flexibility levels. All experiments are repeated three times to obtain the average values of costs and computation time. For all instances in Table 4, there are significant differences in both costs and computation time. ALNS gets the best solution in few seconds, while the exact approach needs 5 min for the smallest instance. When the numbers

[1] https://figshare.com/s/2bbc4c63fd9a7200594f.

Table 5
Comparison on results with flexibility (the proposed model) and without flexibility (Demir et al., 2016).

Case	Weights			Services/Vehicles					Total service costs (EUR)	Total penalty costs (EUR)	Total emissions costs (EUR)	Total costs (EUR)
	w_1	w_2	w_3	1	2	3	4	5				
1[a]	1	0	0	1,2,3	1,2,3	31,5	2,3	21	17 179	6720	782	24 681
1	1	0	0	1,2,3	1,2,3	31,5	2,3	21	17 179 (0%)	6720 (0%)	782 (0%)	24 681 (0%)
2[a]	0	1	0	22	22	22,26	23	28,30	32 284	0	1634	33 919
2	0	1	0	1,2,3	1,2,3	23	2,3	25	24 152 (25%)	0 (0%)	1287 (21%)	25 439 (25%)
3[a]	0	0	1	31,5,25	31,6,25	31,7	2,3	21	22 435	12 200	594	35 229
3	0	0	1	31,5,25	31,6,25	31,7	2,3	21	22 435 (0%)	11 900 (2%)	594 (0%)	34 929 (1%)
4[a]	0.4	0.4	0.2	1,2,3	1,2,3	31,5	2,3	28,30	19 171	3220	894	23 285
4	0.4	0.4	0.2	1,2,3	1,2,3	31,5	2,3	27	18 543 (3%)	3220 (0%)	865 (3%)	22 629 (3%)
5[a]	0.2	0.6	0.2	1,2,3	1,2,3	22,26	2,3	28,30	24 707	0	1303	26 010
5	0.2	0.6	0.2	1,2,3	1,2,3	1,22	2,3	27	22 739 (8%)	0 (0%)	1248 (4%)	23 987 (8%)
6[a]	0.6	0.3	0.1	1,2,3	1,2,3	31,5	2,3	21	17 179	6720	782	24 681
6	0.6	0.3	0.1	1,2,3	1,2,3	31,5	2,3	27	18 543 (−8%)	3220 (52%)	865 (−11%)	22 628 (8%)
7[a]	0.1	0.8	0.1	1,2,3	1,2,3	22,26	2,3	28,30	24 707	0	1303	26 010
7	0.1	0.8	0.1	1,2,3	1,2,3	1,22	2,3	27	22 739 (8%)	0 (0%)	1248 (4%)	23 987 (8%)
8[a]	1	1	1	1,2,3	1,2,3	31,5	2,3	28,30	19 171	3220	894	23 285
8	1	1	1	1,2,3	1,2,3	31,5	2,3	27	18 543 (3%)	3220 (0%)	865 (3%)	22 628 (3%)
9[a]	1	10	10	1,2,3	1,2,3	31,8,27,26	2,3	28,30	25 081	0	1098	26 179
9	1	10	10	1,2,3	1,2,3	1,22	2,3	27	22 739 (9%)	0 (0%)	1248 (−14%)	23 987 (8%)

[a]Means benchmark by Demir et al. (2016), in which all vehicles follow fixed routes and schedules except the truck's departure time is flexible.

of requests and transshipment terminals increase, the computation time of the exact approach increases dramatically and no solution is obtained in a limited time (12 h) when there are five requests at L_2. Increasing the number of transshipment terminals decreases the costs of ALNS, while costs of the exact approach may be higher because it cannot find the optimal solution in the limited time. Moreover, at L_0, both the exact approach and ALNS can find the optimal solution, although the exact approach needs an obviously longer time. At L_1, the exact approach cannot find the optimal solution within 12 h when there are five transshipment terminals and more than three requests. At L_2, the exact approach cannot find the optimal solution for all instances in Table 4, while ALNS finds solutions with significantly lower costs.

6.2. Optimization with and without flexibility under different weight combinations

The transport network studied by Demir et al. (2016) is located in the Danube region between Hungary and Germany, which consists of 10 terminals including inland waterway ports and railway terminals and 3 barge, 18 train, and 11 truck services. Demir et al. (2016) assume trucks can depart with a flexible time and they also consider storage time but storage cost is not included in the objective, i.e., $F_3 = 0$ in their model. To make a fair comparison, we do not consider storage cost in the objective function when comparing with Demir et al. (2016). Moreover, they compare results under different weights for service cost, penalty cost, and emissions cost. Therefore, the objective is as follows:

$$F = w_1(F_1 + F_2 + F_5) + w_2 F_6 + w_3 F_4 \tag{51}$$

The weights enable the reflection of individual preferences regarding different costs. The impact of preferences can also be analyzed with different weights.

ALNS finds all best solutions reported by Demir et al. (2016) under the same setting and hence not reported in this paper. The barge services in Demir et al. (2016) are three consecutive services operated by one barge, therefore the route of the barge is fixed and barges' timetables and truck services could be flexible. In Table 5, results with and without flexibility are compared under cases with different weight combinations for multiple objectives. There are 32 services (vehicles) and 5 requests, and their numbers are the same with Demir et al. (2016). All used services/vehicles of five requests and costs of objectives are listed and the cost saving is shown in brackets. Table 5 shows that ALNS with flexibility finds better solutions on all cases except case 1, where it finds the same solution. The differences are marginal in some cases because the solution found in the case without flexibility is also optimal under flexibility. Although in some cases the differences between sub-costs are 0%, the differences in the total cost are always larger than 0% except for case 1, where our approach finds the same solution with Demir et al. (2016). In case 3, routes for requests are the same but the delay penalty is lower due to the flexible schedule. Sometimes the proposed model sacrifices part of the objectives for a better overall solution, such as case 6. In all other cases, by using flexible vehicles, the proposed model provides better solutions from the perspectives of all objectives under different weight combinations. Moreover, all the best solutions can be found in few seconds by ALNS.

The results are always in line with the preferences (weights) on the objectives. For example, in case 3, the decision-maker prioritizes the minimization of emissions, and the electric trains (services 4–21) are chosen as much as possible, which causes a delay for request 3 and more waiting time for request 5. In this case, the total emissions cost is the lowest, but the service and

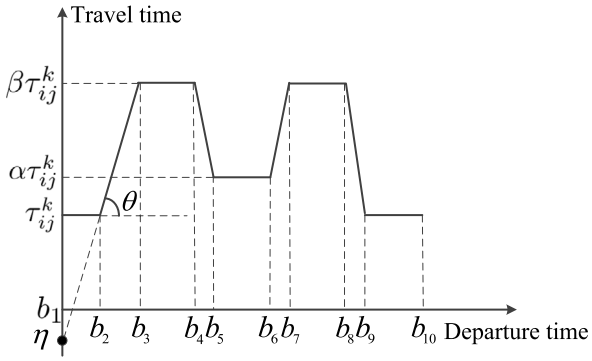

Fig. 10. Time-dependent travel times of truck services.
Source: Guo et al. (2020)

penalty costs are higher than case 8 when there are no preferences (all weights equal to 1). When the decision-maker has preferences on service cost (cases 1, 4, 6, and 8) and penalty cost (cases 2, 5, and 7), the cheaper modes (barges and trains, i.e., services 1–21) and faster modes (trucks, i.e., services 22–32) are chosen, respectively. The preferred costs usually are compensated by other higher costs, but the flexibility make the compensation as low as possible compared with the cases without flexibility.

6.3. Optimization with and without flexibility under congestion

The flexibility considered in this paper is helpful for mitigating congestions in ST. Two types of congestions are considered: arc congestion and node congestion, which are concerned with the limited capacities of roads and terminals, respectively. For arc congestion, we consider the congestions in peak periods on roads. As shown in Fig. 10, there are several time breakpoints in one day, i.e., b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, $b_{10} = 0, 5, 7, 9, 13, 13, 17, 19, 21, 24$. Between b_2 and b_9, there are congestions adjusted by coefficients α and β, which represent multiples of time spent in peak period compared with the travel time in normal period (τ_{ij}^k). According to Guo et al. (2020), when α is 2, double travel time will be needed on the road when departing at 5 pm. More congestion will cause higher costs when we take the time-dependent travel time into account by replacing Eq. (2) with:

$$F_1 = \sum_{k\in K}\sum_{(i,j)\in A}\sum_{r\in R}(c_k^1(\bar{t}_i^{kr} - t_i^{kr}) + c_k^{1'} d_{ij}^k)q_r y_{ij}^{kr} \tag{52}$$

For node congestion, terminal a will be unavailable for vehicles when the served vehicles/containers exceed its capacity, i.e., the following constraints are added:

$$\sum_{j\in N} x_{aj}^k = 0 \quad \forall k \in K \tag{53}$$

$$\sum_{i\in N} x_{ia}^k = 0 \quad \forall k \in K \tag{54}$$

Because the time horizon of the synchromodal transport planning is usually longer than one day, we use hours beginning from 0 to represent the time. For example, time 25 means 1 am on the second day. A simple but illustrative scenario using the data in Guo et al. (2020) is designed to show the function of flexibility under congestions. There are three terminals, two services, and one request:

- Terminals A, B, and C, and all terminals are connected with roads and waterways. The road/waterway distances between A and B, A and C, and B and C are 15 km/15 km, 270 km/262.5 km, 262.5 km/255 km, respectively.
- A truck fleet service with begin and end depots of terminals B and C respectively and a speed of 75 km/h.
- A barge service, whose begin depot is terminal A (fixed departure time is 66), end depot is terminal C (fixed arrival time is 83.5), speed is 15 km/h, and capacity is 160 TEU.
- A request, whose pickup terminal is B (pickup time is 63), delivery terminal is C (due time is 85), and load is 12 TEU.

There is no fixed service on other roads/waterways because the demand is low. As shown in Fig. 11(a), at L_0, the only solution is using the truck to serve this request because both truck and barge routes are fixed. However, because the pickup is at 3 pm (normalized by time 63), the truck will depart at peak time and there will be congestion on road. At L_1, the best solution uses the combination of the truck and barge (Fig. 11(b)), which means the request is picked up by the truck, transferred from truck to barge at terminal A, and delivered by barge. This solution can mitigate the impact of congestion by using inland waterways. Table 6 shows the comparison between costs of L_0 and L_1 at different congestion levels. When the congestion level increases, the cost increase of L_0 is much greater than the increase of L_1. When $\alpha = 14$, the road is disrupted due to severe congestion and there is no feasible solution at L_0.

Table 6
Costs under congestions.

Congestion level α	Cost of L_0 (EUR)	Cost of L_1 (EUR)
2	3240	906
6	5842	1050
10	8444	1194
14	–	1338

– means there is no feasible solution.

