Transportation Research Part B 141 (2020) 164–195

Contents lists available at ScienceDirect

Transportation Research Part B

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

Dynamic discretization discovery for the service network design problem with mixed autonomous fleets

Yannick Oskar Scherr[a,*], Mike Hewitt[b], Bruno Albert Neumann Saavedra[a], Dirk Christian Mattfeld[a]

[a] Technische Universität Braunschweig, Mühlenpfordtstraße 23, Braunschweig 38106, Germany
[b] Quinlan School of Business, Loyola University, 1 E. Pearson, Chicago, IL 60611, USA

ARTICLE INFO

Article history:
Received 17 December 2019
Revised 8 August 2020
Accepted 16 September 2020
Available online 26 September 2020

Keywords:
Service network design
Time-expanded network
Dynamic discretization discovery
Automated driving
Platooning
Two-tier city logistics

ABSTRACT

We consider a service network design problem for the tactical planning of parcel delivery in a city logistics setting. A logistics service provider seeks a repeatable plan to transport commodities from distribution centers on the periphery to inner-city satellites. In a heterogeneous infrastructure, autonomous vehicles in level 4 may only drive in feasible streets but need to be pulled elsewhere by manually operated vehicles in platoons. We formulate an integer program to determine the fleet mix, schedule transportation services, and decide on the routing or outsourcing of commodities. Platooning requires a high level of synchronization between vehicles which demands the time-expanded networks to contain narrow time intervals. Thus, we develop an algorithm based on the dynamic discretization discovery scheme which refines partially time-expanded networks iteratively without having to enumerate the fully time-expanded network a priori. We introduce valid inequalities and provide two enhanced versions of the algorithm that exploit linear relaxations of the problem. Further, we propose heuristic ideas to speed up the search for high-quality solutions. In a computational study, we analyze the efficacy of the algorithm in different versions and observe improvements of computational performance in comparison to a commercial solver. Finally, we solve a case study on a real-world based network to obtain insights into the deployment of a mixed autonomous fleet in an existing heterogeneous infrastructure.

1. Introduction

We consider a two-tier city logistics setting in which a logistics service provider (LSP) delivers parcels (Crainic, 2008). In the first tier, parcels are consolidated in external zones, i.e., distribution centers on the city's periphery, and need to be transported to satellites that are distributed within the city. Satellites act as transshipment points to a second tier in which parcels are delivered to customers or are available for pickup. We focus on the first tier only for which repeatable master tours are determined by solving a service network design (SND) problem for tactical planning (Crainic et al., 2009).

* Corresponding author.
E-mail addresses: y.scherr@tu-bs.de (Y.O. Scherr), mhewitt3@luc.edu (M. Hewitt).

https://doi.org/10.1016/j.trb.2020.09.009

One technology LSPs are increasingly adopting is automated driving and there have been numerous efforts for utilizing autonomous vehicles (AVs) within the delivery process in recent years (CB Insights, 2019). Tests show that vehicles are already prepared to drive autonomously in safe areas, called henceforth AV zones, within "geofences" (Crosbie, 2017). These AV zones dispose of adequate infrastructure like dedicated lanes, specific traffic rules, or technologies like roadside sensors, machine-readable signs, among others (WSP, 2016; Coker, 2018). However, AV zones in cities are still scarce. Preparing cities for fully autonomous driving everywhere will likely take decades because of costly infrastructure enhancements and discussions among stakeholders about safety, legal, and ethical aspects (Mahmassani, 2016). As we consider level 4 of driving automation, AVs require supervision outside of AV zones given such a heterogeneous infrastructure (SAE International, 2018). To this end, a manually operated vehicle (MV) may be linked to one or more AVs in a platoon. The human driver of the MV guides the movements of the platoon, while the AVs behind synchronously accelerate or brake (ACEA, 2017). For example, AVs can be pulled from external zones to inner-city AV zones within platoons, perform deliveries at the same time as MVs do, and are later pulled towards the external zone again.

Tactical planning aims to determine a repeatable plan to utilize resources in the most efficient way based on demand forecasts over a medium-term horizon (Crainic and Laporte, 1997). In our problem setting, the regular utilization of MVs and AVs is critical to the LSP as it impacts long-term and costly decisions regarding the fleet. Also, operating a mixed autonomous fleet can result in significant savings on driver salary expenses for LSPs. The problem of determining the fleet size and mix, routing the MVs and AVs including their synchronization to form platoons, and routing the commodities yields the recently introduced service network design problem with mixed autonomous fleets (SNDMAF).

Scherr et al. (2019) formulate an integer program (IP) for this problem and evaluate this one generic model on networks with different heterogeneous infrastructure that imply strategies on how the LSP operates its mixed autonomous fleet. The three evaluated types of heterogeneous infrastructure, namely corridor, artery, and nanny, describe the role of the AVs in synchronizing with MVs. In the real world, however, AV zones are determined by urban planning authorities and their distribution within the city outlines a heterogeneous infrastructure that may not fall into one of these types. As a motivating example, we consider the city ring road of Braunschweig, Germany, that is prepared for automated driving as part of a research study since 2010 (Nothdurft et al., 2011) and combines aspects of the corridor and the artery heterogeneous infrastructure. In a case study in this paper, we use this network to investigate the deployment of a mixed autonomous fleet in an existing heterogeneous infrastructure.

SND problems are typically formulated on time-expanded networks in which the planning horizon for delivery activities is discretized in time periods. In these networks, a node depicts a location and a time period, whereas the arcs connecting nodes represent services for routing commodities. A fully time-expanded network with fine discretization considering all nodes and arcs provides rich possibilities for synchronizing delivery activities. However, models on fully time-expanded networks are typically too large to be solved using a standard solver (Boland et al., 2018). Previous studies have shown that the SNDMAF is particularly hard to solve which prevents research on real-world sized instances (Scherr et al., 2018; 2019).

With this paper, we provide the following contributions. The model we formulate, compared to the model in Scherr et al. (2019), captures two additional features of city logistics problems. We consider outsourcing of deliveries to third-party service providers as well as prohibit the splitting of commodities to increase operational efficiency. Our methodological contributions comprise the customization and the enhancement of the dynamic discretization discovery (DDD) scheme to provide high-quality solutions to the SNDMAF without building the fully time-expanded network explicitly. The DDD scheme iteratively refines and solves SND problems on partially time-expanded networks, in which only a limited number of nodes and arcs is evaluated (Boland et al., 2017). We extend this general scheme, developing the DDD-SNDMAF, to tackle the problem characteristics and the complexity of the SNDMAF, such that we: 1. add additional properties to the partially time-expanded networks that account for asset management, 2. introduce valid inequalities to strengthen the lower bound by better estimating the fleet size, and 3. solve an IP on a time-expanded network of reduced size to obtain a feasible solution and an upper bound in any iteration. We show that the DDD-SNDMAF is able to solve larger instances faster than an off-the-shelf solver.

However, it still requires solving SNDMAF instances that grow larger and harder to solve in every DDD iteration. Therefore, we propose two enhancements that exploit relaxations to speed up the algorithm but maintain its exact nature. First, we adapt the idea of Hewitt (2019), that proposes a two-phase DDD wherein the first phase solves the linear relaxation of the problem. We consider further relaxing the problem in the first phase by ignoring capacity constraints. In addition, we propose a scheme for dynamically determining which variables should have their domain relaxed in the MIP that is solved to obtain the lower bound. Lastly, we propose heuristic ideas to speed up the search for high-quality primal solutions by restricting the search space of the SNDMAF problems and the exploration of the partially time-expanded networks.

We evaluate the effects of the algorithmic advancements by solving artificially generated instances. Then, we apply the best-performing algorithms to a case study on a realistic network that resembles the street network of Braunschweig. Still, an assignment of AVs to platoons as well as commodities to vehicles is necessary to implement SNDMAF solutions in an operational level. Therefore, we propose a post-processing technique for operationalizing an SNDMAF solution such that the transshipment effort is minimal.

The structure of this article is as follows. In Section 2, we review related literature. We then describe the SNDMAF and formulate the mathematical model as well as the post-processing program as IPs in Section 3. We dedicate Section 4 to the description and evaluation of the DDD-SNDMAF algorithm and its enhancements. In Section 5, we consider the Braunschweig case study. Finally, we conclude and provide perspectives for future work in Section 6.

2. Literature review

In this section, we first give an overview of related literature considering AVs and platooning. We then review SND literature with a focus on problems with resource management considerations and city logistics applications. Finally, we review literature on dynamically refining partially time-expanded networks and proceed with investigating adaptations of the DDD scheme as a solution method to SND-related problems.

2.1. Autonomous vehicles and platooning

AVs are expected to be widely adopted by LSPs in the near future, particularly in the parcel delivery market (Heutger and Kückelhaus, 2014; Joerss et al., 2016). In the operations research literature, there is a strong focus on unmanned aerial vehicles or drones. Several works consider truck-and-drone routing problems in which a truck carries a number of drones that are able to take off from the truck's roof, perform a delivery, and meet with the truck again. Murray and Chu (2015), Poikonen et al. (2017), Agatz et al. (2018), Wang and Sheu (2019), and Murray and Raj (2020) state this problem in similar ways that differ in the number of trucks and drones. Poikonen and Golden (2019) consider multiple visits of a drone between take-off and landing. Boysen et al. (2018b) adapt this concept to small robots that are deployed from trucks and move on sidewalks. Closely related are truck-and-trailer routing problems as in Derigs et al. (2013) or Meisel and Kopfer (2014), in which trucks are active and trailers are passive means of transportation.

Platooning technologies can contribute to reducing CO_2 emissions and utilizing roads more efficiently (Lammert et al., 2014; Lioris et al., 2016). Larsson et al. (2015) introduce the vehicle platooning problem and formulate an IP to derive fuel-optimal solutions for grouping trucks with an origin and a destination to form platoons. Due to the NP-hardness of the problem, computational results are only provided for the so-called unlimited platooning problem which neglects delivery deadlines. Boysen et al. (2018a) solve instances with time windows on networks where all trucks share an identical path. Platooning technology also allows to automate the following vehicles as Schmeitz et al. (2019) show in a real-world use case in Eindhoven as part of the EU AUTOPILOT project. Utilizing the benefits of platooning raises a variety of planning issues which Bhoopalam et al. (2018) point out in a comprehensive literature review that focuses on truck platooning on highways. There is also work related to urban environments, see e.g. Sebe et al. (2019) who group cross-provider platoons based on origin-destination pairs of autonomous pods.

The SNDMAF utilizes the technology of platooning to achieve aspects of synchronization that are similar to truck-and-drone and truck-and-trailer routing problems, but it differs from those problems in other regards. Compared to drones, AVs rely on the street network with restricted areas for autonomous driving, which appears to be a more realistic scenario in a crowded city environment. Further, the capacity of road vehicles generally tends to be much larger than that of drones. Compared to trailers that are passive in any case, AVs can at least drive autonomously in AV zones. Finally, the AVs in the SNDMAF are always required to actively drive even if they are following an MV, thus consuming more energy than drones idling on top of a truck and trailers being physically towed by a truck.

2.2. Service network design

SND problems address the tactical planning level that regards a mid-term planning scope and focuses on the efficient use of assets on a regular basis (Crainic and Rousseau, 1986). Services represent transportation legs or jobs that are operated by a service provider and enable the routing of commodities through the designed service network. Crainic (2000) and Wieberneit (2008) provide literature reviews on SND problems and their different applications in the area of transportation. The first SND problems were modeled on static networks that consider the spatial location of nodes or some kind of logical structure. With more and more temporal attributes coming into play, such as departure and arrival times for commodities, travel times for vehicles, and shift lengths for staff, time is explicitly modeled as an attribute of the network. In time-expanded networks, also called time-space networks, locations are replicated in discrete time periods over a planning horizon. Thus, we obtain a static flow problem which is more tractable to solve (Skutella, 2009) as it does not require big-M constraints to consider temporal aspects (Vigo and Toth, 2014).

Over the recent years, the general SND problem was enriched to account for several planning issues that occur in practice. One such issue is the allocation and repositioning of resources, which comprises vehicles, power units, or staff, among others. Andersen et al. (2009) introduces service network design with asset management. Here, the available assets or resources need to follow cyclic routes. The selection of an asset is associated with fixed costs that typically predominate the service costs. Crainic et al. (2016) expand the formulation and introduce service network design with resource constraints. The resource bound constraints explicitly limit the number of resources available at a facility. Crainic et al. (2017) build upon these extensions but handle multiple types of resources with different features in their scheduled service network design problem with resource acquisition and management. That formulation includes a strategic layer that allows for the acquisition of resources as well as their (re)assignment to a facility.

In city logistics, tactical planning models are used by LSPs to enhance consolidation of transports. Crainic et al. (2009) develop a two-tier city logistics framework consisting of an SND problem that is solved in the first tier and a vehicle routing problem in the second tier. Considering both tiers in an integrated routing problem yields a complex model, see e.g. Perboli et al. (2011), which is why the tiers are often decomposed. The first-tier problem is further described by

Crainic and Sgalambro (2014) as the urban-vehicle service network design problem (UVSND). While the model ensures design balance of vehicles in an uncontrolled variant of the model, the controlled UVSND includes further resource management considerations.

2.3. Dynamic discretization discovery

Most of the SND literature considers fully time-expanded networks by replicating a physical network in discrete time steps. Recently, research also focuses on generating partially time-expanded networks, i.e., time-expanded networks with varying discretization. Fleischer and Skutella (2007) develop an approximation algorithm based on so-called condensed time-expanded networks to solve quickest-flow-over-time problems. Traveling salesman problems with time windows (TSPTW) are also studied in this context with a focus on time window discretization methods. Wang and Regan (2002) develop a method that iteratively refines the discretization of the time windows and converges to the optimal solution value (Wang and Regan, 2009). Dash et al. (2012) present a time bucket formulation for the TSPTW based on partitioning time windows into subwindows and propose an LP-based iterative refinement scheme to obtain good partitions.

Boland et al. (2017) are the first to consider partially time-expanded networks for a problem with design decisions and time windows, the continuous-time service network design problem (CTSNDP). They introduce the DDD scheme which we explain in more detail in Section 4. Recently, DDD is applied to a variety of problems in the area of transportation. Medina et al. (2019) present a combined service network design and routing problem in two formulations and solve it with DDD. Hewitt (2019) solves a continuous-time load plan design problem (CTLPDP) and provides enhancements to the DDD algorithm from which four of them can also be applied to the more general CTSNDP. He proposes a two-phase DDD with linear relaxations solved in the first phase and arc-time window inequalities that define feasible time windows in which a commodity can take an arc. Further, a symmetry-breaking procedure for the refinement of partially time-expanded networks is introduced as well as a branching rule. Vu et al. (2019) solve the time-dependent traveling salesman problem with time windows with a makespan and a duration objective. In each DDD iteration, they generate valid inequalities to the lower bound IP to eliminate subtours and obtain two feasible solutions, called candidate solutions, by solving IPs on two primal time-expanded networks.

The existing literature that is related to the DDD scheme, albeit small, already covers a set of important optimization problems. With the SNDMAF, we adapt DDD to a problem that is richer in terms of features than those problems. The SNDMAF distinguishes itself from the basic SND problem considered in Boland et al. (2017) in several ways. In particular, the SNDMAF: 1. respects asset management considerations, i.e., the number of vehicles entering a node must equal the number leaving it, 2. requires cyclic vehicle routes that start and end at the same location within the planning horizon, and 3. considers a heterogeneous fleet of vehicles that synchronize to form platoons. The first two items also apply to a number of related SND problems with resource management considerations we reviewed in Section 2.2.

3. Service network design with mixed autonomous fleets

In this section, we first define the SNDMAF problem, then formulate the mathematical model as an IP.

3.1. Problem description

We first describe the general setting for two-tier city logistics (Crainic et al., 2009). Then, we adapt this setting to consider AV zones for autonomous driving and the synchronization of MVs and AVs to form platoons. The general setting for two-tier city logistics considers two types of facilities, external zones and satellites. External zones are distribution centers, usually located at city borders. These external zones receive the commodities that carriers need to distribute within the city and facilitate consolidation of shipments. LSPs rely on a dedicated fleet of vehicles to transport commodities from external zones to satellites that are distributed within the city. Satellites are transshipment points without warehousing capacities where parcels are transferred to last-mile dedicated vehicles or pick-up stations. We assume that sufficient parking space for vehicles is available at external zones and satellites. Fig. 1 illustrates an example for the considered problem setting.

The goal in the first tier we focus on is to build a repeatable delivery plan to timely transport commodities from external zones to satellites using a cost-efficient operation of vehicles. To account for repeatability, the fleet's services need to be designed such that each vehicle starts from one of the external zones at the beginning of the day and returns to one of them at the end of the day. The number of vehicles at each external zone needs to be balanced between the beginning and end of each day.

Using the vehicles of the fleet, the LSP needs to satisfy demand for commodities that each feature a specific volume in discrete units. One unit may denote a pallet or a bag filled with parcels. The origin defines the external zone where the commodity is to be picked up and when it is available for doing so. The destination is composed of a satellite to which the commodity is delivered at a mandatory arrival time. We focus on instances of the problem that correspond to next-day delivery, wherein commodities become available at the external zone at the beginning of the day or they appear at a later time point during the day, which may enable same-day delivery. Commodities as a whole can be transshipped between vehicles at any location. Transshipment is a key factor of two-tier city logistics systems to enhance consolidation and avoid poorly loaded vehicles (Crainic, 2008). In recent practical applications, satellites are also denoted as transshipment points,

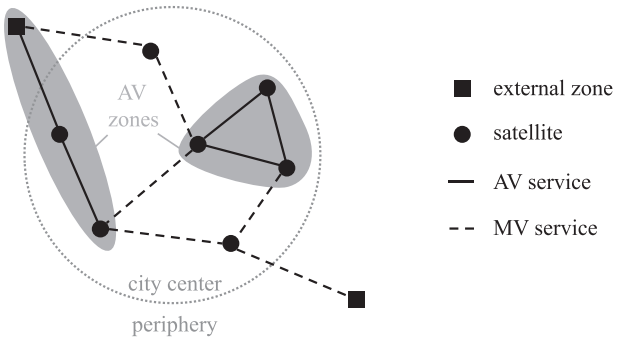


Fig. 1. Problem setting for the first tier of two-tier city logistics with a mixed autonomous fleet based on Scherr et al. (2019).

micro hubs, or micro-consolidation centers (Allen et al., 2012; Janjevic et al., 2013; Taniguchi et al., 2018; Urban Freight Lab, 2020). Splitting commodities and delivering parts of them over separate ways is prohibited.