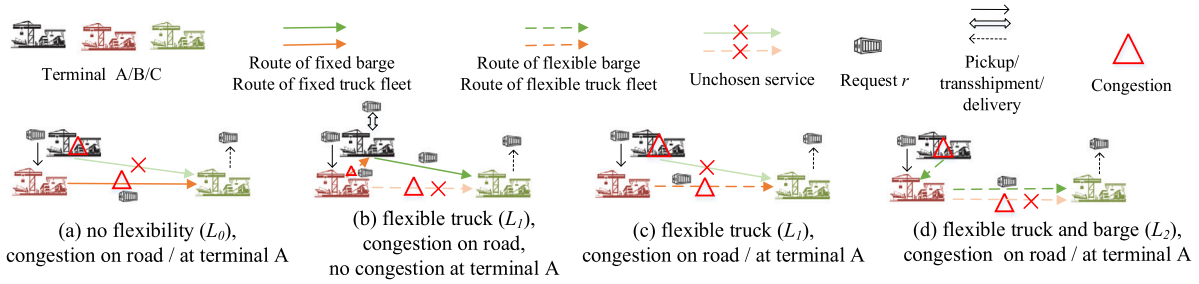

Fig. 11. Optimal routes under congestions on road and at the terminal.

Furthermore, if the truck and barge cannot use terminal A to transfer containers because too many containers are piled up at the terminal or no crane is available at the terminal (node congestion), we can only use the truck to serve the request at L_0 and L_1, as shown in Fig. 11(a) and (c). However, at L_2, we can use the barge to serve this request even though terminal A is unavailable. As shown in Fig. 11(d), The barge will go to terminal B to pick up containers firstly and then deliver the request to terminal C directly with a cost of only 628€.

6.4. Dynamic re-planning with and without flexible services

The attributes of requests may change during the operations, such as changes of pickup and delivery terminals, time windows, and loads, and it may make the original solution infeasible. To solve this problem, the proposed model can be extended to a dynamic setting for re-planning the routes and schedules. The original plan will be adjusted when the information of requests changes, as shown in Algorithm 3.

Algorithm 3: Dynamic re-planning

Input: K, R, N, A; **Output**: X_{best};
set X_{current} as empty routes of K;
$X_{\text{best}} = ALNS(K, R, N, A, X_{\text{current}})$; `// obtain the original solution.`
for *time in time_horizon* **do**
  $X_{\text{current}} \leftarrow X_{\text{best}}$;
  if *new information of requests is revealed* **then**
    define the set of changed requests as R_{change};
    remove unfinished parts of changed $r \in R_{\text{change}}$ from X_{current}, and set this new current solution as X'_{current};
    $X_{\text{best}} = ALNS(K, R_{\text{change}}, N, A, X'_{\text{current}})$
  end
end
return X_{best};

To avoid a huge impact on other requests and reach the real-time requirement of synchromodal transport, only the changed requests will be re-planned. The ALNS algorithm plans all requests only at the beginning of the planning horizon to obtain an original plan. The initial solution of the original plan could be obtained by Method R, S, or M. When new information of requests is revealed, the current solution is updated with ALNS. In this case, the initial solution of ALNS is the current solution X'_{current} in Algorithm 3. If changes are announced before the pickup, the original route will be removed from current solution and the optimization is based on new revealed information. If changes are announced after the pickup, the following constraints will be added to guarantee that request r is still transported by vehicle k:

$$\sum_{j \in N} y^{kr}_{p(r)j} = 1 \tag{55}$$

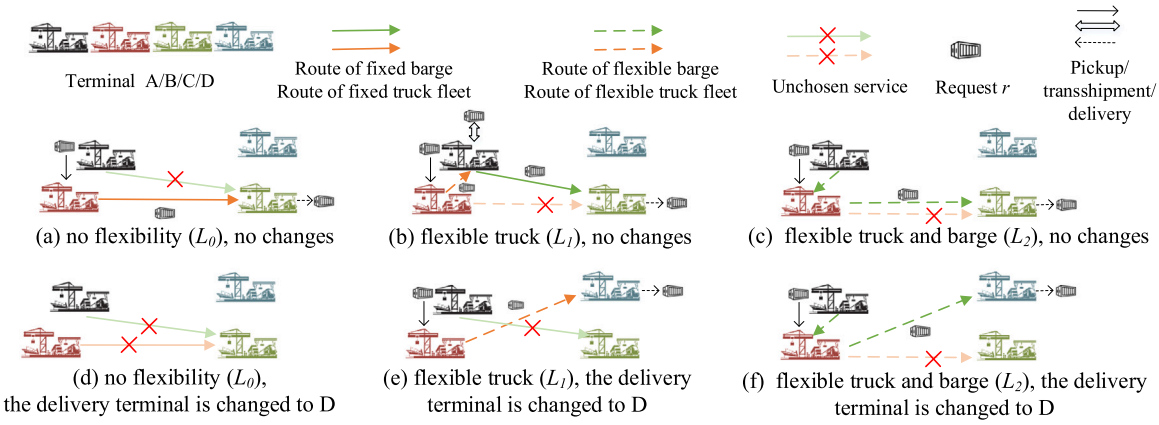

Fig. 12. Dynamic re-planning with and without flexible services.

If request r has been transferred from vehicle k to l, transshipment terminal i needs to be unchanged:

$$s_{ir}^{kl} = 1 \tag{56}$$

$$\sum_{j \in N} y_{ij}^{lr} = 1 \tag{57}$$

In ALNS, only the untraveled part will be removed from current routes, and the insertion operators will optimize the routes and schedules following the original itinerary. When the vehicle is flexible, the routes, departure time, waiting time, and storage time could be modified to cooperate with the changes of requests.

We still use the instance in Section 6.3, and a new terminal D is added, as shown in Fig. 12. The distance between D and A/B/C is 240/247.5/37.5 km (the same for road and waterway). The request's original pickup/delivery terminal is B/C, and the delivery terminal is changed from C to D at time 60. Fig. 12(a), (b), and (c) show the original plan at different flexibility levels, and they are the same with the results in Section 6.3. At time 60, the request's delivery terminal is changed and the re-planning procedure is activated. However, there is no solution under L_0 and the request can only be served by an external vehicle with a high price or rejected, as shown in Fig. 12(d). At L_1 (Fig. 12(e)), only the truck service is flexible and can be used to transport the request with 2669€. When the barge is flexible (Fig. 12(f)), the barge is used and the cost is only 678€. The best solution of the above instance can be found in less than 3 s. Other unexpected events, such as delays, new requests, and changes of time windows and loads, can be handled in a similar way by Algorithm 3. For example, when there are new requests, R_{change} in Algorithm 3 will contain information of these new requests and ALNS will insert them to the current solution X'_{current}. Therefore, the proposed model can react to the changes of the request dynamically and the service flexibility is helpful to handle unexpected events.

6.5. Benchmarking on large instances

Guo et al. (2020)'s instances are based on a transport network operated by European Gateway Services[2], which offers a wide variety of synchromodal transport services between the ports of Rotterdam and Antwerp and the leading economic centers of Western and Central Europe. The instances contain 116 vehicles (49 barges, 33 trains, and 34 trucks), 10 terminals (3 deep-sea terminals in Port of Rotterdam and 7 inland terminals in the Netherlands, Belgium, and Germany), and 10 transshipment terminals. The origins and destinations of requests are independently and identically distributed among deep-sea terminals 1, 2, 3 with probabilities 0.66, 0.2, 0.14 and inland terminals 4, 5, 6, 7, 8, 9, 10 with probabilities 0.306, 0.317, 0.153, 0.076, 0.071, 0.034, 0.043, respectively. The container volumes of requests are drawn independently from a uniform distribution with range [10, 30] and the average container volume is 20 TEU. The earliest pickup time $a_{p(r)}$ of requests is drawn independently from a uniform distribution with range [1, 120]; the latest delivery time $b_{d(r)}$ of requests is generated based on its $a_{p(r)}$ and lead time LD_r, i.e., $b_{d(r)} = a_{p(r)} + LD_r$, and the lead time of requests is independently and identically distributed among 24, 48, 72 (unit: hours) with probabilities 0.15, 0.6, 0.25. Since Guo et al. (2020) do not use time windows, we set $b_{p(r)}$ and $a_{d(r)}$ equal to $b_{d(r)}$ and $a_{p(r)}$, respectively. The objective in Guo et al. (2020) is as same as the objective in this paper. Guo et al. (2020) assume that there are unlimited storage spaces at terminals and loading/unloading infrastructure is always available for multiple vehicles. In the original paper, Guo et al. (2020) provide results with 5–30 requests and 700–1600 requests. To obtain a complete comparison, we asked the authors to run their model again, and then we got the results with 50–400 requests.

Tables 7 and 9 show the results of different levels of flexibility with different methods to obtain the initial solution. In each table, the number of requests increases from 5 to 1600, and results of different levels of flexibility are obtained for each instance. At $L_0^\star$

[2] https://www.europeangatewayservices.com/en.

Table 7
Comparison of results with different levels of flexibility (obtain the initial solution by method R).