Since commodities cannot be stored at satellites, their arrival times at satellites, which we define for the first tier, also imply the time points for synchronization with the second tier of city logistics. Rather than investigating this second tier closer in our problem we assume that commodities should arrive at satellites regularly spread over the day for steady supply. This is to account for the fact that, in the second tier, both last-mile vehicles, such as small vans or cargo bikes, and pickup stations usually have very limited capacity. Waiting delivery persons or empty pickup stations should be avoided as they are relatively costly and not satisfactory for customers. A delivery of a commodity to a satellite in the first tier can also be outsourced to a usually expensive third-party service.

In our problem, we state that LSPs employ two vehicle types, MVs and AVs. Each vehicle type is characterized by a transportation capacity, speed, and a fixed cost for acquisition and usage. We assume that MVs involve higher fixed costs because of the wages of drivers to operate them. The delivery plan determines the fleet size, including the number of MVs and AVs to be used. Using this fleet, transportation services are scheduled that form routes to transport commodities. The street infrastructure is heterogeneous so that AV zones are predefined. MVs have no restrictions to move across the heterogeneous infrastructure, whereas AVs may travel without supervision only in AV zones. The AV zones may be disconnected from each other, so AVs need to be pulled between them by means of platooning. To minimize negative effects on other traffic participants, each MV has a platooning capacity to pull only a limited number of AVs outside of AV zones. Both MVs and AVs are able to idle at any facility. To form platoons, MVs and AVs need to synchronize in space and time. Scherr et al. (2019) define three platoon operations that are performed to achieve synchronization: a vehicle can either split from, merge to, or transit within a platoon.

The aim of the SNDMAF is to produce a tactical plan based on demand forecasts over a medium-term time horizon, e.g., a season. The tactical plan primarily determines the number of vehicles performing particular services, thus resembling capacity planning. Moreover, the platoon size and platoon operations are determined for a specific location and time. The LSP aims to minimize the total costs composed of: 1. Fixed costs for acquiring and maintaining the fleet as well as for wages, 2. Service costs for routing vehicles between locations, 3. Transportation costs for transporting a volume of commodities on vehicles, and 4. Outsourcing costs for contracting a third-party service provider to deliver a commodity. This tactical plan can also inform other decision-making processes, such as scheduling workers at external zones and satellites or quoting transportation capabilities to potential customers in sales and marketing efforts. Incorporating operational detail in tactical planning may be counterproductive. Tactical planning is based on demand forecasts leading to tentative plans subject to revision at the operational level of implementation. To this end, incorporating operational detail at an early stage of planning may unnecessarily deteriorate solution quality.

A tactical plan, however, provides guidelines for a subsequent operational planning level, which we consider in a post-processing step to the SNDMAF. Whereas the SNDMAF is solved once to obtain a tactical plan, the post-processing step can be run on a daily basis with the input of this tactical plan and the respective commodity demand. Based on the aggregated vehicle capacities and commodity paths of the tactical plan, disaggregated routes for each individual vehicle need to be derived and commodities need to be assigned to vehicles. The objective of this operational problem is to minimize the transshipment effort associated with loading and transshipping commodities onto and between vehicles to achieve the consolidations prescribed in the tactical plan.

3.2. Mathematical model

We formulate the SNDMAF as an IP on a time-expanded network to explicitly consider the time-dependent properties of the problem. Table 1 provides the notation of the physical network and the time-expanded network, which we distinguish in the following. We define the physical network as a directed network $D_{ph} = (N_{ph}, A_{ph})$, also called the "flat" network, consisting of nodes N_{ph} representing locations and physical arcs A_{ph} representing the road connections between these locations. The node set N_{ph} contains subsets $N_E \subset N$ for external zones and $N_S \subset N$ for satellites with $N_E \cap N_S = \emptyset$. Regarding the physical

Table 1
Notation of the physical and time-expanded network.

Type	Notation	Description
Physical network D_{ph}	$i \in N_{ph}$	Location
	$i \in N_E$	External zone
	$i \in N_S$	Satellite
	$(i, j) \in A_{ph}$	Physical arc
	$(i, j) \in A_M$	MV arc
	$(i, j) \in A_A$	AV arc
	$\tau_{ij} \in \mathbb{N}$	Travel time on physical arc $(i, j) \in A_{ph}$
Time-expanded network D	$t \in T = \{0, \ldots, t_{\max}\}$	Time period in planning horizon $t_{\max}$
	$(i, t) \in N$	Node
	$((i,t),(j,\bar{t})) \in A$	Arc
	$((i,t),(j,\bar{t})) \in A_\gamma$	Auxiliary arc to assign number of vehicles
	$((i,t),(j,\bar{t})) \in A_h$	Holding arc to idle at external zone
	$((i,t),(j,\bar{t})) \in A_w$	Waiting arc to idle at satellite
	$((i,t),(j,\bar{t})) \in A_m$	Movement arc to travel between locations

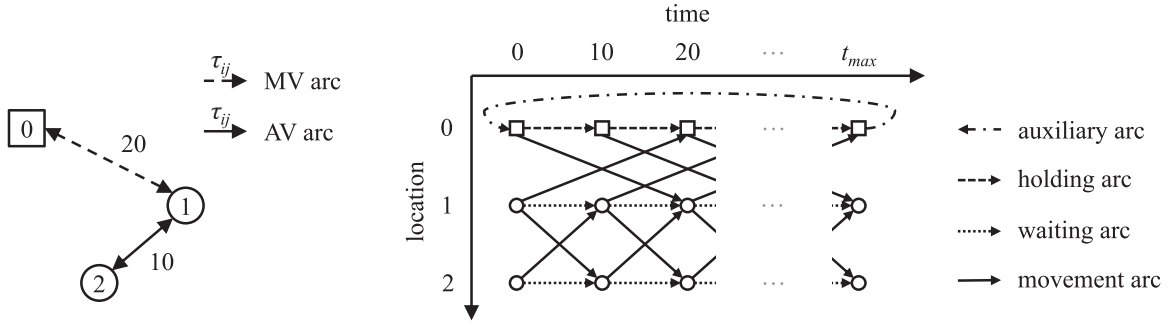


Fig. 2. Notation of the time-expanded network (right) based on a physical network (left).

arcs, we distinguish between AV arcs $A_A \subset A_{ph}$, on which both vehicle types are able to travel alone, and MV arcs $A_M \subset A_{ph}$, on which AVs need to be pulled by MVs. Physical arcs with $i = j$ are included in the set of AV arcs $(i, j) \in A_A$ to permit AVs to idle at any location.

In the time-expanded network $D = (N, A)$, also a directed network, locations are replicated in discrete time steps $t \in T = \{0, \ldots, t_{\max}\}$ over the length of the planning horizon $t_{\max}$. We obtain the nodes $(i, t) \in N$, and an arc $((i,t),(j,\bar{t})) \in A$ connects two nodes of the time-expanded network. We define three types of arcs, called auxiliary arcs, holding or waiting arcs, and movement arcs. Auxiliary arcs $A_\gamma \subset A$ allow us to model the fleet assignment as well as the cyclic vehicle routes. For this reason, they go back in time and connect the last $(t_{\max})$ with the first (t_0) replication of a respective external zone. Holding and waiting arcs link the same logical representatives of locations in subsequent time periods to consider idling vehicles. Holding arcs $A_h \subset A$ are used for external zones and waiting arcs $A_w \subset A$ for satellites. We make this distinction because holding arcs also enable commodities to wait at external zones, whereas waiting arcs only allow for MVs and AVs to wait. Movement arcs $A_m \subset A$ encode traveling between locations and the respective travel time τ_{ij} between locations determines the length of these arcs. Fig. 2 illustrates the notation of a time-expanded network by means of an exemplary physical network.

We define the set K in which each commodity $k \in K$ has a volume v^k and needs to be transported from an origin external zone $e^k = (i_e^k, t_e^k)$ to a destination satellite $s^k = (i_s^k, t_s^k)$. Since we expect timely synchronization with the second tier of city logistics, we define an exact arrival time at the destination satellite. We do not permit splitting a commodity during its transportation from its origin external zone to its destination satellite, so we denote the flow of commodities on an arc as binary variables $x_{ij}^{kt\bar{t}} \in \{0, 1\}$. Outsourcing the delivery of commodities to a third party is modeled using binary variables $y^k \in \{0, 1\}$.

Integer variables $m_{ij}^{t\bar{t}}$ and $a_{ij}^{t\bar{t}}$ indicate the respective number of services installed on an arc $((i,t),(j,\bar{t})) \in A$. A service installed on an auxiliary arc $((i,t),(j,\bar{t})) \in A_\gamma$ implies that a vehicle is acquired for the complete planning horizon and a fixed cost for this acquisition is assigned. The human-driven MVs can travel on all physical arcs A_{ph}. AVs are driverless and can only move alone on AV arcs $A_A \subset A_{ph}$ that include all holding and waiting arcs permitting that AVs can idle at any location. AVs cannot travel alone on arcs of the subset $A_M \subset A_{ph}$, however, an MV can pull up to n_{ij} AVs by means of platooning on these arcs. To build platoons, a group of vehicles need to synchronize in space and time by using the same MV arc of the time-expanded network $((i,t),(j,\bar{t})) \in A : (i, j) \in A_M$. Table 2 displays the variables and parameters.

Table 2
Notation of the variables and parameters.

Type	Notation	Description
Variables	$m_{ij}^{t\bar{t}} \in \mathbb{N}$	MV services on arc $((i,t),(j,\bar{t})) \in A$
	$a_{ij}^{t\bar{t}} \in \mathbb{N}$	AV services on arc $((i,t),(j,\bar{t})) \in A$
	$x_{ij}^{kt\bar{t}} \in \{0,1\}$	Flow of commodity $k \in K$ on arc $((i,t),(j,\bar{t})) \in A$
	$y^k \in \{0,1\}$	Third-party service for commodity $k \in K$
Parameters	$f_M \in \mathbb{R}^+$	Fixed cost for assigning an MV
	$f_A \in \mathbb{R}^+$	Fixed cost for assigning an AV
	$g_{ij} \in \mathbb{R}^+$	Service cost per MV service on arc $(i,j) \in A_{ph}$
	$l_{ij} \in \mathbb{R}^+$	Service cost per AV service on arc $(i,j) \in A_{ph}$
	$c_{ij}^k \in \mathbb{R}^+$	Transportation cost per commodity $k \in K$ on arc $(i,j) \in A_{ph}$
	$b^k \in \mathbb{R}^+$	Cost for outsourcing delivery of one unit of commodity k to third-party service
	$k \in K$	Commodities
	$v^k \in \mathbb{N}$	Volume in units of commodity $k \in K$
	$e^k = (i_e^k, t_e^k) \in N$	External zone and departure time (origin) of commodity $k \in K$
	$s^k = (i_s^k, t_s^k) \in N$	Satellite and arrival time (destination) of commodity $k \in K$
	$u_M \in \mathbb{N}$	Transportation capacity of MVs in volume units
	$u_A \in \mathbb{N}$	Transportation capacity of AVs in volume units
	$n_{ij} \in \mathbb{N}$	Platoon capacity on arc $(i,j) \in A_{ph}$ (max. number of AVs in platoon)

The **SNDMAF model** we seek to solve is as follows:

$$\min \sum_{((i,t),(j,\bar{t}))\in A_\gamma} \left[f_M m_{ij}^{t\bar{t}} + f_A a_{ij}^{t\bar{t}}\right] + \sum_{((i,t),(j,\bar{t}))\in A_m} \left[g_{ij} m_{ij}^{t\bar{t}} + l_{ij} a_{ij}^{t\bar{t}} + \sum_{k\in K} c_{ij}^k x_{ij}^{kt\bar{t}}\right] + \sum_{k\in K} b^k v^k y^k \tag{1}$$

subject to

$$\sum_{((j,\bar{t}),(i,t))\in A} x_{ji}^{k\bar{t}t} - \sum_{((i,t),(j,\bar{t}))\in A} x_{ij}^{kt\bar{t}} = \begin{cases} -1 + y^k, & (i,t) = e^k \\ 1 - y^k, & (i,t) = s^k, \\ 0, & (i,t) \neq e^k, s^k, \end{cases} \qquad \forall k \in K,\ (i,t) \in N \tag{2}$$

$$\sum_{k\in K} x_{ij}^{kt\bar{t}} v^k \leq u_M m_{ij}^{t\bar{t}} + u_A a_{ij}^{t\bar{t}}, \qquad \forall ((i,t),(j,\bar{t})) \in A \setminus A_h \tag{3}$$

$$a_{ij}^{t\bar{t}} \leq n_{ij} m_{ij}^{t\bar{t}}, \qquad \forall ((i,t),(j,\bar{t})) \in A_m : (i,j) \in A_M \tag{4}$$

$$\sum_{((j,\bar{t}),(i,t))\in A} m_{ji}^{\bar{t}t} = \sum_{((i,t),(j,\bar{t}))\in A} m_{ij}^{t\bar{t}}, \qquad \forall (i,t) \in N \tag{5}$$

$$\sum_{((j,\bar{t}),(i,t))\in A} a_{ji}^{\bar{t}t} = \sum_{((i,t),(j,\bar{t}))\in A} a_{ij}^{t\bar{t}}, \qquad \forall (i,t) \in N \tag{6}$$

$$x_{ij}^{kt\bar{t}} \in \{0,1\}, \qquad \forall k \in K,\ ((i,t),(j,\bar{t})) \in A \tag{7}$$

$$y^k \in \{0,1\}, \qquad \forall k \in K \tag{8}$$

$$m_{ij}^{t\bar{t}},\ a_{ij}^{t\bar{t}} \in \mathbb{N}, \qquad \forall ((i,t),(j,\bar{t})) \in A \tag{9}$$

The objective function (1) minimizes the total costs of the LSP. Assigning a vehicle for the complete planning horizon induces fixed costs f_M per MV and f_A per AV. The fixed costs are triggered by service variables on the auxiliary arcs that wrap around from the last time point of an external zone to its first time point. Although Scherr et al. (2019) consider this cost as well, we make it more explicit with a separate term. The service costs of MVs (g_{ij}) and AVs (l_{ij}) to perform transportation services are considered on movement arcs. In addition, transporting commodities induces transportation costs c_{ij}^k and outsourcing the delivery of commodities to third-party services causes costs b^k per unit of commodity volume.

Constraints (2) conserve the flow of commodities $x_{ij}^{kt\bar{t}}$ or alternatively fulfill demand by outsourcing to a third-party service with y^k. Constraints (3) limit the transportation capacity to the capacity of MVs (u_M) and AVs (u_A). The platooning constraints (4) apply to all MV arcs and assign AV services to platoons with limited platoon capacity n_{ij}. Constraints (5) and (6), respectively, are the design-balance constraints for MV and AV services that ensure their connection to feasible routes. The domains of the binary variables for commodities are defined in (7) and (8), the MV and AV service variables are defined as integer in (9).

We introduce an additional set of constraints to complement the coupling constraints (3), which limit the commodity flow to the capacity of MVs and AVs on any arc. Disaggregating these coupling constraints strengthens the LP relaxation of the model, as they consider each commodity individually rather than the sum of commodities. In our model, we apply this type of constraint to all movement and waiting arcs, as these are the arc types on which commodities need to be coupled with vehicles. We apply constraints (10) to waiting arcs and movement arcs on AV arcs. On these types of arcs, both MVs and AVs could travel alone. The constraints ensure that a commodity flows on the arc if at least one vehicle of either type travels on it. On MV arcs, however, AVs can only move in platoons pulled by an MV. Thus, it is sufficient to only consider MVs for the disaggregate coupling constraints (11) here.

$$x_{ij}^{kt\bar{t}} \leq m_{ij}^{t\bar{t}} + a_{ij}^{t\bar{t}}, \qquad \forall k \in K, ((i,t),(j,\bar{t})) \in A_w \cup A_m : (i,j) \in A_A \tag{10}$$

$$x_{ij}^{kt\bar{t}} \leq m_{ij}^{t\bar{t}}, \qquad \forall k \in K, ((i,t),(j,\bar{t})) \in A_m : (i,j) \in A_M \tag{11}$$

As we aim to further tighten the transportation capacity constraints (3), we can replace them with Constraints (12). These additionally consider the total possible commodity volume $V_{ij}^{t\bar{t}}$, i.e., the aggregated volume of all commodities that can feasibly flow on an arc. We determine this feasibility by checking whether a commodity is able to flow on an arc given its earliest departure time from and latest arrival time at a location. These time windows for each location are based on feasible time-dependent paths between the earliest departure time at the origin external zone and the latest arrival time at the destination satellite.

$$\sum_{k \in K} x_{ij}^{kt\bar{t}} v^k \leq m_{ij}^{t\bar{t}} \min\{u_M, V_{ij}^{t\bar{t}}\} + a_{ij}^{t\bar{t}} \min\{u_A, V_{ij}^{t\bar{t}}\}, \qquad \forall ((i,t),(j,\bar{t})) \in A \setminus A_h \tag{12}$$

As a solution to the SNDMAF needs to be interpreted in a post-processing step before it can be implemented, we describe an IP for the operational planning level. The program derives a disaggregated routing for individual vehicles from the aggregated flows in the SNDMAF solution. To this end, we make use of the terms $\hat{m}_{ij}^{t\bar{t}}$ and $\hat{a}_{ij}^{t\bar{t}}$ to indicate the respective number of MV and AV services performed on arc $((i,t),(j,\bar{t})) \in A$. Then, we define $A^{\mathcal{M}} \subset A$ as the set of arcs used by at least one MV in the solution to the SNDMAF, i.e., $((i,t),(j,\bar{t})) \in A^{\mathcal{M}} \leftrightarrow \hat{m}_{ij}^{t\bar{t}} \geq 1$. Similarly, let $A^{\mathcal{A}} \subset A$ be the set of arcs used by at least one AV in an SNDMAF solution, with $((i,t),(j,\bar{t})) \in A^{\mathcal{A}} \leftrightarrow \hat{a}_{ij}^{t\bar{t}} \geq 1$. Note that this program is typically smaller than the SNDMAF as we apply its variables and constraints only to arcs in the time-expanded network that have been used in the SNDMAF solution.

The sets $P = \{p\}$ and $Q = \{q\}$, respectively, represent the fleet of MVs and AVs. The magnitude of these sets equals the total solution values on the auxiliary arcs as they denote the total number of vehicles, hence, $|P| = \sum_{((i,t),(j,\bar{t})) \in A_\gamma^{\mathcal{M}}} \hat{m}_{ij}^{t\bar{t}}$ and $|Q| = \sum_{((i,t),(j,\bar{t})) \in A_\gamma^{\mathcal{A}}} \hat{a}_{ij}^{t\bar{t}}$. We define the binary variable $m_{ijp}^{t\bar{t}}$ taking the value of one if the MV p travels through arc $((i,t),(j,\bar{t})) \in A^{\mathcal{M}}$, otherwise zero. Similarly, the binary variable $a_{ijq}^{t\bar{t}}$ equals one if the AV q travels through arc $((i,t),(j,\bar{t})) \in A^{\mathcal{A}}$, otherwise zero. The binary variable $\pi_{ijpq}^{t\bar{t}}$ takes the value of one if the MV p pulls the AV q via platooning on an MV movement arc $((i,t),(j,\bar{t})) \in A_m^{\mathcal{A}} : (i,j) \in A_M$, otherwise zero.

The term $\hat{x}_{ij}^{kt\bar{t}}$ signals the existence of a flow of commodity $k \in K$ on arc $((i,t),(j,\bar{t})) \in A^{\mathcal{M}} \cup A^{\mathcal{A}}$ in the SNDMAF solution. These commodity paths and volumes, that are based on demand estimations, can be updated with the actual demand in the operational planning level. The program aims to couple these flows with one specific vehicle on each arc. To this end, we define $\mu_{ijkp}^{t\bar{t}}$ to be a binary decision variable taking the value of one if the commodity k is coupled with the MV p on arc $((i,t),(j,\bar{t})) \in A^{\mathcal{M}}$, otherwise zero. Similarly, the binary variable $\alpha_{ijkq}^{t\bar{t}}$ equals one if the commodity k is coupled with the AV q on arc $((i,t),(j,\bar{t})) \in A^{\mathcal{A}}$, otherwise zero.

As transshipping commodities between vehicles may increase operational efforts, we aim to minimize the transshipment effort, which we measure based on the number of vehicles a commodity is loaded on. Therefore, we define the binary variable γ_{kp} taking the value of one if a commodity k is loaded on an MV p, otherwise zero. Binary variable δ_{kq} equals one if a commodity k is loaded on an AV q, otherwise zero.

The **post-processing program** we seek to solve is as follows:

$$\min \sum_{k \in K} \left[\sum_{p \in P} \gamma_{kp} + \sum_{q \in Q} \delta_{kq} \right] \tag{13}$$

subject to

$$\mu_{ijkp}^{t\bar{t}} \leq \gamma_{kp}, \qquad \forall \left((i,t),\left(j,\bar{t}\right)\right) \in A^{\mathcal{M}} \setminus A_h, k \in K, p \in P \tag{14}$$

$$\alpha_{ijkq}^{t\bar{t}} \le \delta_{kq}, \qquad \forall\big((i,t),(j,\bar{t})\big) \in A^{\mathcal{A}} \setminus A_h, k \in K, q \in Q \tag{15}$$

$$\hat{x}_{ij}^{kt\bar{t}} = \sum_{p\in P} \mu_{ijkp}^{t\bar{t}} + \sum_{q\in Q} \alpha_{ijkq}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{M}} \cup A^{\mathcal{A}} \setminus A_h, k \in K \tag{16}$$

$$\sum_{k\in K} v^k \mu_{ijkp}^{t\bar{t}} \le u_M m_{ijp}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{M}} \setminus A_h^{\mathcal{M}}, p \in P \tag{17}$$

$$\sum_{k\in K} v^k \alpha_{ijkq}^{t\bar{t}} \le u_A a_{ijq}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{A}} \setminus A_h^{\mathcal{A}}, q \in Q \tag{18}$$

$$a_{ijq}^{t\bar{t}} \le \sum_{p\in P} \pi_{ijpq}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A_m^{\mathcal{A}} : (i,j) \in A_M, q \in Q \tag{19}$$

$$\sum_{q\in Q} \pi_{ijpq}^{t\bar{t}} \le n_{ij} m_{ijp}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A_m^{\mathcal{A}} : (i,j) \in A_M, p \in P \tag{20}$$

$$\sum_{p\in P} \pi_{ijpq}^{t\bar{t}} \le 1, \qquad \forall((i,t),(j,\bar{t})) \in A_m^{\mathcal{A}} : (i,j) \in A_M, q \in Q \tag{21}$$

$$\sum_{((j,\bar{t}),(i,t))\in A^{\mathcal{M}}} m_{jip}^{\bar{t}t} = \sum_{((i,t),(j,\bar{t}))\in A^{\mathcal{M}}} m_{ijp}^{t\bar{t}}, \qquad \forall(i,t) \in N : t \ne 0, p \in P \tag{22}$$

$$\sum_{((j,\bar{t}),(i,t))\in A^{\mathcal{A}}} a_{jiq}^{\bar{t}t} = \sum_{((i,t),(j,\bar{t}))\in A^{\mathcal{A}}} a_{ijq}^{t\bar{t}}, \qquad \forall(i,t) \in N : t \ne 0, q \in Q \tag{23}$$

$$\sum_{((i,t),(j,\bar{t}))\in A^{\mathcal{M}}:t=0} m_{ijp}^{t\bar{t}} = 1, \qquad \forall p \in P \tag{24}$$

$$\sum_{((i,t),(j,\bar{t}))\in A^{\mathcal{A}}:t=0} a_{ijq}^{t\bar{t}} = 1, \qquad \forall q \in Q \tag{25}$$

$$\sum_{p\in P} m_{ijp}^{t\bar{t}} = \hat{m}_{ij}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{M}} \tag{26}$$

$$\sum_{q\in Q} a_{ijq}^{t\bar{t}} = \hat{a}_{ij}^{t\bar{t}}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{A}} \tag{27}$$

$$\gamma_{kp} \in \{0,1\}, \qquad \forall k \in K, p \in P \tag{28}$$

$$\delta_{kq} \in \{0,1\}, \qquad \forall k \in K, q \in Q \tag{29}$$

$$\mu_{ijkp}^{t\bar{t}} \in \{0,1\}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{M}}, k \in K, p \in P \tag{30}$$

$$\alpha_{ijkq}^{t\bar{t}} \in \{0,1\}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{A}}, k \in K, q \in Q \tag{31}$$

$$m_{ijp}^{t\bar{t}} \in \{0,1\}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{M}}, p \in P \tag{32}$$

$$a_{ijq}^{t\bar{t}} \in \{0,1\}, \qquad \forall((i,t),(j,\bar{t})) \in A^{\mathcal{A}}, q \in Q \tag{33}$$

$$\pi_{ijpq}^{t\bar{t}} \in \{0,1\}, \qquad \forall((i,t),(j,\bar{t})) \in A_m^{\mathcal{A}} : (i,j) \in A_M, p \in P, q \in Q \tag{34}$$

The objective function (13) minimizes the total number of transshipments by considering the number of vehicles each commodity is assigned to. The number of commodities represents a lower bound for the objective function because each commodity must be shipped by at least one vehicle. Constraints (14) and (15) assign commodities to MVs or AVs, respectively. Constraints (16) ensure that each commodity flow on an arc (except for on holding arcs) is coupled with exactly one MV or AV. Constraints (17) and (18) enforce the capacity restriction for each MV and AV. Constraints (19) ensure that an AV traveling on an MV arc is assigned to a platoon. Constraints (20) are the disaggregated platooning constraints that limit the number of AVs that follow an MV to its platoon capacity. Constraints (21) ensure that each AV can platoon with at most one MV on an MV movement arc. Each vehicle's route needs to ensure the design-balance constraints (22) and (23). Constraints (24) and (25) enforce that each vehicle starts exactly one route in $t = 0$. Constraints (26) and (27) couple the disaggregated binary variables to the aggregated services from the solution to the SNDMAF. The domain of the variables is defined in (28) to (34). Finally, we refer to Appendix A, in which we illustrate the operationalization of an SNDMAF solution by means of an example.

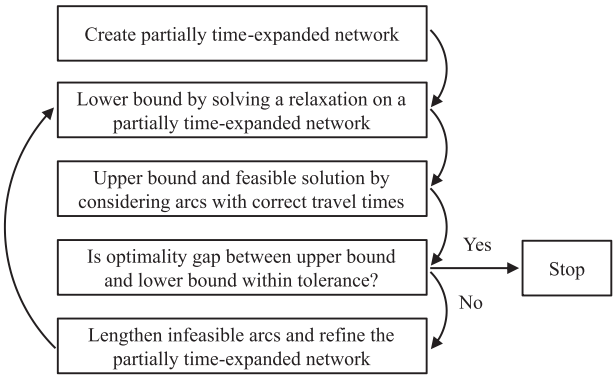


Fig. 3. Dynamic discretization discovery scheme based on Boland et al. (2017).

4. Dynamic discretization discovery

Time-expanded networks are used for SND problems to model time dependencies explicitly within the network. To this end, physical locations are replicated at a finite set of points in time within discrete time intervals over a planning horizon. We call this type of network a fully time-expanded network (D) in the following. The time discretization, i.e., the length of the intervals, is typically a parameter that needs to be decided upon based on problem characteristics. In general, fine discretization leads to large time-expanded networks that enable solutions depicting an accurate representation of time-dependent decisions. Conversely, coarse discretization leads to small networks with a larger tolerance in terms of the accuracy of the solution.

The SNDMAF relies on fine discretization as smaller time intervals enable more opportunities for the synchronization of vehicles by means of platooning. Also, the discretization should at least account for the relatively short travel times in city logistics applications. However, solving large-sized IPs on fully time-expanded networks is computationally expensive even for traditional SND problems (Boland et al., 2018). Solving instances of the SNDMAF showed to be particularly hard because of the numerous opportunities for synchronizing the mixed fleet. Computational studies indicate that the average runtimes of the off-the-shelf solver exceed those required to solve MV-only SND instances by a factor of 8 (Scherr et al., 2019).

We therefore propose a customized solution method based on dynamic discretization discovery (DDD), which was introduced by Boland et al. (2017) to solve the general SND problem. In Section 4.1, we describe the DDD-SNDMAF algorithm in detail, highlighting the modifications required for adapting it to the SNDMAF. To improve the performance of this exact algorithm, we derive two relaxation-based enhancements in Section 4.2. We then evaluate the performance of the three exact variants of the DDD-SNDMAF and Gurobi in the first part of a computational study in Section 4.3. In Section 4.4, we introduce the heuristic search space restriction to all variants and evaluate the heuristic solutions in a second part of the computational study in Section 4.5.

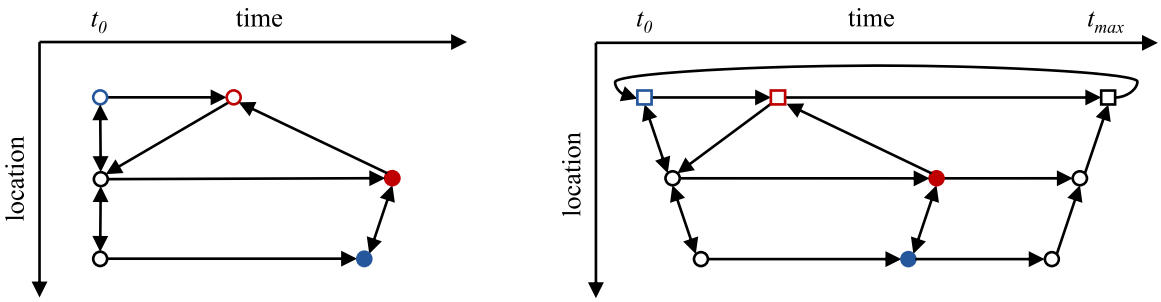
4.1. Adapting DDD to the SNDMAF

The DDD scheme which we outline in Fig. 3 uses the concept of partially time-expanded networks. Such networks are sparse and do not contain all of the nodes and arcs of the fully time-expanded network. Partially time-expanded networks D_T are built in a way that an optimal solution to the service network design problem SND(D_T) solved on this particular network provides a lower bound to the optimal solution of the same problem on a fully time-expanded network. To start with a small number of nodes and arcs, travel times are underestimated such that the network allows for a high degree of consolidation. Thus, the partially time-expanded networks contain arcs of shorter length. As such arcs are not realistic according to the travel time, the obtained solution may be infeasible and needs to be transformed to a feasible solution, yielding an upper bound. The DDD then refines this partially time-expanded network in each iteration by lengthening infeasible arcs and adding nodes. It terminates with an exact solution when the lower bound solution can be transferred to a feasible solution of equal solution value.

In the following, we describe the algorithm and its components in detail. We particularly state the additions and modifications we introduce in our variant of the algorithm, the DDD-SNDMAF, compared to the basic version by Boland et al. (2017). In Section 4.1.1, we introduce a set of properties to define partially time-expanded networks. In Section 4.1.2, we outline the general structure of the DDD algorithm before describing its components in more detail in the subsequent Sections 4.1.3 to 4.1.7.

4.1.1. Properties of the partially time-expanded networks

The partially time-expanded networks D_T need to respect a particular set of properties to provide a valid lower bound to the problem. In this section, we first introduce four properties to address the conventional SND problem. These properties



(a) SND (Properties 1–4) (b) SNDMAF (Properties 1–6)

Fig. 4. Partially time-expanded networks for the different problems (compliant with properties).

are formally defined and proven by Boland et al. (2017), so this paper does not go into further details about them. Then, we introduce two additional properties, since the SNDMAF seeks cyclic vehicle routes over a limited planning horizon, whereas the SND problem of Boland et al. (2017) focuses on routes from commodities' origins to their destinations without respecting asset management.

- Property 1 ensures that D_T contains both the earliest origin node of each commodity as well as the latest destination node.
- Property 2 requires all arcs in D_T to either have the correct or a shorter length in accordance to their physical travel time. Since shorter arcs enable at least as much consolidation opportunities as the later feasible solution, we always underestimate travel times in D_T.
- Property 3 ensures that, for each node in D_T and each physical arc that is connected to the corresponding physical location, there is at least one arc in D_T as well.
- Property 4 as the "longest-feasible arc property" defines that there is no feasible arc in D_T that is longer than a given arc connecting two nodes. In general, the arc that originates from a node is the longest of all feasible arcs which are at most as long as the correct travel time.

We introduce Properties 5 and 6 to account for the problem characteristic that vehicles depart from any external zone at the beginning and return to any external zone at the end of the planning horizon. Therefore, we define feasible paths that respect the correct travel time of each arc.

- Property 5 states that there is at least one feasible path from one of the nodes for external zones at the beginning of the planning horizon (t_0) towards any node in D_T. Respectively, there needs to be at least one feasible path from any node to one of the nodes representing external zones at the end of the planning horizon ($t_{\max}$). Due to this requirement, the earliest feasible time point $t^i_{\min}$ and the latest feasible time point $t^i_{\max}$ of each physical location $i \in N_{ph}$ need to be represented in D_T.
- Property 6 as the "latest-feasible arc property" ensures that every arc in D_T can be used in a feasible solution to SNDMAF once it is lengthened to its correct length. Since vehicles must return to an external zone before the planning horizon ends, D_T must not include arcs that would exceed the latest time point $t^j_{\max}$ of the destination location given the correct travel time. These latest feasible nodes are defined by Property 5.

Fig. 4 illustrates two partially time-expanded networks for the same set of commodities that need to be transported. Colored circles mark the origin nodes, while dots in the same color mark the destination nodes of the two commodities. Fig. 4a on the left displays an initial partially time-expanded network for the conventional SND problem as it is used by Boland et al. (2017). Since this partially time-expanded network only considers Properties 1 to 4, only the commodities' origin and destination nodes as well as a replication of each location in t_0 are included.

Fig. 4b on the right shows a partially time-expanded network for the SNDMAF that additionally considers Properties 5 and 6. Property 5 ensures that the partially time-expanded network is limited by t_0 and $t_{\max}$ according to the length of the planning horizon. The square nodes in the uppermost row represent replications of an external zone. Therefore, we include the earliest nodes replicating a satellite at a time point such that these nodes can be reached from this external zone. Analogously, the latest time points are determined such that there exists a path to the latest replication of the external zone in $t_{\max}$. According to Property 6, the arcs connecting the last nodes of each location either have their correct length or are not included. The partially time-expanded network for the SNDMAF additionally includes auxiliary arcs that "wrap around" connecting the last with the first node of the external zones.

4.1.2. DDD-SNDMAF algorithm

In Algorithm 1, we outline the DDD-SNDMAF. The algorithm runs until the gap between the lower bound $z(D_T)$ and the upper bound $z(D^\tau_T)$ is within tolerance ϵ (Line 2). In iteration 0, we initiate the partially time-expanded network according

Algorithm 1 DDD-SNDMAF.

1: iteration = 0
2: **while** $(z(D_T^\tau) - z(D_T))/z(D_T^\tau) > \epsilon$ **do**
3: **if** iteration = 0 **then**
4: Create-Initial
5: **else**
6: Identify-Nodes N_{add}
7: **for all** $(i, t_{new}) \in N_{add}$ **do**
8: Refine-and-Restore (i, t_{new})
9: **end for**
10: **end if**
11: Solve SNDMAF(D_T) for lower bound $z(D_T)$
12: Correct D_T to D_T^τ
13: Solve SNDMAF(D_T^τ) for upper bound $z(D_T^\tau)$
14: iteration += 1
15: **end while**

to the properties we previously defined (Lines 3–4). Then, the SNDMAF is solved in every iteration on the current partially time-expanded network D_T to obtain a lower bound (Line 11). Boland et al. (2017) apply a linear program (LP) to obtain a feasible solution and an upper bound. We instead correct D_T to yield D_T^τ which only contains arcs with correct, i.e., realistic, travel times (Line 12). On this corrected network, we solve another instance of SNDMAF to obtain a feasible solution and an upper bound (Line 13).

In subsequent iterations, arcs that were used in the incumbent lower bound solution are identified and lengthened such that nodes are added to D_T (Lines 6–9). The DDD-SNDMAF iteratively refines the network and solves versions of the SNDMAF model until it terminates with a proven exact solution to SNDMAF. The individual steps of the DDD-SNDMAF are described in more detail in the following sections.

4.1.3. Creating the initial partially time-expanded network

The first step of the DDD-SNDMAF is creating the initial partially time-expanded network. As described in Algorithm 2,

Algorithm 2 Create-Initial.