R	L	Avg. Costs (EUR)							# of	Avg. Mode share (%)			Avg. CPU (s)		Cost
		F	F_1	F_2	F_3	F_4	F_5	F_6	vehicles	Barge	Train	Truck	Initial	Best	savings (%)
5	$L_0^\star$	4386	1532	2562	269	23	0	0	5	60	0	40	–	–	–
5	L_0	4385	1531	2562	269	23	0	0	5	60	0	40	0.2	0.0(0%)	0
5	L_1	4270	967	2994	290	18	0	0	6	66	0	33	0.4	0.0(0%)	2
5	L_2	3808	482	3312	0	14	0	0	4	100	0	0	0.9	0.0(0%)	13
10	$L_0^\star$	25 988	14 560	6990	2158	186	0	0	14	50	29	21	–	–	–
10	L_0	25 988	14 559	9084	2158	186	0	0	14	50	28	21	0.6	0.0(0%)	0
10	L_1	25 985	14 559	9084	2155	186	0	0	14	50	28	21	1.6	0.3(32%)	0
10	L_2	24 838	15 847	8304	450	211	25	0	12	69	7	23	2.9	23.7(1%)	4
20	$L_0^\star$	44 198	20 870	16 218	5003	287	0	1820	23	57	18	25	–	–	–
20	L_0	44 221	20 604	15 990	5527	279	0	1820	22	57	23	19	1.1	0.0(0%)	0
20	L_1	42 786	22 342	15 210	4940	293	0	0	21	53	23	23	4.5	0.0(53%)	3
20	L_2	36 737	19 757	15 324	1270	276	108	0	20	70	16	12	8.5	105.5(0%)	16
30	$L_0^\star$	65 126	36 953	22 896	4794	483	0	0	35	50	11	39	–	–	–
30	L_0	65 126	36 953	22 896	4794	482	0	0	35	50	11	38	1.5	0.4(0%)	0
30	L_1	64 905	35 873	23 712	4845	473	0	0	33	51	11	37	7.8	18.2(68%)	0
30	L_2	51 665	26 691	24 258	15	415	285	0	29	74	2	23	11.8	419.5(15%)	20
50	$L_0^\star$	135 679	88 930	23 919	9350	1069	0	0	46	42	14	44	–	–	–
50	L_0	139 819	96 241	33 396	9050	1131	0	0	47	41	15	42	2.9	2.1(0%)	−3
50	L_1	131 605	84 662	35 357	10 568	1017	0	0	37	41	16	41	19.7	50.9(69%)	2
50	L_2	106 728	64 815	38 058	2493	885	476	0	40	75	6	18	25.0	639.1(17%)	20
100	$L_0^\star$	181 204	92 873	53 580	15 838	1297	0	0	56	56	12	32	–	–	–
100	L_0	183 818	97 853	68 880	15 748	1337	0	0	56	56	12	30	6.9	60.8(1%)	−1
100	L_1	176 074	89 282	69 852	15 688	1251	0	0	49	55	14	30	45.3	258.4(69%)	2
100	L_2	142 721	68 150	71 016	1812	1115	614	13	57	79	5	15	72.7	3507.9(46%)	20
200	$L_0^\star$	497 380	316 980	97 149	29 648	3950	0	0	70	45	12	43	–	–	–
200	L_0	502 851	325 703	142 016	30 880	4002	0	248	72	44	13	41	13.8	1824.2(11%)	−1
200	L_1	481 192	298 941	146 606	31 694	3741	0	210	69	45	15	38	130.7	2620.3(67%)	3
200	L_2	374 768	216 237	146 145	7683	3194	934	574	73	71	4	24	138.6	7351.4(70%)	24
400	$L_0^\star$	1 100 758	754 771	194 634	62 051	8848	0	0	95	40	19	41	–	–	–
400	L_0	1 115 227	777 999	265 497	61 787	9066	2	875	96	39	18	42	47.4	14 492.7(64%)	−1
400	L_1	1 117 930	773 891	270 943	61 765	9023	0	2308	89	40	18	41	427.2	9695.9(80%)	−1
400	L_2	925 230	623 500	268 031	22 206	8109	1399	1984	82	63	6	29	855.8	29 465.0(77%)	15
700	$L_0^\star$	1 060 077	723 033	197 406	57 334	8483	0	1815	104	39	18	43	–	–	–
700	L_0	1 070 117	740 118	261 762	57 451	8647	0	2138	102	39	17	43	102.6	48 788.1(76%)	−1
700	L_1	1 071 214	735 499	267 561	57 209	8613	0	2330	98	40	17	41	1293.7	27 376.0(82%)	−1
700	L_2	943 710	654 660	249 363	25 799	8242	1530	4115	90	60	7	31	2448.5	99 186.4(85%)	10
1000	$L_0^\star$	1 017 669	692 260	189 822	62 025	8146	0	850	101	41	16	43	–	–	–
1000	L_0	1 028 469	713 238	245 619	59 702	8327	0	1581	100	39	15	45	140.7	95 235.4(75%)	−1
1000	L_1	1 024 973	704 195	251 533	59 992	8257	0	996	93	40	15	44	1318.5	60 259.9(79%)	−1
1000	L_2	934 731	630 058	255 919	35 646	7427	1213	4465	99	59	7	32	4788.3	159 355.5(72%)	7
1300	$L_0^\star$	1 042 481	704 457	196 548	58 404	8336	0	1974	103	42	16	41	–	–	–
1300	L_0	1 057 118	721 333	262 740	62 029	8486	0	2528	102	41	17	41	629.1	111 229.6(63%)	−1
1300	L_1	1 052 389	707 281	270 223	63 145	8357	0	3380	94	42	17	40	3019.4	111 889.8(84%)	−1
1300	L_2	985 232	689 334	236 429	34 801	8471	1304	14 891	97	55	8	36	7610.4	135 318.0(82%)	5
1600	$L_0^\star$	1 020 075	671 262	201 825	62 765	7961	0	408	100	43	21	37	–	–	–
1600	L_0	1 031 499	687 488	269 642	65 989	8110	0	269	101	42	20	37	294.5	146 337.4(83%)	−1
1600	L_1	1 036 620	690 889	269 330	67 511	8156	0	735	95	43	19	37	6522.0	126 142.0(83%)	−1
1600	L_2	991 122	705 891	229 860	33 201	8482	1023	12 663	99	52	11	36	6763.0	140 737.1(84%)	3

R: number of requests; L: flexibility level; $L_0^\star$: benchmark; L_0: flexibility Level 0, i.e., same with benchmark; L_1: flexibility Level 1, i.e., trucks are flexible; L_2: flexibility Level 2, i.e., trucks and barges are flexible; F: total cost, F_1: transit cost, F_2: transfer cost, F_3: storage cost, F_4: carbon tax, F_5: waiting cost, F_6: delay penalty, barge: proportion of requests served by barge, train: proportion of requests served by barge, truck: proportion of requests served by barge, initial: running time of the initial solution, best: running time of the best solution, and the percentage in bracket equals the running time of the best solution divided by the total running time;
size of segment: 20 iterations; cooling rate $c = 0.99$; 200 iterations; time limit: 48 h.

(benchmark) and L_0, the delay penalty is charged when vehicles deliver containers later than the due time, while other flexibilities, such as flexible routing and flexible waiting time, are not considered. In Table 8, there is no results under L_0 because the initial solution given by matching model is optimal under L_0. All experiments are repeated 10 times and Tables 7 to 9 show average values of results. The time limitation for all experiments is 48 h. The computation time of the benchmark is not provided because we use different computers and software with Guo et al. (2020). The Cost Savings column shows the gap in percentage between the cost of ALNS's solution and cost of solutions in Guo et al. (2020).

Table 8
Comparison of results with different levels of flexibility (obtain the initial solution by method M).

R	L	Avg. Costs (EUR)							# of	Avg. Mode share (%)			Avg. CPU (s)		Cost
		F	F_1	F_2	F_3	F_4	F_5	F_6	vehicles	Barge	Train	Truck	Initial	Best	savings (%)
5	$L_0^\star$	4386	1532	2562	269	23	0	0	5	60	0	40	–	–	–
5	L_1	4266	967	2994	286	18	0	0	6	66	0	33	0.3	0.9(10%)	2
5	L_2	3792	782	2993	0	16	0	0	5	84	0	16	0.3	0.8(6%)	13
10	$L_0^\star$	25 988	14 560	6990	2158	186	0	0	14	50	29	21	–	–	–
10	L_1	25 988	14 559	9084	2158	186	0	0	14	50	28	21	0.8	0.0(0%)	0
10	L_2	22 912	13 458	8801	436	184	32	0	11	64	19	16	0.8	14.7(29%)	11
20	$L_0^\star$	44 198	20 870	16 218	5003	287	0	1820	23	57	18	25	–	–	–
20	L_1	42 751	22 450	15 301	4703	296	0	0	21	53	21	25	0.6	16.2(34%)	3
20	L_2	36 604	20 099	14 969	1256	278	0	0	22	66	16	16	0.6	35.0(38%)	17
30	$L_0^\star$	65 126	36 953	22 896	4794	483	0	0	35	50	11	39	–	–	–
30	L_1	64 938	35 862	23 724	4877	473	0	0	34	51	11	37	0.9	13.1(12%)	0
30	L_2	50 111	25 233	24 356	29	398	94	0	30	72	4	22	0.9	264.0(62%)	22
50	$L_0^\star$	135 679	88 930	23 919	9350	1069	0	0	46	42	14	44	–	–	–
50	L_1	131 171	84 358	35 309	10 486	1017	0	0	40	42	15	42	2.8	83.9(54%)	2
50	L_2	106 530	65 774	37 017	2515	888	336	0	42	71	8	20	2.8	401.3(55%)	21
100	$L_0^\star$	181 204	92 873	53 580	15 838	1297	0	0	56	56	12	32	–	–	–
100	L_1	175 830	88 962	70 025	15 594	1249	0	0	52	55	14	30	4.1	435.4(55%)	2
100	L_2	135 866	59 862	72 136	2273	1031	563	0	60	78	6	14	4.1	2164.2(76%)	24
200	$L_0^\star$	497 380	316 980	97 149	29 648	3950	0	0	70	45	12	43	–	–	–
200	L_1	480 191	298 177	148 325	29 863	3750	0	74	72	45	14	40	9.1	1373.4(62%)	3
200	L_2	375 251	215 376	147 082	8383	3175	848	387	73	70	5	24	9.1	11 530.6(76%)	24
400	$L_0^\star$	1 100 758	754 771	194 634	62 051	8848	0	0	95	40	19	41	–	–	–
400	L_1	1 094 750	744 197	280 991	60 806	8755	0	0	97	40	18	40	29.1	7950.8(68%)	0
400	L_2	921 991	616 403	272 201	23 052	7991	1226	1117	88	61	8	29	29.1	31 958.4(81%)	15
700	$L_0^\star$	1 060 077	723 033	197 406	57 334	8483	0	1815	104	39	18	43	–	–	–
700	L_1	1 054 526	714 839	273 176	56 286	8409	0	1815	104	40	17	42	38.4	17 466.6(75%)	0
700	L_2	921 924	617 128	262 654	28 483	7896	1230	4532	101	56	11	31	38.4	97 319.8(82%)	12
1000	$L_0^\star$	1 017 669	692 260	189 822	62 025	8146	0	850	101	41	16	43	–	–	–
1000	L_1	1 010 503	679 957	261 160	60 443	8045	0	898	101	41	15	42	78.9	32 804.0(75%)	0
1000	L_2	919 037	621 175	253 311	33 110	7700	1098	2640	104	52	12	35	78.9	151 032.2(86%)	9
1300	$L_0^\star$	1 042 481	704 457	196 548	58 404	8336	0	1974	103	42	16	41	–	–	–
1300	L_1	1 038 067	693 346	276 066	58 614	8241	0	1798	105	42	16	40	158.6	89 314.2(65%)	0
1300	L_2	971 182	661 358	258 441	37 425	8147	1041	4768	109	50	13	35	158.6	129 940.4(73%)	6
1600	$L_0^\star$	1 020 075	671 262	201 825	62 765	7961	0	408	100	43	21	37	–	–	–
1600	L_1	1 017 824	666 952	280 231	62 303	7929	0	407	102	43	20	36	302.4	142 784.8(82%)	0
1600	L_2	961 277	651 516	258 086	39 327	7965	905	3477	111	50	17	32	302.4	132 404.6(74%)	5

R: number of requests; L: flexibility level; $L_0^\star$: benchmark; L_0: flexibility Level 0, i.e., same with benchmark; L_1: flexibility Level 1, i.e., trucks are flexible; L_2: flexibility Level 2, i.e., trucks and barges are flexible; F: total cost, F_1: transit cost, F_2: transfer cost, F_3: storage cost, F_4: carbon tax, F_5: waiting cost, F_6: delay penalty, barge: proportion of requests served by barge, train: proportion of requests served by barge, truck: proportion of requests served by barge, initial: running time of the initial solution, best: running time of the best solution, and the percentage in bracket equals the running time of the best solution divided by the total running time;
size of segment: 20 iterations; cooling rate $c = 0.99$; 200 iterations; time limit: 48 h.