Require: Directed network $D_{ph} = (N_{ph}, A_{ph})$, commodity set K
for all $k \in K$ **do**
Add nodes (i_e^k, t_e^k) and (i_s^k, t_s^k) to N
end for
for all $i \in N_{ph}$ **do**
if $i \in N_E$ **then**
Add nodes $(i, 0)$ and $(i, t_{\max})$ to N
else
Add node (i, t_e) to N with $t_e = \min$(length of shortest path from any $j \in N_E$ to i)
Add node (i, t_l) to N with $t_l = t_{\max} - \min$(length of shortest path from i to any $j \in N_E$)
end if
end for
for all $(i, t) \in N$ **do**
for all $(i, j) \in A_{ph}$ with $t + \tau_{ij} \leq t^j_{max(k)}$ **do**
Find largest t' such that $(j, t') \in N$ and $t' \leq t + \tau_{ij}$
Add arc $((i, t), (j, t'))$ to A_m
end for
Find smallest t' such that $(i, t') \in N$ and $t' > t$
if $i \in N_E$ **then**
Add arc $((i, t), (i, t'))$ to A_h
if $t = t_{\max}$ **then**
Add arc $((i, t), (i, 0))$ to A_γ
end if
else
Add arc $((i, t), (i, t'))$ to A_w
end if
end for

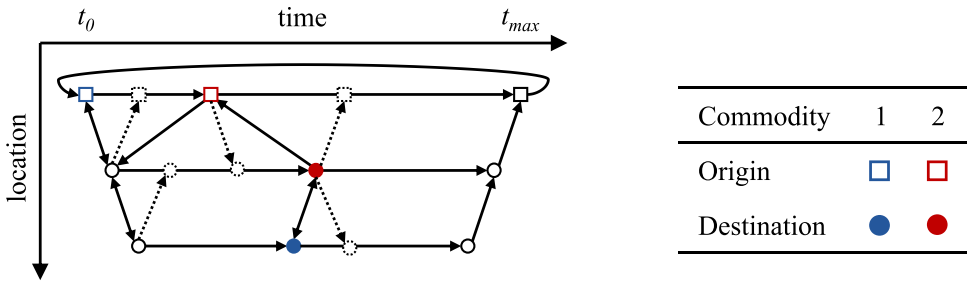


Fig. 5. Example of an initial partially time-expanded network.

we initiate our network in Lines 1–3 with nodes at the time point (t_e^k) each commodity is available at the origin location (i_e^k) and the time point (t_s^k) each commodity may arrive at the destination location (i_s^k). Further, we create nodes for each external zone at the beginning (t_0) and end ($t_{\max}$) of the planning horizon. With the knowledge that vehicles depart or enter these, we place a node for each satellite location at the earliest time point (t_e) they can be reached from any of the external zones departing in t_0 and at the latest time point (t_l) from which any of the external zones in $t_{\max}$ can be reached (Lines 4–11). We determine these time points corresponding to the length of the shortest feasible path.

To connect all nodes, we create movement arcs (A_m) between physical locations, holding (A_h) and waiting arcs (A_w) between subsequent time points of the same location, and auxiliary arcs (A_γ) between the last and first time point of each external zone (Lines 12–26). Movement arcs are only created if the destination time of the arc would not exceed the last time point of its location, given the correct travel time. After performing these steps, we obtain a partially time-expanded network which respects all of the properties defined in Section 4.1.1.

In Fig. 5, we illustrate the initial partially time-expanded network with two commodities to be transported, known from Fig. 4b. Nodes of external zones are depicted as squares and nodes of satellites as circles. For the two commodities, colored nodes mark the origin nodes, while nodes filled in the same color mark the destination nodes. Black nodes mark the earliest and latest feasible time points for each location and therefore define the limits of the planning horizon. Notice that most of the arcs do not have the correct length in this early stage of DDD, some arcs even go backwards in time. With the dotted circles and lines we illustrate how the arcs, and their respective destination nodes, would look with their correct travel time. Considering shorter arcs leads to a smaller network as multiple arcs can share the same node as their respective origin or destination node. Further note that, in this figure, we do not illustrate the additional holding and waiting arcs that would be necessary to connect the dotted nodes.

4.1.4. Obtaining a lower bound by solving an IP with valid inequalities

In this part of the DDD-SNDMAF algorithm, we use an off-the-shelf solver to obtain a solution to SNDMAF(D_T). Although we could solve the model simply as it is formulated in Section 3.2, optimal solutions on partially time-expanded networks underestimate the needed fleet size. The number of vehicles, however, is crucial as they induce fixed costs, which represent the largest share of the total cost in the objective function. As we allow for backward arcs in D_T, vehicles in a solution to SNDMAF(D_T) are able to travel backwards in time. Consequently, services on auxiliary arcs are only installed if transportation capacity for routing commodities is required. As a result, a solution could select services on backward arcs in a way that cyclic routes would be formed that do not cover the complete planning horizon; a behavior that is not possible with conventional time-expanded networks.

The example in Fig. 6 shows a partially time-expanded network that allows for cycles. Although taking shorter arcs is feasible in the SNDMAF(D_T) and the algorithm would converge without valid inequalities, we intend to mitigate the effect

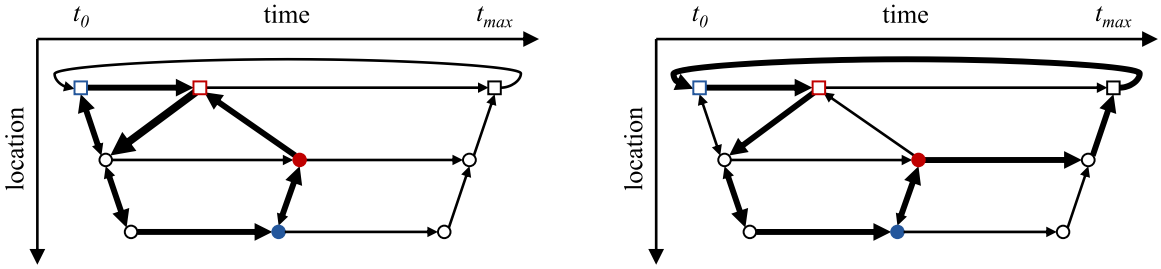


Fig. 6. Effect of the valid inequalities on vehicle routes in a solution to SNDMAF(D_T).

of such cycles. Thus, we propose valid inequalities for the SNDMAF(D_T) to strengthen the lower bound in each iteration in two ways. First, they produce a more accurate estimate of the fleet size and mix. Second, they "unroll" cycles to a certain extent to obtain more realistic vehicle routes. By doing this, fewer arcs are used and lengthened within DDD iterations that are not necessary for the final solution.

Valid inequalities (35) and (36) bound the number of MV and AV services, respectively, on auxiliary arcs that represent the number of vehicles in the fleet. We now aim to enforce a lower bound value on these variables based on an estimate on how many vehicles of a type are necessary given their travel time that is consumed over the limited planning horizon $t_{\max}$. This value is defined on the right-hand side of the inequality by dividing the total travel time consumed by all services of the respective vehicle type by the length of the planning horizon $t_{\max}$. Hereby, we only consider all movement arcs with their correct travel time τ_{ij} as the length of holding and waiting arcs depends on the availability and length of the movement arcs in D_T.

$$\sum_{((i,t),(j,\bar{t}))\in A_\gamma} m_{ij}^{t\bar{t}} \geq \sum_{((i,t),(j,\bar{t}))\in A_m} \frac{\tau_{ij} m_{ij}^{t\bar{t}}}{t_{max}} \tag{35}$$

$$\sum_{((i,t),(j,\bar{t}))\in A_\gamma} a_{ij}^{t\bar{t}} \geq \sum_{((i,t),(j,\bar{t}))\in A_m} \frac{\tau_{ij} a_{ij}^{t\bar{t}}}{t_{max}} \tag{36}$$

With the solutions to SNDMAF(D_T) depicted in Fig. 6, we now intend to illustrate the functionality of these valid inequalities. In each of the two figures, the bold lines represent services of one possible MV route, i.e., the solution value is $m_{ij}^{t\bar{t}} = 1$ on these arcs. Without the valid inequalities in Fig. 6a, the vehicle may take three services on backward arcs after visiting all commodity nodes to form a cyclic route. The backward arc from the red square is even selected twice, such that $m_{ij}^{t\bar{t}} = 2$ on it. The resulting route does not cover the complete planning horizon and does not use the auxiliary arc, which is why no fixed cost is induced for the one MV that would be required.

The cycle highlighted in bold lines in Fig. 6b is a feasible vehicle route we want to enforce with the help of the valid inequalities. Rather than taking the backwards arcs and returning to the external zone in t_0 directly, the vehicle route unrolls to the end of the planning horizon. This is due to the reason that the valid inequalities enforce a positive value for the service variable on the auxiliary arc, as the right-hand side of the inequality is positive. Due to the integrality constraints, the variable on the auxiliary arc obtains a value of $m_{ij}^{t\bar{t}} = 1$, such that the fixed cost for assigning one MV is induced. The design-balance constraints (5) and (6) then ensure an actual route that leads through the auxiliary arcs. Consequently, we obtain a more accurate estimation of the fleet size and a more realistic route.

4.1.5. Identifying additional nodes

We use the same procedure as Boland et al. (2017) to identify nodes that are added to D_T and likely contribute to improving the solution quality in the next iteration. Therefore, we define the set of nodes N_{add} to which we add nodes (i, t). Added nodes permit to lengthen the arcs that were used in SNDMAF(D_T) but had a shorter length than their realistic travel time. Algorithm 3 first checks the solution to SNDMAF(D_T) for such shorter arcs. Therefore, we consider here all movement arcs with a shorter length than their correct travel time. If an arc has a positive service variable value of MVs ($m_{ij}^{t\bar{t}}$) or AVs ($a_{ij}^{t\bar{t}}$) in the last solution to SNDMAF(D_T), we add its realistic destination node to the set of nodes N_{add}. This destination node is comprised of the location j and the time point $t + \tau_{ij}$ according to the respective correct travel time τ_{ij} of the physical arc.

4.1.6. Refining and restoring the network

The identified nodes in set N_{add} are added to D_T in the next step. In Algorithm 4, we describe the procedure of refining the network given the added nodes such that the six properties presented in Section 4.1.1 are restored. First, the new node $(i, t^i_{new}) \in N_{add}$ is added to the network. As it is placed between existing nodes of the same location which are connected by a holding or waiting arc, this arc is replaced with one holding or waiting arc towards the new node and one departing from it (Lines 1–3).

Algorithm 3 Identify-Nodes N_{add}.

Require: Solution to SNDMAF(D_T), node set N, arc set A
1: **for all** $((i,t),(j,\bar{t})) \in A_m$ with $\bar{t} - t < \tau_{ij}$ **do**
2: **if** $m_{ij}^{t\bar{t}} > 0$ or $a_{ij}^{t\bar{t}} > 0$ **then**
3: Add node $(j, t + \tau_{ij})$ to N_{add}
4: **end if**
5: **end for**
6: **return** Set of nodes N_{add}

Algorithm 4 Refine-and-Restore (i, t^i_{new}).

Require: Node $i \in N_{ph}$ and time point $t^i_{new} \in T_i$ with $t^i_k < t^i_{new} < t^i_{k+1}$

1: Add node (i, t^i_{new}) to N
2: Delete arc $((i, t^i_k), (i, t^i_{k+1}))$ from A
3: Add arcs $((i, t^i_k), (i, t^i_{new}))$ and $((i, t^i_{new}), (i, t^i_{k+1}))$ to A
4: **for all** $((i, t^i_k), (j, t)) \in A$ with $t^i_{new} + \tau_{ij} \leq t^j_{max(k)}$ **do**
5: Add arc $((i, t^i_{new}), (j, t))$ to A
6: Set $t' = argmax\{s \in T_j | s \leq t^i_{new} + \tau_{ij}\}$
7: **if** $t' \neq t$ **then**
8: Delete arc $((i, t^i_{new}), (j, t))$ from A
9: Add arc $((i, t^i_{new}), (j, t'))$ to A
10: **end if**
11: **end for**
12: **for all** $((j, t), (i, t^i_k)) \in A$ such that $t + \tau_{ji} \geq t^i_{new}$ **do**
13: Delete arc $((j, t), (i, t^i_k))$ from A
14: Add arc $((j, t), (i, t^i_{new}))$ to A
15: **end for**

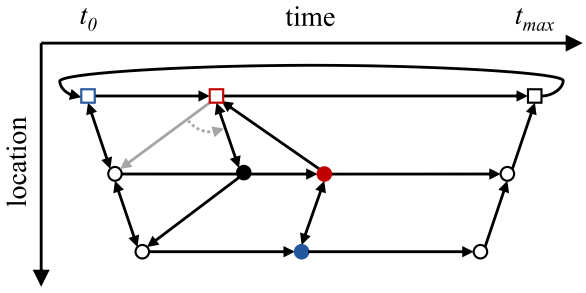


Fig. 7. Adding a node (black dot) as well as refining and restoring the partially time-expanded network.

Further, the new node receives outgoing arcs which mirror those of the immediately earlier node of the same location t^i_k in a way that they arrive at the same node (Line 5). Based on the limited planning horizon in our problem, we make the limitation that arcs are only created if they do not exceed the last time point of the destination location $t^j_{\max(k)}$ given the correct travel time τ_{ij} (see Property 6). As a result, we obtain the desired arcs, however, not all of these arcs may have the longest feasible length, see Property 4.

We begin restoring this property by lengthening the newly created outgoing arcs to arrive at the farthest possible node. Note that this node has to be at most as far from the departure time as the correct travel time (Lines 6–10). Since the arcs that go into the earlier node t^i_k may have a new destination node in t^i_{new}, we check whether this new node can be reached within the correct travel time and, if this is the case, lengthen the respective arcs (Lines 12–15).

In Fig. 7, we show an example of a refined and restored network based on the solution depicted in Fig. 6b. After Algorithm 3 identifies an additional node, Algorithm 4 adds this node (marked as a black dot) as well as refines and restores D_T. The figure shows that the previously used backward arc is lengthened to its correct travel time and shorter arcs originating from the added node are created.

4.1.7. Obtaining a feasible solution

To obtain a feasible solution and, in doing so, an upper bound on the optimal objective function value, we need to consider the actual travel time for all arcs. We therefore correct the partially time-expanded network D_T to a network D^τ_T, that is likely to yield a feasible solution, and solve SNDMAF(D^τ_T) as an IP. The procedure of correcting the network is outlined in Algorithm 5. In principle, we aim to obtain a network that includes all arcs that already have the correct length in D_T and lengthen the short arcs which are used in the current solution to SNDMAF(D_T). Referring back to Fig. 5, which shows an example for a partially time-expanded network, the corresponding corrected network would include the movement arcs and related nodes in dotted lines rather than their shorter versions.

We start building the network D^τ_T by adding all origin and destination nodes of commodities (Lines 1–3) as well as the auxiliary arcs including their respective start and end nodes (Lines 4–7). We then consider all movement arcs in D_T that either have the correct length or are used in the solution to SNDMAF(D_T) of this iteration of the DDD. For each of these arcs, we add the start node (i, t) and the end node $(j, t + \tau_{ij})$ to D^τ_T, considering the correct travel time τ_{ij}, and connect

Algorithm 5 Correct D_T to D_T^τ.

Require: Partially time-expanded network $D_T = (N, A)$, solution to SNDMAF(D_T), commodity set K
1: **for all** $k \in K$ **do**
2: Add nodes (i_e^k, t_e^k) and (i_s^k, t_s^k) to N^τ
3: **end for**
4: **for all** $((i,t),(j,\bar{t})) \in A_\gamma$ **do**
5: Add nodes (i,t) and $(j,\bar{t})$ to N^τ
6: Add arc $((i,t),(j,\bar{t}))$ to A^τ
7: **end for**
8: **for all** $((i,t),(j,\bar{t})) \in A_m$ **do**
9: **if** $\bar{t} = t + \tau_{ij}$ or $m_{ij}^{t\bar{t}} > 0$ or $a_{ij}^{t\bar{t}} > 0$ **then**
10: Add nodes (i,t) and $(j, t+\tau_{ij})$ to N^τ
11: Add arc $((i,t),(j,t+\tau_{ij}))$ to A^τ
12: **end if**
13: **end for**
14: **for all** $(i,t) \in N^\tau$ **do**
15: Add holding/waiting arcs $((i,t),(i,\bar{t}))$ to A^τ
16: **end for**
17: **return** Corrected network $D_T^\tau = (N^\tau, A^\tau)$

them with the respective arc (Lines 8–13). We complete the corrected network by connecting all nodes with holding and waiting arcs (Lines 14–16).

4.2. Relaxation-based enhancements to DDD-SNDMAF

Solving the IPs within iterations of the DDD-SNDMAF can still be a challenging task for an off-the-shelf solver as these programs are hard to solve with increasing instance sizes. The SNDMAF is a special case of the capacitated multicommodity network design problem (CMND) which is an NP-hard problem (Magnanti and Wong, 1984). As each service has a limited capacity, determining the commodities which are routed through an opened service is a problem of combinatorial nature itself. Moreover, integrating the design-balance constraints yields an IP that leads to slow convergence into feasible solutions for the solver as they are deep in the resulting branch-and-bound tree (Pedersen et al., 2009).

To tackle this complexity, we describe in this section two enhanced methods that are based on the previously defined DDD-SNDMAF. Both of these enhancements exploit relaxations of the IP solved in each DDD iteration to obtain a lower bound, the SNDMAF(D_T). Remember that the solution to the SNDMAF(D_T) also determines the arcs that are lengthened in the subsequent DDD iteration. First, we apply a two-phase procedure which makes use of a relaxation in the first phase. Second, we develop a one-phase procedure which solves MIPs in which the domain of variables is gradually refined from continuous to integer in subsequent iterations.

4.2.1. Two-phase DDD-SNDMAF

We apply a two-phase procedure which is similar to the work of Hewitt (2019). In that work, an enhanced DDD method is presented for a load plan design problem. The idea is to leverage the computational advantage of solving a linear program (LP) compared to the integer program (IP). LP solutions consider fractional services and commodity flows for satisfying the design-balance and commodity flow constraints, thus a solver can reach optimal LP solutions in shorter runtimes. The main task of the first phase of two-phase DDD is to build a partially time-expanded network that promises a good starting point for the second phase, without spending too much effort on finding actual optimal solutions in each iteration of the DDD.

In our two-phase DDD-SNDMAF outlined in Algorithm 6, we first execute the DDD-SNDMAF on a version of the SNDMAF in which all of the decision variables are continuous. In contrast to Hewitt (2019), we further relax the problem by ignoring the capacity constraints for transportation (3) in this first phase. The disaggregate coupling constraints defined in (10) and (11) still ensure that commodities are transported by vehicles. Neglecting the capacity of a service allows to route an unlimited number of commodities through it. Thus, an LP solution depicts the largest amount of consolidation possible. Although neglecting the capacity constraints weakens the lower bound even more, it permits to build smaller partially time-expanded networks as more commodities can be routed through the same path. While such a high level of consolidation may not be

Algorithm 6 Two-phase DDD-SNDMAF.

1: Phase 1: Execute linear relaxation of DDD-SNDMAF with $m_{ij}^{t\bar{t}}, a_{ij}^{t\bar{t}}, x_{ij}^{kt\bar{t}}, y^k \in \mathbb{R}$, without capacity constraints (3)
2: Phase 2: Continue with executing DDD-SNDMAF with $m_{ij}^{t\bar{t}}, a_{ij}^{t\bar{t}} \in \mathbb{N}$ and $x_{ij}^{kt\bar{t}}, y^k \in \{0,1\}$ (starting with final partially time-expanded network of Phase 1)

Table 3
Description of evaluated variants of two-phase DDD regarding use of valid inequalities (VI) and consideration of capacity constraints in the first phase (cap).