Based on the results, we obtain the following insights:

1. Initial solution:
   (a) When using the best solution of the predefined schedule (method M) as the initial solution, the costs are lower than other ways in most cases. Therefore, adjusting the plan based on a predefined schedule is the most appropriate way for transport operators because there will be no significant changes and better solutions can be found quicker than optimization from scratch.
   (b) For small instances (less than 50 requests), optimization based on an initial solution provided by method S is better than other methods because method S can find the (near) optimal solution under L_1. However, for instances with more than 50 requests, method S performs worse than method M because the initial solution obtained by method S may be worse than the optimal solution under L_0, which is the initial solution of method M.
2. Costs:
   (a) Higher degree of flexibility usually leads to more cost savings, but sometimes ALNS at L_1 cannot find better solutions than ALNS at L_0 when the solution space is larger but the better solution is harder to find, as in the cases with 400, 700, and 1600 requests in Table 7.

Table 9
Comparison of results with different levels of flexibility (obtain the initial solution by method S).

R	L	Avg. Costs (EUR)							# of	Avg. Mode Share (%)			Avg. CPU (s)		Cost
		F	F_1	F_2	F_3	F_4	F_5	F_6	vehicles	Barge	Train	Truck	Initial	Best	savings (%)
5	$L_0^\star$	4386	1532	2562	269	23	0	0	5	60	0	40	–	–	–
5	L_0	4385	1531	2562	269	23	0	0	5	60	0	40	0.2	0.0(0%)	0
5	L_1	4269	967	2994	289	18	0	0	6	66	0	33	0.2	2.1(14%)	2
5	L_2	3793	757	3020	0	16	0	0	5	85	0	15	2.3	1.9(10%)	13
10	$L_0^\star$	25 988	14 560	6990	2158	186	0	0	14	50	29	21	–	–	–
10	L_0	25 988	14 559	9084	2158	186	0	0	14	50	28	21	0.7	0.0(0%)	0
10	L_1	25 988	14 559	9084	2158	186	0	0	14	50	28	21	0.7	0.0(0%)	0
10	L_2	21 907	12 389	8854	456	175	32	0	11	67	16	16	0.7	34.6(55%)	15
20	$L_0^\star$	44 198	20 870	16 218	5003	287	0	1820	23	57	18	25	–	–	–
20	L_0	44 221	20 604	15 990	5527	279	0	1820	22	57	23	19	1.6	0.0(0%)	0
20	L_1	42 776	22 344	15 210	4929	293	0	0	20	53	23	23	1.6	10.2(14%)	3
20	L_2	36 603	20 119	14 934	1270	278	0	0	23	66	16	16	11.8	41.4(38%)	17
30	$L_0^\star$	65 126	36 953	22 896	4794	483	0	0	35	50	11	39	–	–	–
30	L_0	65 126	36 953	22 896	4794	482	0	0	35	50	11	38	2.4	0.8(0%)	0
30	L_1	64 938	35 862	23 724	4877	473	0	0	34	51	11	37	3.0	16.8(17%)	0
30	L_2	50 468	25 638	24 307	11	402	109	0	29	73	4	23	19.9	304.5(71%)	21
50	$L_0^\star$	135 679	88 930	23 919	9350	1069	0	0	46	42	14	44	–	–	–
50	L_0	139 796	96 241	33 396	9027	1131	0	0	47	41	15	42	3.6	11.4(8%)	−3
50	L_1	131 556	84 339	35 596	10 607	1013	0	0	39	41	16	42	15.1	149.8(57%)	2
50	L_2	104 722	64 172	37 027	2306	870	346	0	42	69	9	21	164.8	622.4(68%)	22
100	$L_0^\star$	181 204	92 873	53 580	15 838	1297	0	0	56	56	12	32	–	–	–
100	L_0	183 818	97 853	68 880	15 748	1337	0	0	56	56	12	30	7.6	63.0(11%)	−1
100	L_1	176 277	89 606	69 740	15 677	1253	0	0	51	55	14	30	65.1	273.3(49%)	2
100	L_2	136 570	60 699	72 043	2251	1041	534	0	60	78	6	14	338.4	2571.0(67%)	24
200	$L_0^\star$	497 380	316 980	97 149	29 648	3950	0	0	70	45	12	43	–	–	–
200	L_0	503 213	326 231	141 763	30 962	4008	0	248	72	44	13	41	16.6	1744.5(57%)	−1
200	L_1	480 447	298 048	146 909	31 575	3735	0	180	73	45	15	39	1761.1	2212.3(70%)	3
200	L_2	375 011	216 329	145 604	8830	3184	847	217	72	69	5	24	3973.4	9941.7(76%)	24
400	$L_0^\star$	1 100 758	754 771	194 634	62 051	8848	0	0	95	40	19	41	–	–	–
400	L_0	1 116 016	778 088	265 674	62 092	9056	0	1105	96	39	18	41	55.9	18 595.5(79%)	−1
400	L_1	1 109 777	765 050	273 158	61 491	8947	2	1128	94	40	18	41	14 840.2	11 397.8(65%)	−1
400	L_2	927 872	621 526	271 452	23 902	8022	1189	1780	87	62	8	28	21 199.0	32 510.2(84%)	15
700	$L_0^\star$	1 060 077	723 033	197 406	57 334	8483	0	1815	104	39	18	43	–	–	–
700	L_0	1 069 477	738 691	261 757	58 154	8643	2	2230	102	39	17	42	117.0	40 557.4(79%)	−1
700	L_1	1 064 366	726 571	269 101	57 923	8539	2	2229	101	40	17	41	33 041.7	20 013.7(67%)	−1
700	L_2	933 777	636 168	256 529	27 271	8038	1347	4423	96	58	9	31	29 483.4	45 423.0(77%)	11
1000	$L_0^\star$	1 017 669	692 260	189 822	62 025	8146	0	850	101	41	16	43	–	–	–
1000	L_0	1 028 896	713 804	245 254	60 155	8333	0	1349	101	39	15	45	136.3	43 698.0(74%)	−1
1000	L_1	1 021 288	697 265	253 339	61 216	8192	0	1275	101	40	15	44	39 740.6	39 115.2(68%)	0
1000	L_2	931 311	643 213	243 915	32 493	7892	1202	2595	104	50	12	37	82 678.7	49 847.9(84%)	7
1300	$L_0^\star$	1 042 481	704 457	196 548	58 404	8336	0	1974	103	42	16	41	–	–	–
1300	L_0	1 057 678	722 566	262 007	62 183	8494	0	2426	102	41	17	41	232.6	30 366.8(51%)	−1
1300	L_1	1 053 422	711 026	268 448	62 944	8392	0	2611	100	42	17	40	25 422.0	35 414.2(60%)	−1
1300	L_2	1 016 902	690 033	244 096	39 575	8410	1214	33 572	102	52	10	37	25 292.9	41 472.2(68%)	2
1600	$L_0^\star$	1 020 075	671 262	201 825	62 765	7961	0	408	100	43	21	37	–	–	–
1600	L_0	1 032 359	689 402	268 823	65 788	8129	0	217	100	42	20	37	345.0	47 730.8(78%)	−1
1600	L_1	1 031 410	686 880	270 284	65 872	8112	0	260	100	42	19	37	45 326.4	32 501.3(54%)	−1
1600	L_2	1 009 904	706 790	237 215	40 799	8449	898	15 752	103	48	13	37	29 850.2	49 071.6(78%)	0

R: number of requests; L: flexibility level; $L_0^\star$: benchmark; L_0: flexibility Level 0, i.e., same with benchmark; L_1: flexibility Level 1, i.e., trucks are flexible; L_2: flexibility Level 2, i.e., trucks and barges are flexible; F: total cost, F_1: transit cost, F_2: transfer cost, F_3: storage cost, F_4: carbon tax, F_5: waiting cost, F_6: delay penalty, barge: proportion of requests served by barge, train: proportion of requests served by barge, truck: proportion of requests served by barge, initial: running time of the initial solution, best: running time of the best solution, and the percentage in bracket equals the running time of the best solution divided by the total running time;
size of segment: 20 iterations; cooling rate $c = 0.99$; 200 iterations; time limit: 48 h.

(b) The cost savings increase when the number of requests increases from 5 to 200 requests. When there are more than 200 requests, the cost savings decrease because of two reasons: (i) the solutions are tighter due to limited capacities; (ii) ALNS cannot find (near) optimal solutions in limited time due to the complexity of larger size instances.

(c) By using flexible vehicles, cost savings of up to 24% (200 requests at L_2) and 170,000€ (400 requests at L_2) are obtained compared with the cost without flexible vehicles. On average, the proposed model at L_2 reduces the cost

by 14% compared with the best solutions without flexibility. It is worth noting that the cost savings are related to parameters and may differ from one instance to the other.

(d) The cost savings mainly come from the reduction in transit cost, storage cost, and carbon tax. The transfer cost, waiting cost, and delay penalty usually increase slightly when the total cost decreases because more transshipments and more barges are used.

(e) The carbon tax decreases when there are more flexibilities because more barges are used. This insight is obtained when there is no restriction on the schedules of barges. If the schedules were very restricted we would not be able to have emissions reductions as we would be stuck with trucks in many cases.

3. Number of vehicles and mode share:

(a) At Level 1, trucks are flexible but the mode share of trucks will decrease and the mode share of trains and barges will increase in most cases. Using more trucks will not reduce cost, but flexible trucks increase the possibilities of using more trains and barges by intermodal transport. The number of used vehicles may decrease because fewer trucks are used.

(b) At Level 2, the requests will be shifted from trucks to barges. However, the number of used vehicles sometimes increases compared with Level 1, especially for instances with 1000 to 1600 requests, because more barges and transshipments are used.

(c) Using more barges may cause more waiting time and a little delay, but will reduce costs significantly.

4. Computation time:

(a) For instances with 5–30 requests, the best solution is found in 3 s, 30 s, and 8 min at L_0, L_1, L_2, respectively. For instances with 50–400 requests, the best solution is found in 5 h, 3 h, and 10 h at L_0, L_1, L_2, respectively. This time is 41 h, 40 h, and 39 h for instances with 700–1600 requests. Therefore, for small instances, a higher degree of flexibility means longer computation time. For large instances, using limited resources to serve a large number of requests in the most appropriate way is difficult when no flexibility is considered. More alternatives are provided when more flexibility is considered, therefore the best solution may be found in a shorter time compared with the lower degree of flexibility.

(b) When using the best solution of the problem with fewer flexibilities as the initial solution, the total computation time is obviously longer than the other ways because it spends significant time to obtain the initial solution. However, the running time of the best solution (time after the initial solution is found) is less than other methods for large instances, although the best solution may not be better than other methods.

(c) For small instances, the best solution is found in the early iterations of ALNS. When the instance size increases, more iterations are needed.

(d) The computation time of large instances reflects the real-time optimization ability of the proposed model. Take the instance with 50 requests as an example, the best solution can be found in less than 15 min, which means the proposed model is able to handle 50 changed requests every 15 min under the same hardware used in this paper.

At L_0 and L_1, ALNS is stable and the differences among multiple optimization runs of ALNS are usually less than 1%. At L_2, the differences of runs are bigger due to larger solution space and limited running time. Fig. 13 shows the box plots of different numbers of requests at L_2. The cost savings compared with the benchmark is calculated and different methods for the initial solution are compared. The proposed model provides better solutions on all instances at L_2 when using methods R and M to obtain the initial solution. When using method S, all solutions are better except for the instance with 1300 requests. When there are 5 to 400 requests, the proposed model with flexible vehicles reaches at least 4% and up to 24% cost savings. When there are 700 to 1600 requests, the cost savings are between 0% and 16% except for the instance with 1300 requests and the S method. From the perspective of overall performance, method M performs better than the other two methods in cost savings. For instances with 10 and 50 requests, method S is the best one. For instances with less than 100 requests, method R has very stable performance, although it performs worst. For instances with 200 requests to 1000 requests, methods S and R have similar performances. When there are more than 1300 requests, method R performs better than method S but not stable. According to the results, adjusting the original plan (method M) is more appropriate than making a plan from scratch (methods R and S) for transport operators who consider flexible services.

6.6. Evaluations on the customized operators

As illustrated in Section 5.2, a new Swap operator is proposed and some operators are customized depending on the characteristics of ST. Using instances with 5–100 requests in Section 6.5, Fig. 14 shows the comparison on the (a) results with and without Swap operator and (b) results of inserting one/multiple request(s) in one construction step in Regret Insertion operator. In Fig. 14(a), the cost and computation time of the optimal solution are compared. In Fig. 14, solutions are obtained by the Regret Insertion operator from scratch and only results of the initial solution are presented to avoid influences from other operators. Fig. 14(a) shows that the cost with Swap operator is always lower than without Swap operator, except the instance with 5 requests, where the costs are the same. Moreover, the cost gap is increasing with the number of requests and reaches 35% on the instance with 100 requests. The reason behind it is that the Swap operator inserts more requests into better positions for a larger instance. Therefore, it is shown that the Swap operator has an important role in obtaining superior solutions, although it slightly increases the computation time.

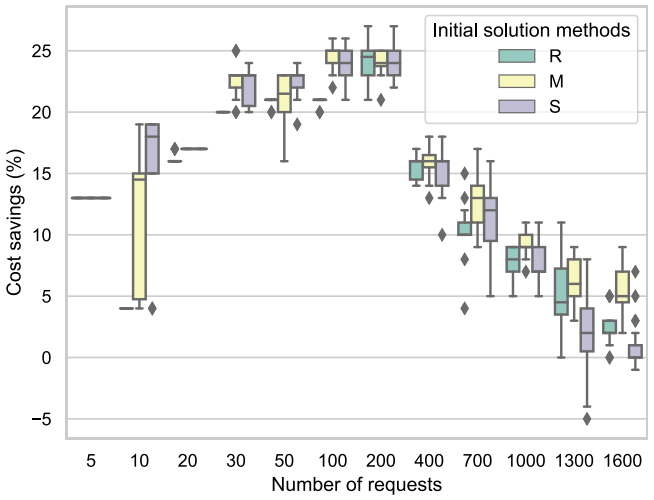

Fig. 13. Box plots of different numbers of requests at L_2. The y-axis is the cost savings compared with the benchmark.

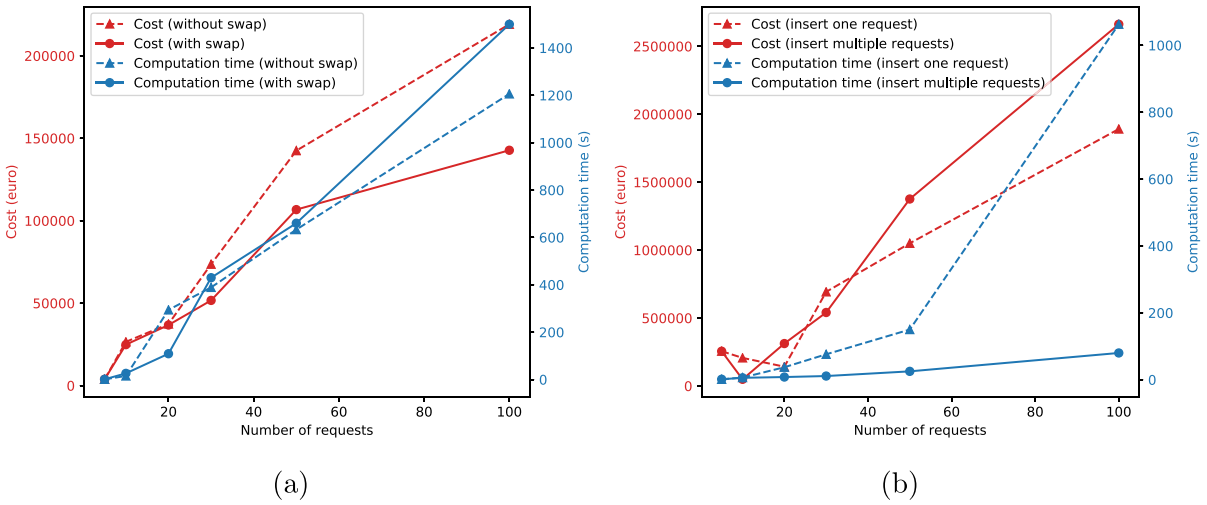

Fig. 14. Evaluations on the customized operators. (a) Comparison on results with and without Swap operator; (b) Comparison on inserting one/multiple request(s) in Regret Insertion operator.

Fig. 14(b) shows that inserting one request in Regret Insertion operator does not always obtain a lower cost than inserting multiple requests, such as instances with 10 requests and 30 requests. In contrast, the computation time is decreased dramatically by inserting multiple requests in one step, especially on large instances. For example, the computation time is reduced by 92% on the instance with 100 requests.

6.7. Summary

By comparing with the exact approach, the results verify that ALNS reduces computation time significantly. Compared with existing models in the literature (Demir et al., 2016; Guo et al., 2020), the proposed model provides considerable cost savings by using flexible vehicles. When allowing flexibility, the requests will be shifted to low-cost and low-emission modes, especially inland waterways. The cost savings are mainly due to the better utilization of capacities of low-cost vehicles, such as trains and barges. Additional cost savings for small instances are usually obtained by avoiding unnecessary trips and transshipments. For large instances, reducing storage costs by flexible vehicles is another important source of cost savings.

The proposed model performs well under different weights for the individual objectives and generates better solutions compared with solutions in the literature (Demir et al., 2016). When there are congestions, the proposed model can mitigate the impact of congestions by using flexible vehicles. In large instances, solutions with lower costs (benchmark by Guo et al. (2020)) can always be found when allowing flexibility. Moreover, the proposed model performs consistently well on different transport networks published in the literature.

In addition, the results show that the ALNS with customized operators performs better than without these operators.

7. Conclusions and future directions

In order to exploit the service flexibility in synchromodal transport planning, a novel MILP model is formulated and a customized Adaptive Large Neighborhood Search (ALNS) is proposed to solve the problem efficiently. The features of synchromodal transport, such as multiple modes, transshipment, the mix of fixed and flexible vehicles, complex schedules, and synchronization are considered in the proposed model. To achieve flexible synchromodal transport, vehicle and request routes are planned simultaneously. In order to benefit from the proposed model for realistic size instances, several customized operators are designed and performance improvement methods are proposed in ALNS. The proposed model performs well under different weights for the individual objectives and can mitigate congestions by using flexible vehicles. When attributes of requests change, the proposed model can also switch transport modes flexibly in (near) real-time according to the latest information. By comparing with models published in the literature, the results demonstrate that the proposed model can reduce cost by 14% on average when using flexible vehicles. Moreover, the proposed model performs consistently well on large instances and different transport networks. Mode share of barges increases obviously and the carbon tax is reduced when using flexible vehicles, therefore the proposed model is promising for green transportation.

The proposed model provides an optimization framework for synchromodal transport with flexible services and shows promising potential for cost savings and emissions reduction by exploiting the flexibility. The transport operators can use the proposed model to make schedules for the mix of fixed and flexible vehicles and achieve economic and sustainable transport operations. Based on the experimental results, the following managerial insights are obtained:

1. Utilizing service flexibility can reduce costs under given resources and enables the transport operator to be more competitive. More cost savings need a higher degree of flexibility, which allows containers to be shifted to low-cost modes by utilizing the capacity of barges and trains. However, blindly pursuing low cost will cause delays and longer waiting times. The proposed model can be used to make decisions taking into account the trade-off between costs, emissions, delays and waiting times.
2. Flexible services facilitate the modal shift in synchromodal transport. When trucks are flexible, the mode share of trucks will decrease because it increases the possibilities of using more trains and barges by transshipments. Moreover, flexible barges are necessary for reducing emissions because many requests are still stuck with trucks when only trucks are flexible.
3. Considering the types of cargoes and the characteristics of companies, different transport operators have different preferences about transportation. Many of them consider multiple objectives to optimize, yet, the importance of different objective terms may be different. Satisfying one sub-objective is often detrimental to other sub-objectives, while flexible services make the detriment as low as possible.
4. When there are congestions, especially severe congestions, the impacts can be alleviated more with a higher level of flexibility because more options are provided.
5. Flexible services provide more alternatives in case of changes in the request and react to unexpected events in (near) real-time more efficiently. Service flexibility also plays an important role even when demand is known beforehand and there are no dynamics. Namely, demand might be different every day, thus services need to be flexible to avoid empty miles, delay, and low load factors. Furthermore, flexible services are able to reduce cost and shift towards sustainable modes.
6. Compared with planning from scratch, adjusting the transport plan with predefined schedules is the best way for transport operators to adopt flexible services, which will not change the original plan significantly and provide more cost savings.

It is worth noting that managerial insights 2 and 4 have been provided by the existing literature (Ambra et al., 2019; Guo et al., 2020) and confirmed by this study.