Name	Description	VI	cap
2-DDD	Two-phase DDD-SNDMAF	✓	
2-DDD w/o VI	Two-phase DDD-SNDMAF without valid inequalities		
2-DDD w/ cap	Two-phase DDD-SNDMAF with capacity constraints in first phase	✓	✓

reached in the IP solution once the capacity constraints are restored, some arcs opened in the LP solution may also be utilized in an IP solution of high quality.

Note that this relaxation in the first phase not only applies to the SNDMAF(D_T) for obtaining the lower bound but also to the SNDMAF(D_T^τ) for obtaining the upper bound to permit for a termination criterion. After the first phase terminates, the final partially time-expanded network we obtain from this DDD-SNDMAF is used as an initial network for the second phase, providing it with a "warm start" in terms of the network. In the second phase, the domains of all variables are defined as integer and binary, respectively, see Section 3.2 for the formulation of the SNDMAF model. With this model, we perform the DDD-SNDMAF algorithm again until it terminates with the exact solution.

We analyze the impact of the different components of two-phase DDD-SNDMAF (2-DDD) on its performance by considering three variants which we summarize in Table 3. First, we consider 2-DDD as defined earlier in this section. Second, we test a variant without the valid inequalities to evaluate their impact (2-DDD w/o VI). Finally, we consider a variant that includes the valid inequalities as well as considers capacity constraints in the first phase (2-DDD w/ cap). With this version, we evaluate the impact of ignoring these capacity constraints in the first phase as this is the case in our version of the two-phase DDD.

4.2.2. Partially-relaxed DDD-SNDMAF

In this enhancement, we use the conventional DDD-SNDMAF algorithm, as outlined in Algorithm 1, and start with solving an IP to obtain a lower bound in iteration 0. In the following DDD iterations, we set service and commodity flow variables to continuous if they have not been used in previous iterations. Over the iterations of the partially-relaxed DDD-SNDMAF, these variables are set to integer (or binary in the case of commodity flow variables) once they have been part of a lower bound solution.

With Algorithm 7 we determine the used variables in the previous solutions to SNDMAF(D_T) and create a partially-relaxed model of SNDMAF(D_T). The index $r \in \{1, 2, \ldots, R\}$ denotes the number of DDD iterations the respective arc is in the network, with R being the current iteration. Each arc is attributed an MV and AV service variable as well as a flow variable for each commodity. In the partially-relaxed model, we relax all variables that did not have a positive solution value in any of the previous iterations of the DDD (Lines 4 and 5). Variables on lengthened arcs therefore are also relaxed until they are used in any subsequent iteration. Further note that we also lengthen arcs that are used by a continuous variable, similar to the first phase of two-phase DDD-SNDMAF. The received model produces a valid lower bound just as the fully relaxed and the integer model would do. In contrast to two-phase DDD-SNDMAF, we do not relax the SNDMAF(D_T^τ) to obtain the upper bound, such that we obtain feasible solutions in any iteration.

Algorithm 7 Create partially-relaxed model of SNDMAF(D_T).

Require: Partially time-expanded network $D_T = (N, A)$, solution to SNDMAF(D_T)
1: **for all** $((i,t),(j,\bar{t})) \in A$ **do**
2: **if** $\exists\ m_{ij}^{t\bar{t}r} > 0$ (or $a_{ij}^{t\bar{t}r} > 0$ or $x_{ij}^{kt\bar{t}r} > 0$) **then**
3: Set domain of variable $m_{ij}^{t\bar{t}R} \in \mathbb{N}$ (integer) (or $a_{ij}^{t\bar{t}R} \in \mathbb{N}$ or $x_{ij}^{kt\bar{t}R} \in \{0,1\}$)
4: **else**
5: Set domain of variable $m_{ij}^{t\bar{t}R} \in \mathbb{R}$ (continuous) (or $a_{ij}^{t\bar{t}R} \in \mathbb{R}$ or $x_{ij}^{kt\bar{t}R} \in \mathbb{R}$)
6: **end if**
7: **end for**
8: **return** Partially-relaxed model SNDMAF(D_T)

4.3. Evaluation of the exact algorithms

In this section, we compare the performance of the different exact algorithms to solve SNDMAF instances. First, we define in Section 4.3.1 the instances which we use for the evaluation of the exact algorithms here as well as of the heuristic algorithms later. We compare the algorithms with the commercial solver Gurobi, which we use to solve the complete SNDMAF model on a fully time-expanded network, and report on their overall performance in Section 4.3.2. In Table 4, we give an overview of the evaluated variants. First, we test the regular DDD-SNDMAF (1-DDD). Then, we evaluate the two-phase DDD-SNDMAF (2-DDD), for which we analyze its components in more detail. Finally, we report the performance of

Table 4
Description of evaluated algorithms.

Name	Description
Gurobi	SNDMAF model solved on fully time-expanded network
1-DDD	DDD-SNDMAF
2-DDD	Two-phase DDD-SNDMAF
R-DDD	Partially-relaxed DDD-SNDMAF

the partially-relaxed DDD-SNDMAF (R-DDD). If not stated otherwise, we apply the valid inequalities to all DDD variants. In Section 4.3.3, we report the obtained lower bound in these algorithms to explain their performance.

4.3.1. Instances

In this section, we describe the instances we consider in the computational study. We create a total of 5 physical networks that are illustrated in Appendix B in Fig. B.15. Four of them comprise 1 external zone each and $|N_S| = \{6, 7, 8, 9\}$ satellites, respectively. One of them comprises 2 external zones and 8 satellites. All arcs in the physical networks are bidirected and both directions are attributed the same weight, i.e., the travel time τ_{ij} in minutes between locations i and j. We design the physical networks such that every arc has a travel time $\tau_{ij} \in \{10, 15, 20, 25\}$ to appropriately depict the city logistics setting. To account for these travel times, we consider fully time-expanded networks with a discretization of 5 min. The length of the planning horizon $t_{\max}$ is 8 h to represent the length of a work shift. Each physical network contains 4 bidirectional AV arcs and features a specific heterogeneous infrastructure which is defined by the location of the AV arcs. We refer to Scherr et al. (2019) for a characterization of different types of heterogeneous infrastructure and a study on their implications.

We define the fixed cost ratio (*FCR*) to depict the different relation in costs between MVs and AVs as the MVs require a human driver. In this study, we consider a fixed cost ratio $FCR = 2.0$, i.e., deploying an MV for a day ($f_M = 200$ cost units) costs twice as much as deploying an AV ($f_A = 100$ cost units). The costs for performing transportation services (g_{ij} and l_{ij}) amount to 0.5 cost units per minute of travel time equally for both vehicle types. To mainly focus on the effect of platooning operations, we assume MVs and AVs do not differentiate each other in terms of driving speed and transportation capacity. The transportation capacity u_M and u_A is limited to 20 volume units of commodities. The number of AVs pulled by one MV in a platoon is limited to the platoon capacity $n_{ij} = 2$. The costs c_{ij}^k for transporting a commodity amount to 0.01 cost units per minute of travel time. Outsourcing the delivery of a commodity to a third-party service provider costs $b^k = 100$ cost units for each volume unit.

For the commodities that the LSP seeks to deliver from the external zones to satellites, we consider the following demand pattern. For each commodity, we randomly choose one origin from the set of external zones with equal probability. Each satellite receives a commodity at 2 to 4 time points over the day following a uniform distribution *Uni*(2, 4, 3) (min, max, mode). Each commodity has a mandatory arrival time at its given destination satellite. The arrival time points are regularly spread over the feasible arrival time window. This time window spans from the earliest feasible time point to the latest feasible time point that are determined by the travel time distance from the external zone in t_0 and to the external zone in $t_{\max}$, respectively. We assume that commodities arrive at the external zone 4 h ahead of their arrival time. Hence, some commodities are already available at the start of the planning horizon, whereas others arrive later during the day. Each commodity $k \in K$ features a commodity volume $v^k \sim Tri(1, 5, 3)$ in discrete units that follows a triangular distribution. We further assume that there is no correlation between the volume of different commodities in one demand instance. We create 10 demand instances for each physical network, yielding 50 test instances in total.

4.3.2. Overall performance

All experiments are conducted on an AMD Ryzen Threadripper 2990WX 32-core processor and coded using Python 3.7. Whenever we solve LPs and (M)IPs in this computational study, we use Gurobi in version 8.1.1 on default settings except for the following parameters. The gap at which the solver terminates is set to 1% (with Gurobi parameter MIPGap). For the lower bound LPs and (M)IPs, we focus on proving optimality with a stronger lower bound (MIPFocus = 2). We further limit the available threads to 16, cut off solutions at the upper bound value of the previous iteration (Cutoff = $z(D_T^\tau)$), and provide the solution of the previous iteration as a warm start. The runtime limit is 1 h for each IP solved within the DDD, and we always extract the best known bound on the objective value to update the incumbent lower bound value $z(D_T)$. For all algorithms, including solving the complete model with Gurobi, the termination criterion is set to $\epsilon = (z(D_T^\tau) - z(D_T))/z(D_T^\tau) = 1\%$ and the total runtime limit is 5 h.

We now report the overall results of the three DDD algorithms compared to Gurobi as average results over the whole set of instances, the individual results are listed in Appendix C. In Table 5, we first list the runtime until termination in seconds, then the number of iterations each DDD variant requires (#Iter.). We further report in Column $|A|$ the number of arcs in the partially time-expanded network D_T. As most of the variables and constraints of SNDMAF relate to arcs, the number of arcs also indicates the resulting model size. We then analyze the quality of the solutions found within the time limit. Column Gap reports the optimality gap defined as $(z_{UB} - z_{LB})/z_{UB}$. Then, we compare the average solution value z with the one by Gurobi, i.e., the solution quality (Sol. qual.) defined as $(z - z_{Gurobi})/z_{Gurobi}$. Finally, we report the percentage

Table 5
Results for the exact algorithms.

Algorithm	Runtime	#Iter.	$\|A\|$	Gap	Sol. qual.	%Opt. 5 h	%Opt. 1 h
Gurobi	8,412 s	–	3204.60	1.47%	–	82.00%	32.00%
1-DDD	5,163 s	14.82	725.86	1.06%	−0.24%	86.00%	62.00%
2-DDD	3,109 s	10.16	964.98	0.88%	−0.36%	90.00%	80.00%
R-DDD	4,768 s	13.66	840.30	1.06%	−0.29%	86.00%	70.00%

Table 6
Results for variants of two-phase DDD with different components.

Algorithm	Phase	Runtime	#Iter.	$\|A\|$	Gap	Sol. qual.	%Opt. 5 h
2-DDD	1+2	3,109 s	10.16	964.98	0.88%	−0.36%	90.00%
	1	90 s	6.13	773.58	–	−6.65%	–
2-DDD w/o VI	1+2	3,705 s	10.08	956.14	0.95%	−0.30%	88.00%
	1	119 s	6.48	831.28	–	−6.37%	–
2-DDD w/ cap	1+2	3,694 s	9.62	976.38	0.92%	−0.35%	90.00%
	1	122 s	5.60	849.46	–	−6.08%	–

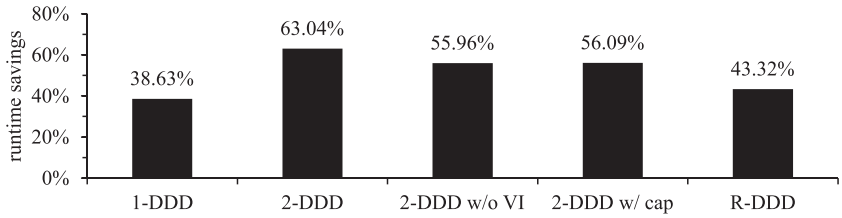


Fig. 8. Runtime savings of DDD algorithms compared to Gurobi.

of instances that were solved to optimality, i.e., terminated within the 1% gap tolerance. We distinguish between instances solved to optimality until the runtime limit of 5 h (%Opt. 5 h) and until 1 h of elapsed runtime (%Opt. 1 h).

The results show that all three DDD variants are able to outperform Gurobi both in terms of computational runtime and solution quality. 1-DDD requires the smallest partially time-expanded networks as IPs are solved to obtain the lower bound. The produced solutions are of slightly better quality than Gurobi's. 2-DDD requires the fewest number of iterations and is the fastest exact algorithm. 2-DDD also provides the best solutions with the smallest reported gap and 90% of the instances solved to optimality. We also note that 2-DDD solves 80% of the instances within 1 h, whereas Gurobi solves only 32% in this time. However, the explored partially time-expanded network is large compared to the other DDD algorithms. R-DDD requires fewer nodes and arcs but also more iterations. The total runtime and solution quality are still slightly improved compared to 1-DDD.

We further compare the average runtime of all algorithms to Gurobi and report the runtime savings in Fig. 8. We observe that all versions of 2-DDD are able to solve the instances in less than half of the time the solver requires. We analyze the impact of 2-DDD's different components in more detail hereafter. 2-DDD also clearly outperforms 1-DDD and R-DDD which both are still over 38% faster than Gurobi.

We report results on the three 2-DDD variants based on the same measures as previously. In Table 6, we further distinguish between the overall results, i.e., Phase 1+2, and results of the first phase only (Phase 1). Note that for Phase 1 we determine the solution quality (Sol. qual.) based on the final solution to the linear relaxation of SNDMAF(D_T^τ), which is not a feasible solution. We observe that adding the valid inequalities reduces the number of iterations until the first phase terminates due to a more focused exploration of the partially time-expanded network. On average, Phase 1 involves only around 3% of the total runtime considering both phases and yet leads to a partially time-expanded network that is useful for the second phase. Afterwards, Phase 2 requires around 4 iterations on average to terminate. Although 2-DDD w/ cap requires fewer iterations to terminate and produces a stronger bound in the first phase, the generated partially time-expanded networks are larger. The reason is that capacity constraints eliminate consolidation opportunities leading to more alternative vehicle paths generated within the DDD.

4.3.3. Quality of the lower bounds

In general, the DDD solution approaches provide a valid lower bound in each iteration. Their quality is different, however, as only 1-DDD solves the full IP to obtain that lower bound, 2-DDD solves a relaxation in the first phase, and R-DDD solves relaxations in every iteration. In Fig. 9, we report the average gap of the lower bound value in each iteration compared to the best found solution value. The gap is defined as $(z^* - z_{LB})/z^*$ and allows us to measure the quality of the lower bounds. We determine the solution value z^* using the best found solution over all algorithms. We show results for 1-DDD, 2-DDD, the variant of 2-DDD without valid inequalities (2-DDD w/o VI) to show their effect, and R-DDD.

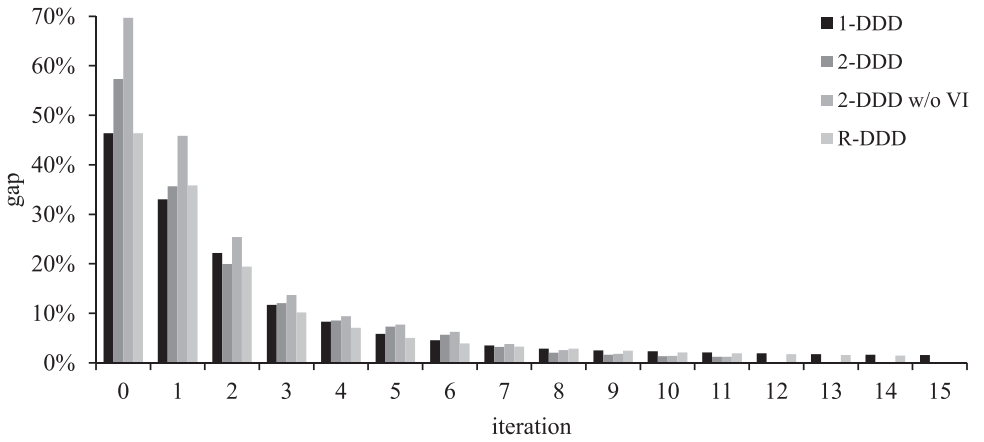


Fig. 9. Gap of lower bound to optimal solution per iteration.

We observe strong convergence of the lower bound from the early iterations on, with all algorithms having a lower bound gap of under 10% after 4 iterations. Although 2-DDD is still using the linear relaxation in this phase of the algorithm, its gap even temporarily undercuts the gap of 1-DDD in iteration 2 due to a further developed, larger partially time-expanded network. We investigate the network size in more detail in Section 4.5.2. R-DDD first follows the path of 1-DDD, as the program in iteration 0 is integer, then it follows the path of 2-DDD, however, with a tighter lower bound gap in the next iterations. In the later iterations, it struggles to close this gap to terminate earlier. In the lower bound values for 2-DDD, we observe slower convergence before the first phase terminates at around iteration 6. In the second phase, the gap closes at a faster rate compared to 1-DDD and R-DDD during the same iterations. Consequently, 2-DDD requires fewer iterations to terminate, and it terminates with a better average solution quality. The results also show that the valid inequalities have a particularly positive effect on the quality of the lower bound within early iterations. This explains the overall better performance of 2-DDD with valid inequalities we noted earlier.

4.4. Heuristic search space restriction

In the previous sections, we have utilized problem knowledge to derive valid inequalities as well as model knowledge to relax SNDMAF(D_T). In this section, we intend to speed up the algorithm based on knowledge that we obtain from within our solution method itself. Although the DDD algorithm may have found feasible solutions of high quality after few iterations, the obtained lower bound may remain relatively weak and the algorithm requires several additional iterations to prove optimality. Strengthening this lower bound either requires a larger share of variables to be integer or a considerably larger network with more arcs of correct length. Both of these measures generally lead to MIPs that are harder to solve.

The iterative approach of DDD-SNDMAF contains historic information, mainly on the solution to SNDMAF(D_T) in previous iterations, which is not known to the solver. This information might be exploited to speed up the solver exploration. With this idea in mind, we generate heuristic cuts on MV and AV service variables based on their values throughout the execution of DDD-SNDMAF. Their first purpose is to reduce the runtimes for solving MIPs, particularly in later iterations, by handing additional information to the solver. Second, exploration of the partially time-expanded networks is restricted as the number of alternative paths reduces due to the cuts. We apply these cuts to the SNDMAF(D_T), however, they not necessarily ensure the lower bound property to the problem, providing us with a heuristic approach instead.

For the current iteration R of the DDD, we use the inequalities (37) and (38) to impose lower bounds on MV and AV service variables, respectively. The lower bound is set to the average number of services on the same arc in previous iterations $\{1, \ldots, R-1\}$. We define the average service value as the sum of a variable's solution value in all iterations divided by the number of iterations the variable exists, i.e., the number of iterations the respective arc is in the partially time-expanded network. As variables always have to be integer on arcs that were previously used, we round down the average service value to the smaller integer. Parameter $r_{\min}$ allows to set a minimum number of iterations an arc must be in the network such that its average service value is considered and the cuts are generated.

$$m_{ij}^{t\bar{t}R} \geq \left\lfloor \frac{\sum_{r=1}^{R-1} m_{ij}^{t\bar{t}r}}{R-1} \right\rfloor, \qquad \forall((i,t),(j,\bar{t})) \in A : R > r_{min} \tag{37}$$

$$a_{ij}^{t\bar{t}R} \geq \left\lfloor \frac{\sum_{r=1}^{R-1} a_{ij}^{t\bar{t}r}}{R-1} \right\rfloor, \qquad \forall((i,t),(j,\bar{t})) \in A : R > r_{min} \tag{38}$$

Table 7
Results for the algorithms after applying heuristic cuts with different values for $r_{\min}$.