We suggest the following topics for further research:

1. In reality, it may not always be easy to achieve full flexibility because flexible routing and scheduling need the collaboration of terminal operators. In the numerical experiments, we assume all trucks/barges are flexible and the situations that only some specific vehicles can be flexible are not considered, although specific flexible vehicles are possible in the proposed model. To build a more practical model, future research should look into what is the best strategy to use flexible vehicles and (a) which vehicle can be flexible? (b) how much flexibility is allowed for a flexible vehicle?
2. When transport operators have different preferences on the components of the objective function, different flexible transport operations are needed. Although this study considers preferences as weights, the conflicts between objectives are not studied. Approaches in multi-objective optimization, such as Pareto Optimality, can represent different trade-offs of conflict objectives. Therefore, a preference-based multi-objective optimization model is an interesting research direction.
3. Multiple transport operators may want to collaborate to reduce costs, time, or emissions. It is worth studying whether flexible vehicles can contribute to collaborative planning. In the same terminal, multiple transport operators may have conflicts due to limited capacity and resources, hence the terminal operator needs to allocate storage resources, schedule vehicles of the same type, and arrange transshipments. Therefore, another potential direction is to study how the terminal operator makes plans with transport operators cooperatively.
4. Uncertainties exist in operations of synchromodal transport. Transport operators might be able to handle uncertainties in a better way by allowing flexibility. For example, a spot market releases some new requests, which cannot be served or only be served at high costs by predefined schedules but can be served when considering flexible routes of vehicles. Therefore, it is interesting to research optimization under flexibility and uncertainty.

CRediT authorship contribution statement

Yimeng Zhang: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Wenjing Guo:** Software, Validation, Writing – review & editing. **Rudy R. Negenborn:** Writing – review & editing, Supervision. **Bilge Atasoy:** Conceptualization, Writing – review & editing, Supervision.

Acknowledgments

This research is supported by the China Scholarship Council (CSC) under Grant 201906950085 and the project "Complexity Methods for Predictive Synchromodality" (project 439.16.120) of the Netherlands Organisation for Scientific Research (NWO). This research is also supported by the project "Novel inland waterway transport concepts for moving freight effectively (NOVIMOVE)". This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 858508. We wish to extend our special thanks to Emrah Demir and Martin Hrušovskỳ for the data they provided and detailed explanation. We also gratefully thank the reviewers for the constructive comments.

Appendix A. Valid inequalities for mathematical model

The valid inequalities are divided into three categories and the reduced variables are indicated in brackets.

1. Valid inequalities related to requests (y_{ij}^{kr}):

   (a) Terminal i or j cannot be dummy depot.

   $$y_{ij}^{kr} = 0 \quad \forall k \in K,\ \forall r \in R,\ \forall i \in \overline{O},\ \forall j \in N \tag{58}$$

   $$y_{ij}^{kr} = 0 \quad \forall k \in K,\ \forall r \in R,\ \forall i \in N,\ \forall j \in \overline{O} \tag{59}$$

   (b) $K_{\text{small}}^{r} \subseteq K$ represents set of vehicles with a capacity that cannot accommodate request r, i.e., violate capacity constraints (15).

   $$y_{ij}^{kr} = 0 \quad \forall k \in K_{\text{small}}^{r},\ \forall r \in R,\ \forall (i,j) \in A \tag{60}$$

   (c) $K_{\text{early}}^{r} \subseteq K_{\text{fix}}$ represents set of fixed vehicles whose latest departure time b_i^k is earlier than request r's earliest pickup time $a_{p(r)}$, i.e., violate Constraints (34).

   $$y_{ij}^{kr} = 0 \quad \forall k \in K_{\text{early}}^{r},\ \forall r \in R,\ \forall (i,j) \in A \tag{61}$$

   (d) $K_{\text{late}}^{r} \subseteq K$ represents set of vehicles whose earliest departure time at pickup terminal, i.e., the time from begin depot to pickup terminal plus loading time, later than request r's latest pickup time $b_{p(r)}$. K_{late}^{r} will be removed due to Constraints (34).

   $$y_{ij}^{kr} = 0 \quad \forall k \in K_{\text{late}}^{r},\ \forall r \in R,\ \forall (i,j) \in A \tag{62}$$

2. Valid inequalities related to vehicles (x_{ij}^{k}):

   (a) A vehicle $k \in K$ cannot go to other vehicles' dummy depots.

   $$x_{ij}^{k} = 0 \quad \forall k \in K,\ \forall i \in \overline{O} \setminus \overline{o}(k), \forall j \in N \tag{63}$$

   $$x_{ij}^{k} = 0 \quad \forall k \in K,\ \forall i \in N, \forall j \in \overline{O} \setminus \overline{o'}(k) \tag{64}$$

   (b) If there is a dummy depot in x_{ij}^{k}, it must be together with a depot.

   $$x_{\overline{o}(k)j}^{k} = 0 \quad \forall k \in K,\ \forall j \in N \setminus o(k) \tag{65}$$

   $$x_{i\overline{o'}(k)}^{k} = 0 \quad \forall k \in K,\ \forall i \in N \setminus o'(k) \tag{66}$$

   (c) Begin depot cannot be j when i is not dummy begin depot; end depot cannot be i when j is not dummy end depot.

   $$x_{io(k)}^{k} = 0 \quad \forall k \in K,\ \forall i \in N \setminus \overline{o}(k) \tag{67}$$

   $$x_{o'(k)j}^{k} = 0 \quad \forall k \in K,\ \forall j \in N \setminus \overline{o'}(k) \tag{68}$$

   (d) Dummy begin depot cannot be j in x_{ij}^{k}; dummy end depot cannot be i in x_{ij}^{k}.

   $$x_{i\overline{o}(k)}^{k} = 0 \quad \forall k \in K,\ \forall i \in N \tag{69}$$

   $$x_{\overline{o'}(k)j}^{k} = 0 \quad \forall k \in K,\ \forall j \in N \tag{70}$$

(e) Remove x_{ij}^{k} when there is no compatible y_{ij}^{kr}.

$$x_{ij}^{k} \leqslant \sum_{r \in R} y_{ij}^{kr} \quad \forall k \in K,\ \forall (i,j) \in A \tag{71}$$

3. Valid inequalities related to transshipment (s_{ir}^{kl}):

(a) A transshipment only happens when request r can be transported by both vehicles k and l at transshipment terminal i.

$$s_{ir}^{kl} \leqslant \sum_{j \in N} y_{ji}^{kr} \quad \forall r \in R,\ \forall i \in T,\ \forall k,l \in K \tag{72}$$

$$s_{ir}^{kl} \leqslant \sum_{j \in N} y_{ij}^{lr} \quad \forall r \in R,\ \forall i \in T,\ \forall k,l \in K \tag{73}$$

(b) Request r's pickup/delivery terminal cannot be transshipment terminal i.

$$s_{p(r)r}^{kl} = 0 \quad \forall r \in R,\ \forall k,l \in K \tag{74}$$

$$s_{d(r)r}^{kl} = 0 \quad \forall r \in R,\ \forall k,l \in K \tag{75}$$

(c) For a fixed vehicle k, the terminals in the predefined route should contain transshipment terminal i when k is used to transfer a request. K_{noT}^{r} represents set in which vehicles cannot meet the mentioned requirements.

$$s_{ir}^{kl} = 0 \quad \forall r \in R,\ \forall k \in K_{\mathrm{noT}}^{r},\ \forall l \in K,\ \forall i \in T \tag{76}$$

$$s_{ir}^{kl} = 0 \quad \forall r \in R,\ \forall k \in K,\ \forall l \in K_{\mathrm{noT}}^{r},\ \forall i \in T \tag{77}$$

(d) When request r is transferred from vehicle k to vehicle l trough transshipment terminal i, vehicle k's begin depot and l's end depot cannot be i.

$$s_{o(k)r}^{kl} = 0 \quad \forall r \in R,\ \forall k,l \in K \tag{78}$$

$$s_{o'(l)r}^{kl} = 0 \quad \forall r \in R,\ \forall k,l \in K \tag{79}$$

(e) When request r is transferred from vehicles k to vehicle l trough transshipment terminal i and both vehicles k and l have fixed time schedules, l's departure time cannot be earlier than k's arrival time at i. K_{earlyT} represents set in which vehicle combinations violate this rule.

$$s_{ir}^{kl} = 0 \quad \forall r \in R,\ \forall (k,l) \in K_{\mathrm{earlyT}}^{i},\ \forall i \in T \tag{80}$$

Moreover, the other variables, such as z_{ij}^{k} and t_{i}^{kr}, are reduced when there is no compatible variables x_{ij}^{k}, y_{ij}^{kr}, or s_{ir}^{kl}.

Appendix B. Feasibility checking on time constraints

Before calculating times, the vehicle's start time needs to be defined. If the vehicle is a fixed vehicle and not truck, its start time at begin depot is $a_{o(k)}^{k}$. Otherwise, there are two situations: (a) if $o(k)$ is pickup terminal p_{r_1}, assign $a_{p(r_1)}$ to $t_{o(k)}^{k}$; (b) if $o(k)$ is transshipment terminal T_{r_1} of the first served request, assign delivery time at transshipment terminal $Td_{o(k)}^{r_1}$ to $t_{o(k)}^{k}$. If it not belongs to any above situations, the vehicle will start from begin depot at time 0.

Flow Chart 15 shows the flexibility check for barge (or train) k when it at terminal j. The different situations of fixed/flexible vehicles and transshipments are also distinguished. Flow Chart 16 shows how to assign time to the truck fleet. $useT$ means the request is transferred before and Tp_{i}^{r} means request r's pickup time at transshipment terminal i. There are no waiting times and infeasible situations when using trucks because trucks can serve requests immediately and delay is allowed.

Appendix C. Performance improvement

Although the ALNS is a powerful heuristic, it is still hard to solve the proposed problem efficiently on real-life instances due to the complexity brought by characteristics mentioned in Section 3. Therefore, several methods are used to improve the performance of ALNS, in which preprocessing heuristics are used to reduce the solution space before the optimization and both hash table and bundle insertion are used to speed up the search process during the optimization.