Algorithm	$r_{\min}$	Runtime	Sol. found	#Iter.	$\|A\|$	Sol. qual.	$\leq$ Gurobi
2-DDD+cuts	3	1,929 s	1,484 s	8.76	920.96	0.19%	62.00%
1-DDD+cuts	3	529 s	508 s	8.46	558.78	3.81%	10.00%
	4	958 s	786 s	10.54	612.02	1.47%	34.00%
	5	1,582 s	1,425 s	12.44	671.46	0.52%	48.00%
R-DDD+cuts	3	765 s	601 s	8.40	683.98	1.44%	26.00%
	4	1,209 s	1,022 s	9.46	739.70	0.36%	44.00%
	5	2,000 s	1,722 s	11.06	779.66	0.12%	54.00%

4.5. Evaluation of the heuristic algorithms

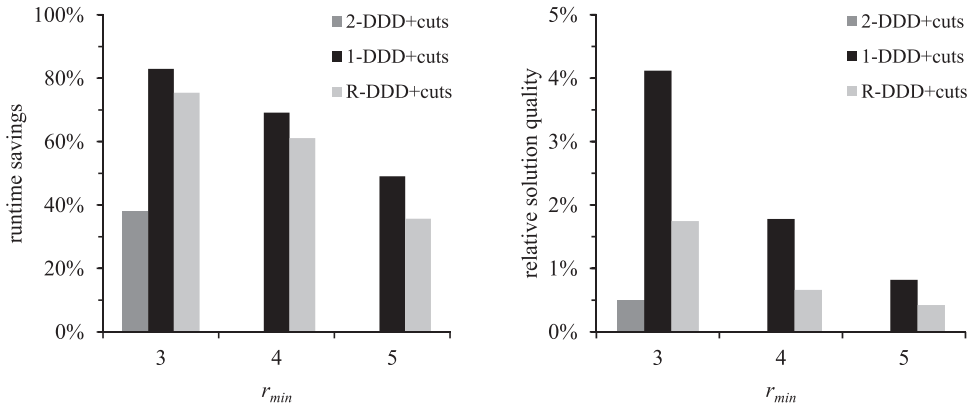
In this section, we add the heuristic search space restriction to all three DDD algorithms we have investigated previously. We evaluate its effect and compare the size of the time-expanded networks.

4.5.1. Effect of the heuristic search space restriction

We start with using $r_{\min} = 3$ as the default setting for this parameter. Later, we vary the parameter for 1-DDD+cuts and R-DDD+cuts to take on values of $r_{\min} = \{4, 5\}$ to analyze its effect on computational performance and solution quality. Since solving the LPs in the first phase of 2-DDD is computationally inexpensive, we only add cuts in the second phase. However, we consider the previous service values of the first phase as input for generating the cuts as well.

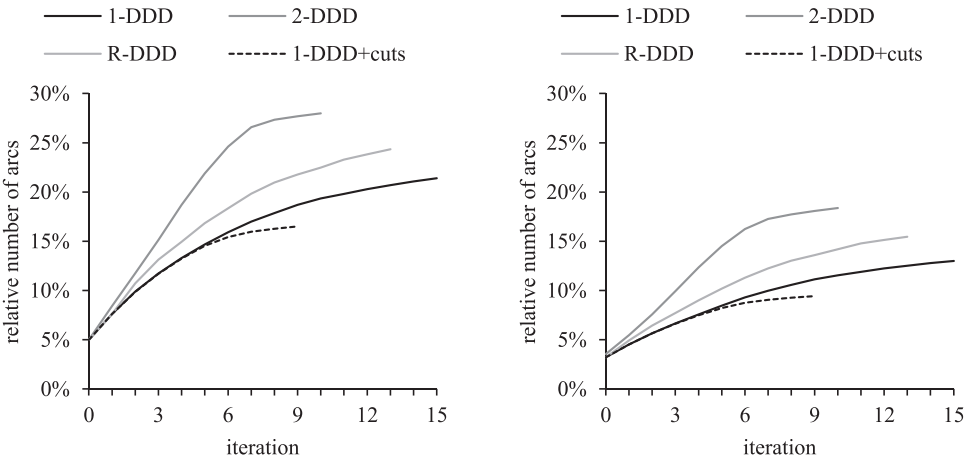
We report results in Table 7 in a similar manner as in previous sections. In addition to the runtime until termination, we also report the elapsed runtime at which the best primal solution is found (Sol. found). As the algorithms are heuristic and do not produce valid lower bounds, we do not report the share of instances solved to proven optimality here. In the last column ($\leq$ Gurobi), we instead list the share of instances for which the found solution was at least as good as Gurobi's or better. The results show that applying the cuts to 2-DDD reduces the total runtime by around 38%. One reason for this is reduced runtimes for solving the lower bound IP due to the cuts. Also, the algorithm terminates after fewer iterations and, thus, the partially time-expanded network at termination is smaller. The obtained solution quality is within 0.5% gap to both Gurobi and 2-DDD, with 62% of the instances solved at least as good as Gurobi. Both 1-DDD+cuts and R-DDD+cuts with the same parameter setting $r_{\min} = 3$ terminate much quicker after a similar number of iterations. However, we note that these algorithms do not explore a large share of the fully time-expanded network and, consequently, the solution quality is worse than that of 2-DDD+cuts and the exact algorithms.

In the following, we vary this parameter for 1-DDD+cuts and R-DDD+cuts to take on values of $r_{\min} = \{3, 4, 5\}$. Fig. 10 compares the performance of the cuts with different parameter values to the fastest exact algorithm, 2-DDD without any cuts. In Fig. 10a, we first show the obtained runtime savings. As expected, we observe that a smaller value for parameter $r_{\min}$ generates stronger cuts after fewer iterations, leading to larger runtime savings. With $r_{\min} = 3$, we can save more than 70% of the runtime of 2-DDD for both tested algorithms. As these cuts are heuristic, we further compare the quality of the obtained solutions against the solutions of 2-DDD. We denote this gap as the relative solution quality and report it in Fig. 10b. With $r_{\min} = 3$, the large savings in runtime for 1-DDD+cuts are associated with solutions that are on average 4% worse. R-DDD+cuts achieves similar runtime savings with a better solution quality that is less than 2% off from 2-DDD's.



(a) Runtime savings (b) Relative solution quality

Fig. 10. Runtime savings and relative solution quality compared to two-phase DDD for different values of $r_{\min}$.



(a) Partially time-expanded network D_T (b) Corrected network D_T^τ

Fig. 11. Relative network size per iteration compared to the fully time-expanded network.

With $r_{\min} = 4$, the relative solution quality of R-DDD+cuts is already under 1% and similar to 2-DDD+cuts, although large runtime savings can still be observed in Fig. 10a. Setting $r_{\min} = 5$ further improves the solution quality and is associated with meaningful runtime savings. In conclusion, the parameter $r_{\min}$ allows to easily adapt the strength of the heuristic cuts with a trade-off between solution quality and computational runtime.

4.5.2. Size of the time-expanded networks

We now investigate the size of the partially time-expanded networks D_T within the different DDD variants. Secondarily, the network size and particularly the number of arcs depicts the resulting model size. We include results for the exact approaches 1-DDD, 2-DDD, and R-DDD, as well as for the heuristic 1-DDD+cuts with $r_{\min} = 3$. With this parameter setting, the smallest network was obtained, see Table 7. In Fig. 11, we report the relative network size, which we define as the number of arcs in relation to the fully time-expanded network of the respective instance. The lines mark the relative network size per iteration. For the sake of simplicity, all lines end when the respective average number of iterations for each algorithm is met. There are instances, however, that require more or fewer iterations to be solved.

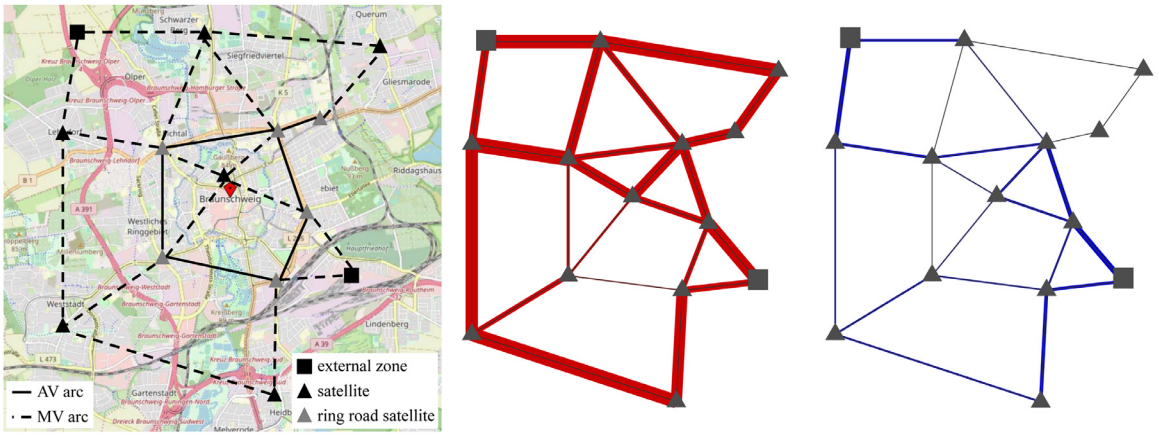
Fig. 11a shows the relative network size of D_T on which we obtain the lower bound. The slope of the curves translates to the added arcs in an iteration. We observe that 2-DDD explores a larger amount of arcs than 1-DDD, and does so particularly within the first phase until around iteration 6. In the second phase, its slope resembles the slope of 1-DDD. As arcs and nodes are never removed from D_T, the relative network size of 2-DDD (30.11%) exceeds 1-DDD's (22.65%) by around a third at termination. Although R-DDD shows a curve similar in shape to 1-DDD's, the size of the network is larger (26.22%). The more extensive exploration of the network is due to the use of continuous variables that provide more flexibility for using arcs, e.g., by splitting services and commodity flows. Applying the cuts to 1-DDD reduces the slope of the curve until the algorithm terminates after fewer iterations with an even smaller partially time-expanded network (17.44%) than 1-DDD.

We additionally depict the relative network size of the corrected network D_T^τ that is used to obtain the upper bound in Fig. 11b. Since this network mainly corrects the length of the arcs in D_T, its size essentially grows in proportion to D_T. These corrected networks, however, are considerably smaller. At termination, they contain between 62% (1-DDD) and 67% (2-DDD) of the arcs of the respective D_T. 1-DDD finds exact solutions with only 14.11% of the arcs in the fully time-expanded network, while 1-DDD+cuts finds heuristic solutions of sufficient quality with only 10.35% of the arcs.

5. Case study

In this case study, we solve instances of the SNDMAF on a network that resembles the street network of Braunschweig, Germany. In the city of Braunschweig, there is a pilot study regarding automated driving titled "Stadtpilot" (Nothdurft et al., 2011). Since 2010, AVs from SAE level 3 upwards regularly drive for testing purposes on the city ring road among regular traffic. Ulmer and Streng (2019) recently propose to use AVs on this ring road for same-day delivery with pickup stations. The goal of our study on this network is to show the ability of DDD-SNDMAF to solve instances of real-world size and to investigate the deployment of a mixed autonomous fleet in an already existing heterogeneous infrastructure.

The physical networks we used in the previous computational experiments are composed of satellites that all receive deliveries. Solutions to the SNDMAF on such networks with three different types of heterogeneous infrastructure are analyzed and discussed extensively in previous work (Scherr et al., 2019). The structure of the street network of Braunschweig and particularly the availability of AV arcs on the ring road, however, determine a special case that was not considered in pre-



(a) Braunschweig network (b) MV services (c) AV services

Fig. 12. Braunschweig network and transportation services.

vious studies. First, the heterogeneous infrastructure combines aspects of the artery type, i.e., AVs can travel through large cohesive parts of the network autonomously, and the corridor type, i.e., AVs need to be pulled between the external zone and the ring road. Second, some satellites do not receive deliveries at all but are rather used as meeting points for platoon merges and splits due to their beneficial location at junctions of the ring road.

We model a physical network with two external zones and 12 satellites. One external zone is located at DHL's delivery base for the city (Persiel, 2017), the other one next to the highway. The network is illustrated in Fig. 12a with the external zones marked as black squares. Half of the satellites (gray triangles) are located along the ring road near junctions allowing for platoon merge and split operations. The other half of the satellites (black triangles) are placed in areas with a high expected demand for orders, such as residential areas or shopping areas. For example, we denote a parking lot near the city center as a satellite. We extract average travel times from Google Maps between locations and round them up to the next time interval with a discretization of 5 min to obtain a conservative estimate on the actual travel time. We denote arcs on the ring road that are feasible for automated driving as AV arcs in both directions (black lines). The rest of the arcs, including the arcs connected to the external zone, are denoted as MV arcs (dashed lines).

Commodities are only delivered to satellites outside of the ring road (black triangles). We generate 10 demand instances for this physical network according to the procedure described in Section 4.3.1, and we apply the same parameters as in the previous experiments. Eventually, LSPs also need to collect commodities of customers from satellites and transport them to external zones. In this case study, we additionally consider this kind of demand which is referred to as outbound demand in the literature (Fontaine et al., 2017). As the majority of goods are typically imported into a city, we generate outbound demand to make up for one third of total demand. Every second inbound commodity destination also equals the origin of an outbound commodity. Outbound commodities need to arrive at the external zone until the end of the planning horizon.

We compare the ability of Gurobi to solve these instances on a fully time-expanded network to two-phase DDD-SNDMAF (2-DDD), the best-performing exact solution method. Based on the results of Section 4.5, we decide to include two heuristic approaches by adding cuts to 2-DDD with $r_{\min} = 3$ (2-DDD+cuts) and by testing DDD-SNDMAF with cuts with $r_{\min} = 3$ (1-DDD+cuts) as the fastest evaluated approach. In Table 8, we report the results for these algorithms based on the measures introduced in the previous section. The results resemble those of the previous test instances. 2-DDD outperforms Gurobi both in computational runtime and in obtained solution quality. The heuristic version of it, 2-DDD+cuts, achieves high-quality solutions in reduced runtime. The gap to optimal solutions is larger for 1-DDD+cuts but the algorithm produces these solutions with considerably smaller partially time-expanded networks and in shorter time.

We now give insights into the structure of the obtained solutions. In Fig. 12, we show the aggregated transportation services over all demand instances on the respective arcs in the network. MV services are depicted in Fig. 12b and AV services in Fig. 12c. The colored width of the arcs represents the frequency, i.e., the number of services on an arc. We

Table 8
Results for solving the Braunschweig instances.

Algorithm	Runtime	#Iter.	$\lvert A\rvert$	Sol. qual.	%Opt.	$\leq$ Gurobi
Gurobi	16,965 s	–	5,696.00	–	10.00%	–
2-DDD	15,377 s	12.20	2,817.00	-3.02%	20.00%	80.00%
2-DDD+cuts	9,910 s	11.20	2,775.30	-3.16%	–	70.00%
1-DDD+cuts	2,587 s	12.70	1,216.50	4.99%	–	30.00%

observe that MVs perform more services as they are able to operate anywhere in the network. This finding is in line with the fleet mix that is composed of 2.1 MVs and 1.5 AVs on average in these results. MVs also heavily use the artery that leads through the city center as they do not rely on the ring road. We further analyze the use of platooning in the depicted services. While MVs use 15.25% of their total travel time to lead platoons, AVs spend 85.64% of their total travel time as platoon followers. We thus notice that, given this heterogeneous infrastructure, the AVs mainly follow MVs in platoons to expand the transportation capacity of MVs. We observe from Fig. 12c that the AV services are rather focused on the parts of the network close to the external zones. Some AV arcs, which only lead towards satellites for which MVs are necessary to perform the deliveries, are frequented rarely.

After applying the post-processing step to the flow-based SNDMAF solutions, we obtain solutions to the operational problem in which commodities are assigned to individual vehicle routes such that the transshipment effort is minimized. The commodity paths determined in the SNDMAF solution allow for an average of 4.71 transshipment opportunities, i.e., the number of intermediary satellites a commodity passes through on its path from origin to destination. The operational solutions show that each commodity is transshipped 0.50 times on average and 58.89% of the commodities are shipped directly without any transshipment between vehicles. The remaining commodities are transshipped 1.23 times on average. Observing the vehicle routes, we notice that AVs switch between different platoons. One AV follows on average 1.67 different MVs over the course of the planning horizon.

6. Conclusions and future work

In this paper, we consider the service network design problem with mixed autonomous fleets (SNDMAF). We formulate a model to determine the fleet size and mix, the routing of vehicles including the synchronization to form platoons, and the routing of commodities. We adapt the DDD scheme to account for design-balance constraints for cyclic vehicle routes and the mixed fleet requirements. In our variant, the DDD-SNDMAF, we consider partially time-expanded networks with two additional properties, formulate valid inequalities, and consider a corrected network on which we solve IPs to obtain an upper bound. We then enhance the DDD-SNDMAF with a two-phase variant, a partially-relaxed variant, and with applying a heuristic search space restriction. Particularly the two-phase DDD-SNDMAF clearly outperforms the commercial solver in solving our test instances to optimality. The heuristic search space restriction provides a valuable addition to DDD-SNDMAF when there is need for much shorter runtimes, however, this is at the expense of provable exact solutions.

In future work, the DDD-SNDMAF variants may be further improved. One direction for research is the removal of nodes and arcs from partially time-expanded networks between DDD iterations if they are no longer required for obtaining high-quality solutions. Further, the proposed solution approaches can be generalized to a range of SND problems with resource management considerations and may be applied to solve related problems. The SNDMAF in particular motivates further studies on producing robust solutions considering uncertainty and on how multiple service providers may cooperate in forming shared platoons. More advanced post-processing techniques for transforming a solution from this tactical planning model to an operational plan may be studied that recognize features such as time needed to transfer AVs between platoons. Further research may also be dedicated to study other modeling approaches such as VRP formulations to depict related problem settings focusing on more detailed operational considerations.

CRediT authorship contribution statement

Yannick Oskar Scherr: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Mike Hewitt:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Bruno Albert Neumann Saavedra:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Dirk Christian Mattfeld:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 227198829 / GRK1931.

Appendix A. Post-processing for operationalizing a tactical plan

In this section, we illustrate the process of operationalizing an SNDMAF solution using an example. Fig. A.13 shows an excerpt of a time-expanded network in which the origin and destination nodes of two commodities, denoted as commodity a and commodity b, are marked in different colors. The SNDMAF determines aggregate MV and AV services as well as commodity paths that minimize total costs. The left part of the figure shows an excerpt of an SNDMAF solution. Each arc is labeled with the number of MV services ($\hat{m}_{ij}^{t\bar{t}}$), the number of AV services ($\hat{a}_{ij}^{t\bar{t}}$), and the commodities k for which there is a commodity flow $\hat{x}_{ij}^{kt\bar{t}} = 1$ on this arc. The solution values of these variables, which we denote as $\hat{m}_{ij}^{t\bar{t}}$, $\hat{a}_{ij}^{t\bar{t}}$, and $\hat{x}_{ij}^{kt\bar{t}}$, are passed on as an input to the post-processing step.

In the operational problem considered in the post-processing step, individual MVs and AVs are assigned to the aggregate services of the SNDMAF solution and commodities are assigned to individual vehicles. In this way, AVs are also assigned

to MVs which lead platoons. The objective of this post-processing step is to operationalize an SNDMAF solution such that transshipment of commodities is minimized. Considering the example on the left, one MV and two AVs are required to perform the services specified in the SNDMAF solution. Deriving a route for the MV in the post-processing step is straightforward as there exists only one feasible path in the SNDMAF solution. The two AVs, however, can be assigned to a total of eight alternative paths, and the two commodities can be transported by any of the three vehicles on their paths.

We illustrate a solution to the operational problem on the right, in which the route of each vehicle is encoded in a different line style. The MV route is depicted as a solid line, the routes of AV 1 and AV 2 are depicted as dashed lines with different margins. If a commodity is assigned to a vehicle on an arc, the line on this arc is colored to match the color of the commodity, i.e., commodity a is assigned to AV 1 (blue) and commodity b is assigned to AV 2 (red). The solution also specifies which AV follows the MV via platooning on each arc. We conclude that this operational solution minimizes transshipment effort as both commodities are shipped directly from their origin to destination without any transshipment necessary at intermediary nodes.

To depict the value of the post-processing step, we illustrate a sub-optimal operational solution in Fig. A.14. The operational solution is based on the same SNDMAF solution and uses the same notation as in the previous figure. Since it is not derived using the post-processing step, however, the transshipment effort is not minimal. With the letter *T*, we depict the transshipment of a commodity at a node that would be required for implementing this operational solution. The color of the letter indicates the respective commodity that is transshipped between vehicles. We observe that the assignment of AVs to different paths than in the previous, optimal solution impacts the assignment of commodities to vehicles, and, ultimately, increases the transshipment effort.

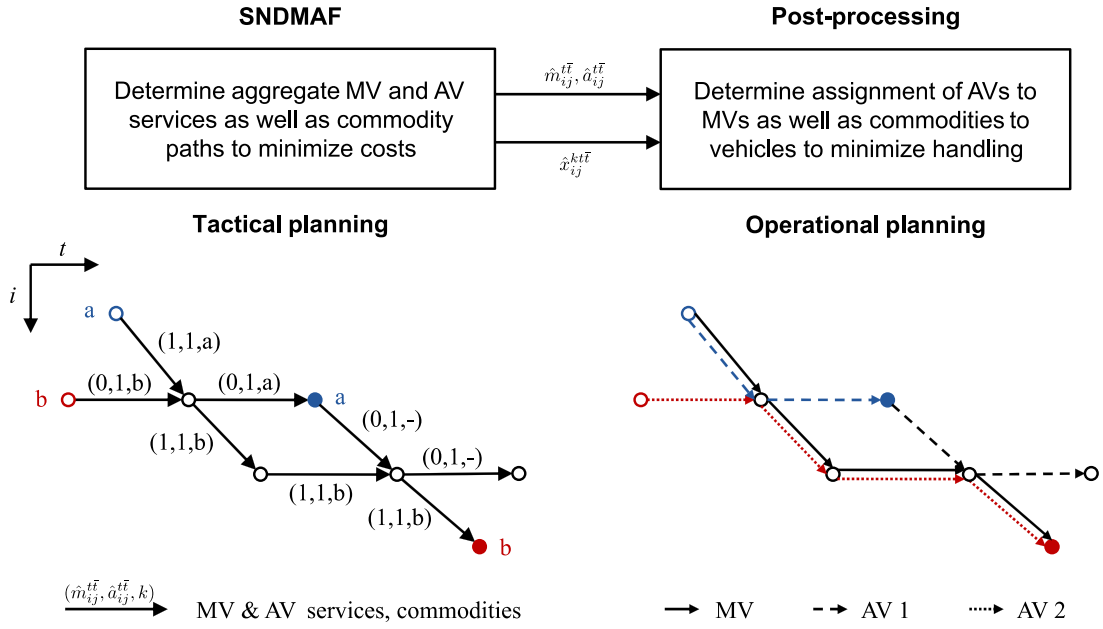


Fig. A.13. Operationalization of an SNDMAF solution.

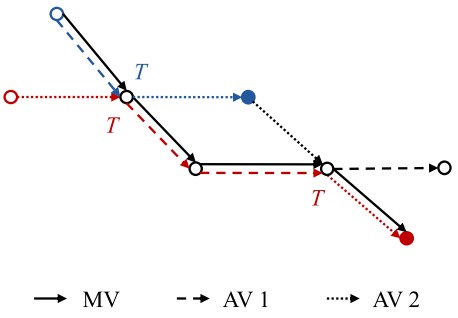


Fig. A.14. Sub-optimal operationalization of an SNDMAF solution.

Appendix B. Physical networks for the computational study

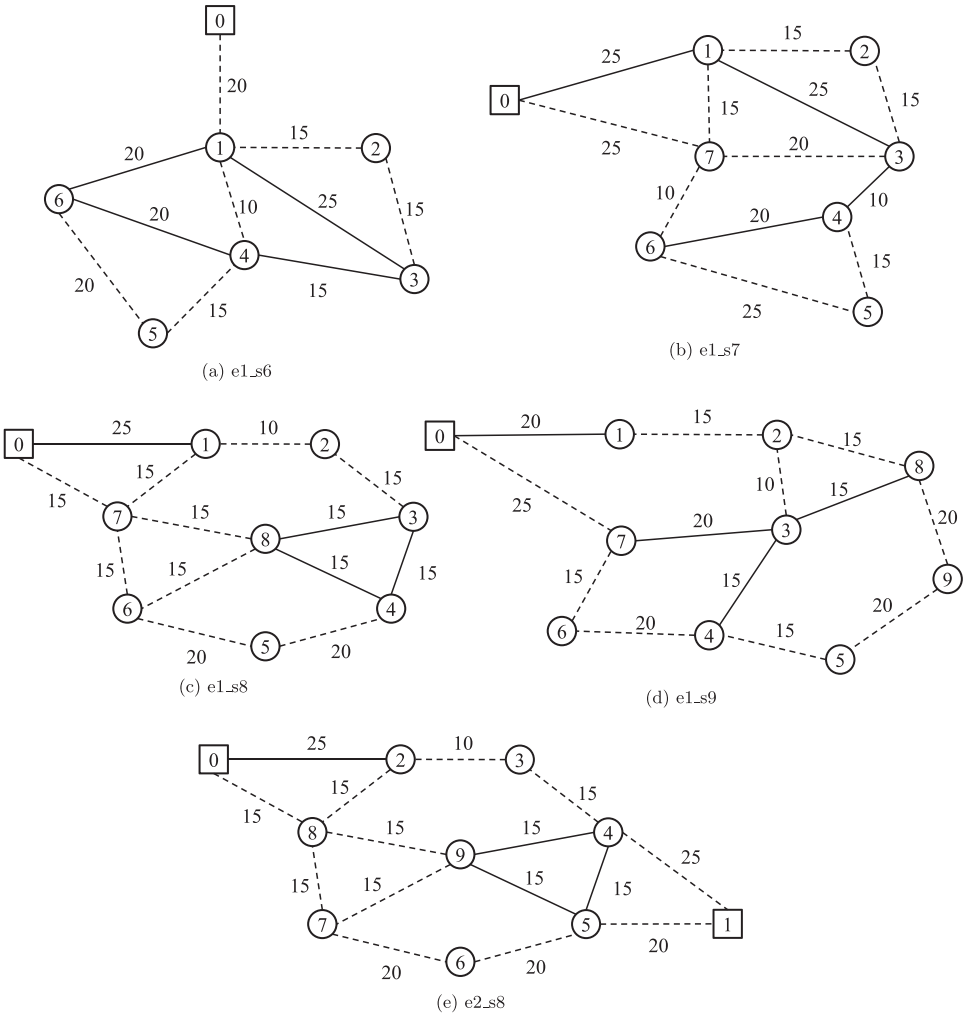


Fig. B.15. Physical networks with external zones (squares), satellites (circles), MV arcs (dashed lines), AV arcs (solid lines), and travel time (arc weights).

Appendix C. Results for all instances of the computational study

This section contains results for each instance composed by the number of external zones (e), number of satellites (s), number of commodities (k), and the demand (d). For each algorithm, we report the runtime in seconds (Runtime), the number of arcs in the partially time-expanded network (|A|), and the objective value of the primal solution (Obj.) (Tables C.9–C.13).

Table C.9
Results for all instances solved with Gurobi, 1-DDD, and R-DDD.

Instance	Gurobi			1-DDD			R-DDD		
	Runtime	|A|	Obj.	Runtime	|A|	Obj.	Runtime	|A|	Obj.
e1_s6_k18_d0	601	2543	7612	34	380	7612	37	368	7612
e1_s6_k18_d1	1983	2543	8570	547	632	8570	972	704	8570
e1_s6_k18_d2	1102	2543	8273	366	500	8273	632	700	8273
e1_s6_k18_d3	185	2543	7522	379	583	7543	915	850	7522
e1_s6_k18_d4	79	2543	7250	132	471	7252	163	469	7251
e1_s6_k18_d5	76	2543	7562	58	413	7562	163	463	7639
e1_s6_k18_d6	1289	2543	7547	1168	655	7547	657	663	7547
e1_s6_k18_d7	414	2543	7497	1252	656	7497	1147	709	7497
e1_s6_k18_d8	7288	2543	7967	673	427	7967	833	542	7967
e1_s6_k18_d9	293	2543	8552	109	409	8552	294	535	8552
e1_s7_k21_d0	1327	3009	8209	337	539	8209	503	630	8215
e1_s7_k21_d1	14,456	3009	9215	5888	939	9215	6685	944	9215
e1_s7_k21_d2	5528	3009	8786	2655	807	8786	1449	846	8797
e1_s7_k21_d3	16,015	3009	9360	2594	710	9360	2693	797	9360
e1_s7_k21_d4	12,817	3009	9387	4142	769	9387	3180	811	9387
e1_s7_k21_d5	10,393	3009	9473	7613	714	9473	3721	736	9473
e1_s7_k21_d6	11,963	3009	9731	18,000	872	9869	18,000	1032	9869
e1_s7_k21_d7	16,458	3009	8233	2244	735	8233	3527	948	8233
e1_s7_k21_d8	568	3009	9080	624	547	9080	180	556	9080
e1_s7_k21_d9	18,000	3009	10,561	18,000	836	10,617	18,000	1062	10,599
e1_s8_k24_d0	12,353	3305	9538	2123	689	9538	2863	835	9538
e1_s8_k24_d1	18,000	3305	10,617	12,093	850	10,559	13,335	1009	10,559
e1_s8_k24_d2	4296	3305	7957	434	534	7957	357	703	7963
e1_s8_k24_d3	18,000	3305	9481	5418	780	9471	7658	964	9471
e1_s8_k24_d4	4238	3305	10,108	1790	712	10,108	638	742	10,108
e1_s8_k24_d5	4520	3305	9612	1372	671	9612	2994	908	9612
e1_s8_k24_d6	7537	3305	9940	18,000	978	9958	18,000	1106	9940
e1_s8_k24_d7	562	3305	9414	3222	631	9439	1553	739	9414
e1_s8_k24_d8	3039	3305	9619	7871	821	9624	3315	837	9619
e1_s8_k24_d9	245	3305	8330	2914	588	8330	673	659	8330
e1_s9_k27_d0	18,000	3581	13,539	18,000	906	13,066	18,000	1137	13,034
e1_s9_k27_d1	11,138	3581	12,961	649	673	12,961	1594	755	12,961
e1_s9_k27_d2	18,000	3581	12,335	5213	749	12,324	3332	924	12,324
e1_s9_k27_d3	18,000	3581	12,693	1953	697	12,190	3481	853	12,190
e1_s9_k27_d4	18,000	3581	14,062	18,000	909	13,632	18,000	1016	13,444
e1_s9_k27_d5	8767	3581	11,336	3306	777	11,336	3787	790	11,336
e1_s9_k27_d6	14,442	3581	12,510	3311	685	12,510	2289	836	12,510
e1_s9_k27_d7	11,335	3581	12,417	6477	838	12,417	6412	869	12,417
e1_s9_k27_d8	15,436	3581	11,497	2293	746	11,497	2487	775	11,497
e1_s9_k27_d9	18,000	3581	11,560	18,000	888	11,560	18,000	1020	11,560
e2_s8_k24_d0	8036	3585	10,037	4156	957	10,037	3456	1020	10,037
e2_s8_k24_d1	10,800	3585	10,951	2823	883	10,951	2693	1181	10,951
e2_s8_k24_d2	12,149	3585	10,847	11,293	1053	10,847	10,827	1281	10,847
e2_s8_k24_d3	2792	3585	10,730	842	781	10,730	1175	985	10,730
e2_s8_k24_d4	6793	3585	10,687	2508	743	10,687	1859	855	10,687
e2_s8_k24_d5	9102	3585	10,146	1642	820	10,146	1524	1119	10,146
e2_s8_k24_d6	1361	3585	9630	7944	818	9630	2135	974	9630
e2_s8_k24_d7	18,000	3585	11,180	18,000	1078	11,192	18,000	1192	11,192
e2_s8_k24_d8	6658	3585	9755	9210	899	9756	3698	946	9755
e2_s8_k24_d9	171	3585	9378	473	545	9397	538	620	9378

Table C.10
Results for all instances solved with 2-DDD, 2-DDD w/o VI, and 2-DDD w/ cap.

Instance	2-DDD			2-DDD w/o VI			2-DDD w/ cap		
	Runtime	$\|A\|$	Obj.	Runtime	$\|A\|$	Obj.	Runtime	$\|A\|$	Obj.
e1_s6_k18_d0	45	597	7612	58	645	7612	131	647	7612
e1_s6_k18_d1	866	730	8570	800	745	8570	1069	689	8570
e1_s6_k18_d2	174	634	8273	369	625	8273	386	690	8273
e1_s6_k18_d3	266	756	7522	195	723	7522	556	804	7522
e1_s6_k18_d4	48	508	7245	62	542	7245	676	534	7250
e1_s6_k18_d5	63	462	7594	72	480	7562	30	454	7562
e1_s6_k18_d6	624	742	7547	462	748	7547	628	724	7547
e1_s6_k18_d7	474	648	7497	580	682	7497	1086	698	7497
e1_s6_k18_d8	231	499	7967	156	535	7967	96	549	7967
e1_s6_k18_d9	173	589	8552	123	626	8552	229	587	8552
e1_s7_k21_d0	166	668	8209	282	631	8209	132	714	8209
e1_s7_k21_d1	2637	1046	9215	5580	1045	9215	3080	1119	9215
e1_s7_k21_d2	1081	876	8786	1600	813	8797	1399	891	8797
e1_s7_k21_d3	665	805	9360	818	871	9360	1365	886	9360
e1_s7_k21_d4	748	955	9387	898	944	9387	1056	953	9387
e1_s7_k21_d5	1727	940	9473	4987	948	9473	3002	806	9473
e1_s7_k21_d6	4983	1086	9731	10,754	1076	9731	4853	1092	9731
e1_s7_k21_d7	587	921	8233	1334	933	8233	1364	891	8233
e1_s7_k21_d8	166	961	9080	203	901	9080	254	1020	9080
e1_s7_k21_d9	8085	1227	10,418	18,000	1234	10,537	12,400	1043	10,418
e1_s8_k24_d0	1914	1065	9538	560	977	9538	2044	1054	9538
e1_s8_k24_d1	8414	1127	10,559	9679	1155	10,559	7781	1102	10,559
e1_s8_k24_d2	143	756	7957	131	760	7957	138	715	7963
e1_s8_k24_d3	1506	1056	9471	3879	953	9471	3132	1036	9471
e1_s8_k24_d4	1246	1040	10,108	533	923	10,108	2605	1001	10,108
e1_s8_k24_d5	850	821	9612	653	920	9612	1233	864	9612
e1_s8_k24_d6	18,000	1161	9940	18,000	1239	9940	18,000	1189	9940
e1_s8_k24_d7	505	906	9414	495	856	9414	462	877	9414
e1_s8_k24_d8	735	912	9619	1426	1004	9619	2531	939	9619
e1_s8_k24_d9	193	972	8330	154	931	8330	226	1061	8331
e1_s9_k27_d0	18,000	1153	13,034	18,000	1159	13,063	18,000	1121	13,041
e1_s9_k27_d1	317	757	12,961	367	906	12,961	344	799	12,961
e1_s9_k27_d2	4371	1026	12,324	2654	953	12,406	3471	1039	12,406
e1_s9_k27_d3	1314	1098	12,190	3549	1051	12,301	2249	1015	12,190
e1_s9_k27_d4	18,000	1221	13,444	18,000	1112	13,455	18,000	1165	13,455
e1_s9_k27_d5	392	833	11,336	783	903	11,336	623	961	11,336
e1_s9_k27_d6	2596	998	12,510	2245	980	12,510	3785	1033	12,510
e1_s9_k27_d7	760	1057	12,417	948	1028	12,419	3558	1260	12,417
e1_s9_k27_d8	495	1024	11,497	1346	1011	11,497	2250	1014	11,497
e1_s9_k27_d9	18,000	1333	11,560	18,000	1322	11,507	18,000	1243	11,513
e2_s8_k24_d0	1590	1174	10,037	2531	1186	10,063	2636	1198	10,037
e2_s8_k24_d1	1084	1366	10,951	1117	1270	10,951	2165	1414	10,951
e2_s8_k24_d2	6508	1264	10,847	1565	1157	10,847	4795	1335	10,847
e2_s8_k24_d3	650	1286	10,730	568	1245	10,730	1398	1321	10,730
e2_s8_k24_d4	1244	1332	10,687	1025	1268	10,687	1880	1193	10,687
e2_s8_k24_d5	382	1042	10,146	292	977	10,146	548	1100	10,146
e2_s8_k24_d6	651	1156	9630	902	1207	9630	551	1122	9630
e2_s8_k24_d7	18,000	1403	11,192	18,000	1399	11,192	18,000	1440	11,192
e2_s8_k24_d8	3587	1191	9755	10,087	1183	9755	10,149	1246	9755
e2_s8_k24_d9	219	1069	9397	410	1025	9397	355	1171	9397

Table C.11
Results for all instances solved with 1-DDD+cuts with parameter $r_{\min} = \{3, 4, 5\}$.