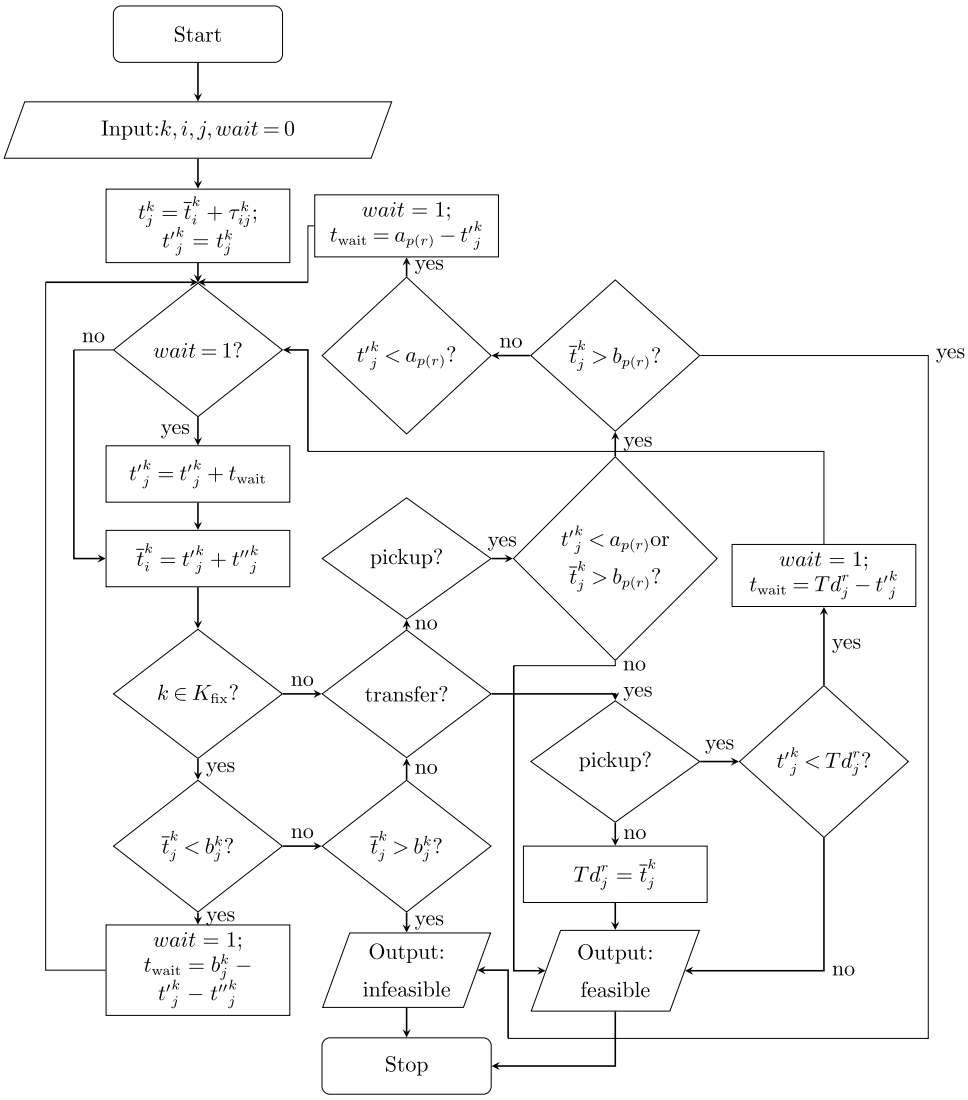

Fig. 15. Barge and train's time at pickup/delivery/transshipment terminal in ALNS.

C.1. Preprocessing heuristics

Similar to the valid inequalities in Appendix A, we designed some preprocessing heuristics to reduce the solution space of ALNS. Some new sets are used in ALNS. K_r^{nk} represents vehicle combinations that can serve the same request r. The K_r^{1k} is the set of vehicles that can serve request r by itself. K_r^p represents vehicles that can pick up request r. K_r^i ($i \in T$) represents vehicles that serve request r with specific transshipment terminal i. In ALNS, these sets are used when the related type of vehicles is needed to serve requests. The preprocessing heuristics are divided into two categories and the reduced sets in ALNS are indicated in brackets. The reference to valid inequalities in Appendix A will be given if the meaning of the preprocessing heuristic is as same as the valid inequalities.

1. Preprocessing heuristics related to requests (K_r^{1k}, K_r^{nk} and K_r^p):

   (a) k in Appendix A 1b, 1c, and 1d will be removed from related K_r^{1k}, K_r^{nk}, and K_r^p.

   (b) For $k \in K_{\text{fix}} \cap K_r^{1k} \cap K_r^p$, its route should contain arc $(p(r), d(r))$. For $k \in K_{\text{fix}} \cap K_r^{nk} \cap K_r^p$, its route should contain $p(r)/d(r)$ if it is used to pick up/deliver r. Moreover, when two fixed vehicles serving the same request r in succession, their routes should contain the same transshipment terminal. Vehicles which violate the above rules will be removed from related K_r^{1k}, K_r^{nk}, and K_r^p.

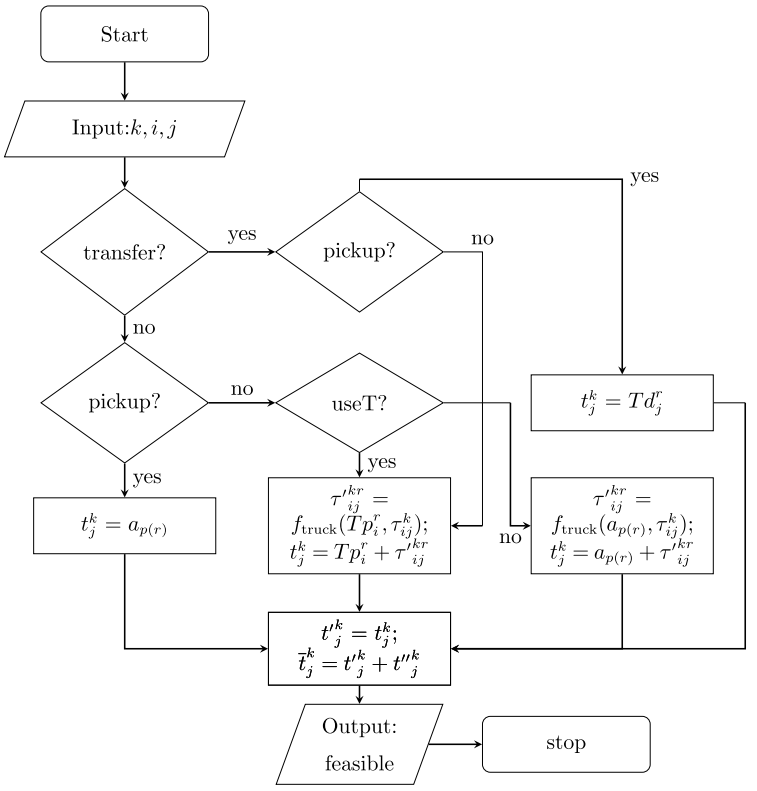

Fig. 16. Truck's time at pickup/delivery/transshipment terminal in ALNS.

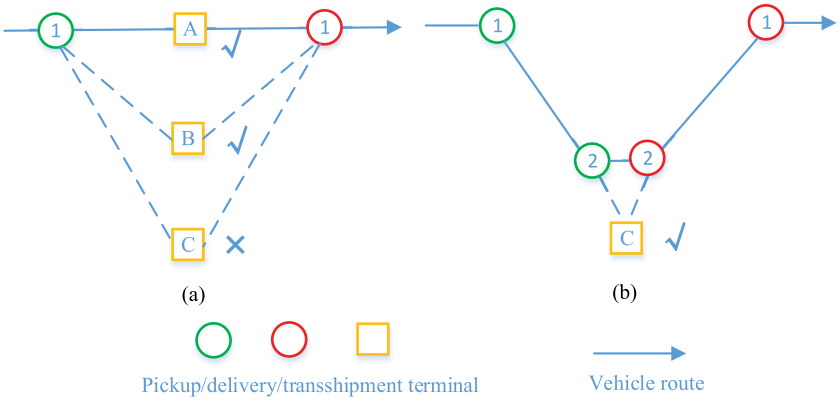

Fig. 17. Reducing transshipment terminals.

2. Preprocessing heuristics related to transshipment (K_r^i):

   (a) k in Appendix A 3b, 3c, 3d, and 3e will be removed from K_r^i.

   (b) Vehicles that use transshipment terminal i will be removed from K_r^i when using terminal i increases too much distance, i.e., $d^k_{p(r)i} + d^k_{id(r)} > \varphi d^k_{p(r)d(r)}$, where φ is a coefficient set according to the specific transportation network. As shown in Fig. 17(a), three transshipment terminals are considered for request 1, and vehicles that use transshipment terminal C will be removed from K_r^i. However, vehicles that use transshipment terminal C will be added to K_r^i during the optimization of ALNS if the vehicle goes to nearby terminals, which is illustrated in Fig. 17(b) with request 2 that is nearby transshipment terminal C.

   (c) The vehicle combinations which are not in K_r^{nk} will be removed from K_r^i.

Table 10
The keys and values in hash tables of successful insertion.

Name	Keys	Value
All_{1k}	$(r, route)$, $position_{1k}$	$value_{1k}$
$Best_{1k}$	$(r, route)$, $position_{1k}^{\text{best}}$	$value_{1k}^{\text{best}}$
All_{2k}	$(r, route_1, route_2)$, T, $position_{2k}^{\text{best}}$	$value_{2k}$
$Best_{2k}$	$(r, route_1, route_2)$, T^{best}, $position_{2k}^{\text{best}}$	$value_{2k}^{\text{best}}$

Table 11
The components in keys and values.

Name	Components
r	$(p(r), d(r), a_{p(r)}, b_{p(r)}, a_{d(r)}, b_{d(r)}, q_r)$
$route$	$(i, v_k, u_k, t_i^k, t'^k_i, \bar{t}_i^k, label_i), i \in N_k$
$position$	(m, n)
$position_{2k}^{\text{best}}$	$((m_1, n_1), (m_2, n_2))$
$value_{1k}$	$(route^{inserted}, cost_r)$
$value_{2k}$	$(route_1^{inserted}, route_2^{inserted}, cost_r, k_1, k_2)$

C.2. Hash table

When using insertion operators, it is typically necessary to evaluate the same move repeatedly during the optimization. Avoiding these repetitive computations can significantly reduce computation time, especially for large instances. Inspired by the idea proposed in Qu and Bard (2012), a cache structure that uses hash tables is implemented. Specifically, the hash table holds the best insertion positions and infeasible insertion positions for a given request and route.

Tables 10 and 11 give an example and illustrate how to establish hash tables with and without transshipment. The keys and values of hash tables of successful insertion are shown in Table 10. Table 11 shows the components in keys and values. The first two hash tables are for insertions without transshipment, which include all possible positions (All_{1k}) and the best position ($Best_{1k}$) during the search separately. Both of them have two keys and therefore have three layers. The first layer is key $(r, route)$, which includes the inserted request and route. r includes all information of the request except index to avoid unnecessary storage when there is the same request in the hash table. $route$ includes all visited terminals $i \in N_k$, speed, capacity, time, and the label $label_i$ of the visited terminal, e.g., delivery request 1. The second layer is key $position_{1k}$, which is the inserted position (m, n) of pickup and delivery. The third layer is the value, which includes the route after insertion and the cost of the inserted request. The other two hash tables are for insertion with transshipment and they have four layers due to a new key T, which is the transshipment terminal. T divides the request into two sub-requests, therefore $position_{2k}$ has two position tuples at two routes. Correspondingly, $value_{2k}$ also has two routes and names of two vehicles. $cost_r$ in $value_{2k}$ is the cost of inserted request at both routes.

Similarly, the hash tables for failed insertion include the same keys but they do not have values because the solution is infeasible.

C.3. Bundle insertion

The requests with the same pickup and delivery terminals are called bundle requests. The basic cost of request r includes request cost, loading/unloading cost and carbon tax, which are not dependent on time, as the following equation shows:

$$F_{basic} = \sum_{k\in K}\sum_{(i,j)\in A}(c_k^1\tau_{ij} + c_k^{1'}d_{ij}^k)q_r y_{ij}^{kr} + \sum_{k,l\in K,k\neq l}\sum_{i\in T}(c_k^2 + c_l^2)q_r s_{ir}^{kl} + \sum_{k\in K}\sum_{(i,j)\in A_p} c_k^2 q_r y_{ij}^{kr} + \sum_{k\in K}\sum_{(i,j)\in A_d} c_k^2 q_r y_{ij}^{kr} + \sum_{k\in K}\sum_{(i,j)\in A} c_k^4 e_k q_r d_{ij}^k y_{ij}^{kr} \tag{81}$$

If there are no other costs, such as delay penalty, storage cost, and waiting cost, the insertion cost of bundle requests will be the same for the same route(s). If the best position of a request is found greedily, then it is also the best position for bundle requests when there is only basic cost. After each insertion, the bundle requests will be inserted into the same positions when it passes the feasibility check and there is only a basic cost. In this way, the computation time can be saved by not considering other possible positions. However, maybe there are other requests more suitable for this vehicle. Therefore, not all bundle requests will be inserted, which will avoid occupying too much capacity of this vehicle. The number of inserted requests in the bundle is randomly chosen based on distribution $[x_1, x_2, \ldots, x_3]$ for $[1, 2, \ldots, m]$, where m is the number of requests in the bundle, $x_1 = 1/\varsigma$ and $x_i = x_{i-1}/\varsigma$ when $i > 1$, where ς is a parameter for adjusting the distribution.

Table 12
Parameters used in the paper.

Parameter	Value	Parameter	Value	Parameter	Value
c^1_{truck}	30.98	c^1_{train}	7.54	c^1_{barge}	0.6122
$c^{1'}_{truck}$	0.2758	$c^{1'}_{train}$	0.0635	$c^{1'}_{barge}$	0.0213
c^2_{truck}	3	c^2_{train}	18	c^2_{barge}	18
c^3_{truck}	1	c^3_{train}	1	c^3_{barge}	1
c^4_{truck}	8	c^4_{train}	8	c^4_{barge}	8
c^5_{truck}	1	c^5_{train}	1	c^5_{barge}	1
e_{truck}	0.8866	e_{train}	0.3146	e_{barge}	0.2288
γ_{1-4}	0.25	ξ	1.3	σ	0.5
ς	1.1	ρ	0.2	φ	1.3
t_1	0	t_2	5	t_3	7
t_4	9	t_5	13	t_6	13
t_7	17	t_8	19	t_9	21
t_{10}	24	α	2	β	1.5
ϖ_1	0.5	ϖ_2	0.2	ϖ_3	0.3

Appendix D. Parameters used in this paper

Most parameters are derived from Guo et al. (2020) and Demir et al. (2016), and the rest parameters are defined by tuning ALNS. Part of parameters are shown in Table 12.

Note that the cost parameters in Table 12 are used in the comparison with results of Guo et al. (2020). Demir et al. (2016) use different cost parameters for different vehicles/terminals. The values of other parameters which have different values for different vehicles/requests/terminals can be found at a research data website.[3]

References

Agamez-Arias, A.-d.-M., Moyano-Fuentes, J., 2017. Intermodal transport in freight distribution: a literature review. Transp. Rev. 37 (6), 782–807.

Ambra, T., Caris, A., Macharis, C., 2019. Towards freight transport system unification: reviewing and combining the advancements in the physical internet and synchromodal transport research. Int. J. Prod. Res. 57 (6), 1606–1623.

Behdani, B., Fan, Y., Wiegmans, B., Zuidwijk, R., 2014. Multimodal schedule design for synchromodal freight transport systems. Eur. J. Transp. Infrastruct. Res. 16 (3), 424–444.

Cornuéjols, G., 2008. Valid inequalities for mixed integer linear programs. Math. Program. 112 (1), 3–44.

Danloup, N., Allaoui, H., Goncalves, G., 2018. A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. Comput. Oper. Res. 100, 155–171.

Delbart, T., Molenbruch, Y., Braekers, K., Caris, A., 2021. Uncertainty in intermodal and synchromodal transport: Review and future research directions. Sustainability 13 (7), 3980.

Demir, E., Burgholzer, W., Hrušovskỳ, M., Arıkan, E., Jammernegg, W., Van Woensel, T., 2016. A green intermodal service network design problem with travel time uncertainty. Transp. Res. B 93, 789–807.

Di Febbraro, A., Sacco, N., Saeednia, M., 2016. An agent-based framework for cooperative planning of intermodal freight transport chains. Transp. Res. C 64, 72–85.

Dong, L., Miao, G., Wen, W., 2021. China's carbon neutrality policy: Objectives, impacts and paths. East. Asian. Policy 13 (01), 5–18.

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

Drexl, M., 2013. Applications of the vehicle routing problem with trailers and transshipments. European J. Oper. Res. 227 (2), 275–283.

Drexl, M., 2014. Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. Networks 63 (1), 119–133.

EEA, 2020. European Environment Agency (EEA): Greenhouse gas emissions from transport in Europe. https://www.eea.europa.eu/data-and-maps/indicators/transport-emissions-of-greenhouse-gases-7/assessment [Online; accessed 14-September-2021].

Ghane-Ezabadi, M., Vergara, H.A., 2016. Decomposition approach for integrated intermodal logistics network design. Transp. Res. E 89, 53–69.

Ghilas, V., Demir, E., Van Woensel, T., 2016. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. Comput. Oper. Res. 72, 12–30.

Giusti, R., Manerba, D., Bruno, G., Tadei, R., 2019. Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues. Transp. Res. E 129, 92–110.

Grangier, P., Gendreau, M., Lehuédé, F., Rousseau, L.-M., 2016. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. European J. Oper. Res. 254 (1), 80–91.

Guo, W., Atasoy, B., Beelaerts van Blokland, W.W.A., Negenborn, R.R., 2020. A dynamic shipment matching problem in hinterland synchromodal transportation. Decis. Support Syst. 134, 113289.

Hojabri, H., Gendreau, M., Potvin, J.-Y., Rousseau, L.-M., 2018. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. Comput. Oper. Res. 92, 87–97.

Hrušovskỳ, M., Demir, E., Jammernegg, W., Van Woensel, T., 2018. Hybrid simulation and optimization approach for green intermodal transportation problem with travel time uncertainty. Flexible Serv. Manuf. J. 30 (3), 486–516.

Kallas, S., 2011. Transport 2050: Comission Outlines Ambitious Plan to Increase Mobility and Reduce Emmisions. Technical report, Technical Report March.

Li, L., Negenborn, R.R., De Schutter, B., 2015. Intermodal freight transport planning–A receding horizon control approach. Transp. Res. C 60, 77–95.

[3] https://figshare.com/s/2bbc4c63fd9a7200594f.

Lin, M.-H., Carlsson, J.G., Ge, D., Shi, J., Tsai, J.-F., 2013. A review of piecewise linearization methods. Math. Probl. Eng. 2013, 101376.
Liu, R., Tao, Y., Xie, X., 2019. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. Comput. Oper. Res. 101, 250–262.
Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transp. Sci. 47 (3), 344–355.
Mes, M.R., Iacob, M.-E., 2016. Synchromodal transport planning at a logistics service provider. In: Logistics and Supply Chain Innovation. Springer, pp. 23–36.
Mitrović-Minić, S., Laporte, G., 2006. The pickup and delivery problem with time windows and transshipment. INFOR: Inf. Syst. Oper. Res. 44 (3), 217–227.
Moccia, L., Cordeau, J.-F., Laporte, G., Ropke, S., Valentini, M.P., 2011. Modeling and solving a multimodal transportation problem with flexible-time and scheduled services. Networks 57 (1), 53–68.
Öncan, T., Altınel, İ.K., Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. Comput. Oper. Res. 36 (3), 637–654.
Qu, Y., Bard, J.F., 2012. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. Comput. Oper. Res. 39 (10), 2439–2456.
Qu, W., Rezaei, J., Maknoon, Y., Tavasszy, L., 2019. Hinterland freight transportation replanning model under the framework of synchromodality. Transp. Res. E 131, 308–328.
Rais, A., Alvelos, F., Carvalho, M.S., 2014. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. European J. Oper. Res. 235 (3), 530–539.
Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.
Sarasola, B., Doerner, K.F., 2020. Adaptive large neighborhood search for the vehicle routing problem with synchronization constraints at the delivery location. Networks 75 (1), 64–85.
Shang, J.S., Cuff, C.K., 1996. Multicriteria pickup and delivery problem with transfer opportunity. Comput. Ind. Eng. 30 (4), 631–645.
SteadieSeifi, M., Dellaert, N.P., Nuijten, W., Van Woensel, T., Raoufi, R., 2014. Multimodal freight transportation planning: A literature review. European J. Oper. Res. 233 (1), 1–15.
UNCTAD, 2020. United nations conference on trade and development (UNCTAD): Review of maritime transport 2020. https://unctad.org/system/files/official-document/rmt2020_en.pdf [Online; accessed 14-September-2021].
Van Riessen, B., Negenborn, R.R., Dekker, R., 2015. Synchromodal container transportation: an overview of current topics and research opportunities. In: International Conference on Computational Logistics. Springer, Delft, The Netherlands, pp. 386–397.
Van Riessen, B., Negenborn, R.R., Dekker, R., Lodewijks, G., 2013. Service Network Design for an Intermodal Container Network with Flexible Due Dates/times and the Possibility of Using Subcontracted Transport. Technical report.
Wolfinger, D., 2021. A large neighborhood search for the pickup and delivery problem with time windows, split loads and transshipments. Comput. Oper. Res. 126, 105110.
Wolfinger, D., Salazar-González, J.-J., 2021. The pickup and delivery problem with split loads and transshipments: A branch-and-cut solution approach. European J. Oper. Res. 289 (2), 470–484.
Wolfinger, D., Tricoire, F., Doerner, K.F., 2019. A matheuristic for a multimodal long haul routing problem. Eur. J. Transp. Logist. 8 (4), 397–433.
Zhang, Y., Atasoy, B., Negenborn, R.R., 2022. Preference-based multi-objective optimization for synchromodal transport using Adaptive Large Neighborhood Search. Transp. Res. Rec. 2676 (3), 71–87.
Zhang, Y., Atasoy, B., Souravlias, D., Negenborn, R.R., 2020. Pickup and delivery problem with transshipment for Inland Waterway Transport. In: Proceedings of the International Conference on Computational Logistics. Springer, Twente, The Netherlands, pp. 18–35.
Zhang, M., Pel, A., 2016. Synchromodal hinterland freight transport: Model study for the port of Rotterdam. J. Transp. Geogr. 52, 1–10.