Instance	1-DDD+cuts ($r_{\min} = 3$)			1-DDD+cuts ($r_{\min} = 4$)			1-DDD+cuts ($r_{\min} = 5$)		
	Runtime	$\|A\|$	Obj.	Runtime	$\|A\|$	Obj.	Runtime	$\|A\|$	Obj.
e1_s6_k18_d0	23	329	8794	37	357	7612	30	347	7612
e1_s6_k18_d1	20	349	8603	70	457	8570	132	505	8570
e1_s6_k18_d2	52	385	8689	95	411	8684	226	496	8585
e1_s6_k18_d3	82	363	7566	184	457	7522	140	431	7522
e1_s6_k18_d4	88	420	7668	95	450	7625	93	453	7517
e1_s6_k18_d5	46	371	8415	82	434	7938	77	435	7682
e1_s6_k18_d6	163	465	7573	274	583	7547	518	586	7547
e1_s6_k18_d7	95	351	7532	76	376	7544	231	491	7497
e1_s6_k18_d8	60	344	8613	53	356	8623	172	405	8613
e1_s6_k18_d9	36	354	8614	34	357	8614	118	401	8614
e1_s7_k21_d0	202	486	8209	240	518	8209	337	545	8215
e1_s7_k21_d1	361	574	9562	1060	670	9430	1785	732	9215
e1_s7_k21_d2	607	696	8900	729	688	8900	1315	788	8797
e1_s7_k21_d3	222	495	9987	930	633	9360	843	641	9360
e1_s7_k21_d4	2237	484	10,523	1987	549	9949	2991	673	9695
e1_s7_k21_d5	271	536	9704	332	569	9601	1620	619	9554
e1_s7_k21_d6	846	775	9991	3484	887	9869	1913	825	9991
e1_s7_k21_d7	414	537	8913	312	547	8235	619	627	8599
e1_s7_k21_d8	479	452	9325	644	550	9080	597	521	9080
e1_s7_k21_d9	849	732	10,688	4828	886	10,537	7808	951	10,537
e1_s8_k24_d0	341	520	10,368	419	546	10,368	898	652	9565
e1_s8_k24_d1	786	629	11,011	1446	612	10,873	2213	673	10,755
e1_s8_k24_d2	361	535	8803	238	511	7971	435	536	7957
e1_s8_k24_d3	411	539	9572	857	639	9634	1434	680	9497
e1_s8_k24_d4	626	719	10,183	260	548	10,678	877	717	10,108
e1_s8_k24_d5	136	522	9725	540	605	9612	676	675	9612
e1_s8_k24_d6	101	513	10,092	1363	708	10,092	1270	766	10,092
e1_s8_k24_d7	618	531	10,222	721	595	9540	746	547	9414
e1_s8_k24_d8	137	582	10,073	773	580	10,300	2173	748	9624
e1_s8_k24_d9	1064	543	8330	2021	555	8334	1601	563	8330
e1_s9_k27_d0	438	591	14,138	1428	680	13,415	1558	722	13,269
e1_s9_k27_d1	286	587	13,658	456	595	13,527	490	630	13,147
e1_s9_k27_d2	752	621	12,494	602	631	12,523	1028	617	12,458
e1_s9_k27_d3	871	630	12,862	826	661	12,233	1190	652	12,190
e1_s9_k27_d4	1270	718	13,461	2641	794	13,632	6230	878	13,502
e1_s9_k27_d5	1082	582	11,622	1167	615	11,345	1749	664	11,336
e1_s9_k27_d6	566	635	12,510	523	625	12,510	627	649	12,510
e1_s9_k27_d7	313	606	13,233	2710	787	12,465	3580	830	12,465
e1_s9_k27_d8	819	676	11,886	906	714	11,537	1067	706	11,543
e1_s9_k27_d9	736	499	11,560	447	521	11,560	2260	854	11,560
e2_s8_k24_d0	464	659	10,632	822	819	10,531	2454	911	10,180
e2_s8_k24_d1	186	602	12,040	344	699	11,832	2001	986	11,157
e2_s8_k24_d2	154	596	11,777	1037	802	10,881	2423	891	11,101
e2_s8_k24_d3	248	748	10,811	369	745	10,730	294	702	10,730
e2_s8_k24_d4	793	678	11,271	742	680	10,687	815	695	10,687
e2_s8_k24_d5	983	643	10,698	1520	675	10,146	1667	783	10,146
e2_s8_k24_d6	1182	799	9792	2225	745	9722	3820	863	9722
e2_s8_k24_d7	531	777	11,316	999	808	11,232	5103	1046	11,192
e2_s8_k24_d8	2524	573	10,560	3469	802	9771	6300	878	9770
e2_s8_k24_d9	493	588	9462	505	569	9397	563	587	9378

Table C.12
Results for all instances solved with R-DDD+cuts with parameter $r_{\min} = \{3, 4, 5\}$.

Instance	R-DDD+cuts ($r_{\min} = 3$)			R-DDD+cuts ($r_{\min} = 4$)			R-DDD+cuts ($r_{\min} = 5$)		
	Runtime	$\|A\|$	Obj.	Runtime	$\|A\|$	Obj.	Runtime	$\|A\|$	Obj.
e1_s6_k18_d0	42	387	7915	22	394	7612	23	390	7612
e1_s6_k18_d1	36	428	8570	87	494	8570	200	611	8570
e1_s6_k18_d2	40	412	8684	109	561	8585	111	539	8585
e1_s6_k18_d3	74	421	7566	48	424	7522	53	437	7522
e1_s6_k18_d4	37	371	7761	71	452	7558	76	524	7602
e1_s6_k18_d5	50	358	8415	44	421	7656	165	536	7849
e1_s6_k18_d6	73	420	7573	250	679	7547	327	619	7547
e1_s6_k18_d7	71	387	7523	41	376	7532	555	746	7497
e1_s6_k18_d8	81	412	8190	54	411	8187	180	498	8133
e1_s6_k18_d9	75	417	8655	89	448	8655	125	531	8552
e1_s7_k21_d0	339	527	8209	433	556	8209	450	575	8209
e1_s7_k21_d1	447	676	9415	4142	855	9215	6212	948	9215
e1_s7_k21_d2	269	754	8990	973	959	9010	979	726	8797
e1_s7_k21_d3	939	719	9360	1252	711	9360	2823	781	9360
e1_s7_k21_d4	2147	836	9505	3303	770	9457	4921	918	9597
e1_s7_k21_d5	280	709	9554	867	717	9554	2395	854	9535
e1_s7_k21_d6	1209	850	9869	2727	961	9731	1689	804	9731
e1_s7_k21_d7	250	575	8915	1402	759	8248	1909	730	8233
e1_s7_k21_d8	79	460	9176	156	593	9080	228	519	9154
e1_s7_k21_d9	6579	980	10,637	5534	1069	10,599	8218	1134	10,537
e1_s8_k24_d0	477	709	9565	914	839	9538	1479	942	9538
e1_s8_k24_d1	276	595	11,129	2367	705	10,617	4353	724	10,721
e1_s8_k24_d2	117	545	7971	405	656	7957	326	700	7963
e1_s8_k24_d3	1855	784	9600	2442	864	9497	4277	877	9471
e1_s8_k24_d4	264	679	10,286	327	656	10,258	474	698	10,258
e1_s8_k24_d5	527	750	9725	847	750	9612	1852	846	9612
e1_s8_k24_d6	200	641	10,092	287	671	10,092	4384	953	9940
e1_s8_k24_d7	274	604	9485	413	694	9513	531	657	9423
e1_s8_k24_d8	531	811	9624	790	786	9624	3105	906	9624
e1_s8_k24_d9	475	622	8330	536	650	8337	802	698	8330
e1_s9_k27_d0	393	702	13,159	3485	974	13,034	795	751	13,086
e1_s9_k27_d1	311	623	13,527	569	710	12,961	600	680	13,016
e1_s9_k27_d2	886	738	12,324	1328	753	12,324	3043	861	12,324
e1_s9_k27_d3	601	669	12,369	1505	862	12,190	1009	727	12,192
e1_s9_k27_d4	3859	896	13,456	1898	745	13,814	3034	846	13,445
e1_s9_k27_d5	921	656	11,336	1321	729	11,336	1129	783	11,336
e1_s9_k27_d6	1612	906	12,852	790	831	12,599	799	813	12,510
e1_s9_k27_d7	1962	837	12,628	3985	903	12,659	14,603	987	12,587
e1_s9_k27_d8	2786	810	11,733	2596	821	11,537	2341	779	11,497
e1_s9_k27_d9	744	722	11,821	859	672	11,560	4220	1061	11,560
e2_s8_k24_d0	368	636	10,581	470	935	10,163	3008	1169	10,063
e2_s8_k24_d1	1024	1118	10,951	2226	1051	11,297	1251	952	11,026
e2_s8_k24_d2	970	1061	10,964	2370	1111	10,925	2113	1038	10,870
e2_s8_k24_d3	463	993	10,730	750	952	10,730	707	902	10,759
e2_s8_k24_d4	590	840	10,793	1264	899	10,793	1702	863	10,687
e2_s8_k24_d5	250	884	10,146	516	908	10,146	745	972	10,146
e2_s8_k24_d6	639	811	10,339	570	758	9635	1095	881	9680
e2_s8_k24_d7	1197	1083	11,192	1236	951	11,232	2767	1034	11,192
e2_s8_k24_d8	380	720	9797	1648	917	9760	1645	833	9756
e2_s8_k24_d9	158	655	9378	143	622	9397	155	630	9397

Table C.13
Results for all instances solved with 2-DDD+cuts with parameter $r_{min} = 3$.

Instance	2-DDD+cuts ($r_{min} = 3$)			Instance	2-DDD+cuts ($r_{min} = 3$)		
	Runtime	$\|A\|$	Obj.		Runtime	$\|A\|$	Obj.
e1_s6_k18_d0	116	596	7612	e1_s9_k27_d0	5376	1075	13,034
e1_s6_k18_d1	157	521	8570	e1_s9_k27_d1	485	804	12,961
e1_s6_k18_d2	178	695	8585	e1_s9_k27_d2	2290	1032	12,324
e1_s6_k18_d3	413	719	7522	e1_s9_k27_d3	397	935	12,369
e1_s6_k18_d4	73	488	7546	e1_s9_k27_d4	3296	1052	13,513
e1_s6_k18_d5	29	400	7674	e1_s9_k27_d5	314	796	11,470
e1_s6_k18_d6	208	638	7547	e1_s9_k27_d6	3755	1028	12,510
e1_s6_k18_d7	68	510	7499	e1_s9_k27_d7	2475	1098	12,417
e1_s6_k18_d8	56	447	8042	e1_s9_k27_d8	602	929	11,529
e1_s6_k18_d9	228	625	8614	e1_s9_k27_d9	4046	1030	11,560
e1_s7_k21_d0	350	667	8308	e2_s8_k24_d0	2889	1184	10,037
e1_s7_k21_d1	2139	1070	9311	e2_s8_k24_d1	1364	1367	10,951
e1_s7_k21_d2	702	815	8797	e2_s8_k24_d2	5704	1260	10,917
e1_s7_k21_d3	556	901	9360	e2_s8_k24_d3	591	1293	10,730
e1_s7_k21_d4	949	868	9443	e2_s8_k24_d4	1230	1337	10,687
e1_s7_k21_d5	1518	797	9473	e2_s8_k24_d5	480	1047	10,146
e1_s7_k21_d6	3276	899	10,125	e2_s8_k24_d6	804	1153	9630
e1_s7_k21_d7	1053	892	8233	e2_s8_k24_d7	18,000	1413	11,192
e1_s7_k21_d8	306	963	9080	e2_s8_k24_d8	6015	1208	9755
e1_s7_k21_d9	13,497	1250	10,418	e2_s8_k24_d9	197	1054	9378
e1_s8_k24_d0	578	787	10,134				
e1_s8_k24_d1	1448	955	10,559				
e1_s8_k24_d2	145	747	7957				
e1_s8_k24_d3	1104	1039	9497				
e1_s8_k24_d4	1006	1046	10,108				
e1_s8_k24_d5	639	823	9679				
e1_s8_k24_d6	3981	982	9940				
e1_s8_k24_d7	412	878	9414				
e1_s8_k24_d8	757	971	9693				
e1_s8_k24_d9	191	964	8330				

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.trb.2020.09.009.

References

ACEA, 2017. European Automobile Manufacturers Association – EU Roadmap for Truck Platooning. https://www.acea.be/uploads/publications/Platooningroadmap.pdf.

Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization approaches for the traveling salesman problem with drone. Transp Sci 52 (4), 965–981. doi:10.1287/trsc.2017.0791.

Allen, J., Browne, M., Woodburn, A., Leonardi, J., 2012. The role of urban consolidation centres in sustainable freight transport. Transp Rev 32 (4), 473–490. doi:10.1080/01441647.2012.688074.

Andersen, J., Crainic, T.G., Christiansen, M., 2009. Service network design with asset management: formulations and comparative analyses. Transp. Res. Part C 17 (2), 197–207. doi:10.1016/j.trc.2008.10.005.

Bhoopalam, A.K., Agatz, N., Zuidwijk, R., 2018. Planning of truck platoons: a literature review and directions for future research. Transp. Res. Part B 107, 212–228. doi:10.1016/j.trb.2017.10.016.

Boland, N., Hewitt, M., Marshall, L., Savelsbergh, M., 2017. The continuous-time service network design problem. Oper. Res. 65 (5), 1303–1321. doi:10.1287/opre.2017.1624.

Boland, N., Hewitt, M., Marshall, L., Savelsbergh, M., 2018. The price of discretizing time: a study in service network design. EURO J. Transp. Logist. doi:10.1007/s13676-018-0119-x.

Boysen, N., Briskorn, D., Schwerdfeger, S., 2018. The identical-path truck platooning problem. Transp. Res. Part B 109, 26–39. doi:10.1016/j.trb.2018.01.006.

Boysen, N., Schwerdfeger, S., Weidinger, F., 2018. Scheduling last-mile deliveries with truck-based autonomous robots. Eur. J. Oper. Res. 271 (3), 1085–1099. doi:10.1016/j.ejor.2018.05.058.

CB Insights, 2019. 40+ Corporations Working On Autonomous Vehicles. https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/.

Coker, A., 2018. Lyft looks to create human, autonomous hybrid network in ride-sharing space – FreightWaves. https://www.freightwaves.com/news/lyft-looks-to-create-human-autonomous-hybrid-network-in-ridesharing-space.

Crainic, T.G., 2000. Service network design in freight transportation. European Journal of Operational Research 122 (2), 272–288. doi:10.1016/S0377-2217(99)00233-7. https://www.sciencedirect.com/science/article/pii/S0377221799002337.

Crainic, T.G., 2008. City Logistics. In: State-of-the-Art Decision-Making Tools in the Information-Intensive Age. INFORMS, pp. 181–212. doi:10.1287/educ.1080.0047. http://pubsonline.informs.org/doi/abs/10.1287/educ.1080.0047.

Crainic, T.G., Hewitt, M., Toulouse, M., Vu, D.M., 2016. Service Network Design with Resource Constraints. Transportation Science 50 (4), 1380–1393. doi:10.1287/trsc.2014.0525. http://pubsonline.informs.org/doi/10.1287/trsc.2014.0525.

Crainic, T.G., Hewitt, M., Toulouse, M., Vu, D.M., 2017. Scheduled service network design with resource acquisition and management. EURO Journal on Transportation and Logistics 7 (3), 277–309. doi:10.1007/s13676-017-0103-x. http://link.springer.com/10.1007/s13676-017-0103-x.

Crainic, T.G., Laporte, G., 1997. Planning models for freight transportation. Eur. J. Oper. Res. 97 (3), 409–438. doi:10.1016/S0377-2217(96)00298-6.

Crainic, T.G., Ricciardi, N., Storchi, G., 2009. Models for Evaluating and Planning City Logistics Systems. Transportation Science 43 (4), 432–454. doi:10.1287/trsc.1090.0279. ISSN: 0041-1655 (Print), 1526-5447 (Online) http://pubsonline.informs.org/doi/abs/10.1287/trsc.1090.0279.
Crainic, T.G., Rousseau, J.-M., 1986. Multicommodity, multimode freight transportation: Ageneral modeling and algorithmic framework for the service network design problem. Transportation Research Part B: Methodological 20 (3), 225–242. doi:10.1016/0191-2615(86)90019-6. http://www.sciencedirect.com/science/article/pii/0191261586900196.
Crainic, T.G., Sgalambro, A., 2014. Service network design models for two-tier city logistics. Optim. Lett. 8 (4), 1375–1387. doi:10.1007/s11590-013-0662-1.
Crosbie, J., 2017. Ford's Self-Driving Cars Will Live Inside Urban Geofences–Inverse. https://www.inverse.com/article/28876-ford-self-driving-cars-geofences-ride-sharing.
Dash, S., Günlük, O., Lodi, A., Tramontani, A., 2012. A time bucket formulation for the traveling salesman problem with time windows. INFORMS J. Comput. 24 (1), 132–147. doi:10.1287/ijoc.1100.0432.
Derigs, U., Pullmann, M., Vogel, U., 2013. Truck and trailer routing–Problems, heuristics and computational experience. Computers and Operations Research 40 (2), 536–546. doi:10.1016/j.cor.2012.08.007. https://www.sciencedirect.com/science/article/pii/S0305054812001724.
Fleischer, L., Skutella, M., 2007. Quickest flows over time. SIAM J. Comput. 36 (6), 1600–1630. doi:10.1137/S0097539703427215.
Fontaine, P., Crainic, T.G., Fontaine, P., Crainic, T.G., Jabali, O., Rei, W., 2017. Multi-Modal scheduled service network design with resource management for two-tier city logistics multi-modal scheduled service network design with resource management for two-tier city logistics. CIRRELT Working Paper (CIRRELT-2017-27). https://www.cirrelt.ca/documentstravail/CIRRELT-2017-27.pdf.
Heutger, M., Kückelhaus, M., 2014. Self-Driving Vehicles in Logistics. http://www.dhl.com/content/dam/downloads/g0/aboutus/logisticsinsights/dhlselfdrivingvehicles.pdf.
Hewitt, M., 2019. Enhanced dynamic discretization discovery for the continuous time load plan design problem. Transp. Sci. 53 (6). doi:10.1287/trsc.2019.0890.
Janjevic, M., Kaminsky, P., Ndiaye, A.B., 2013. Downscaling the consolidation of goods-state of the art and transferability of micro-consolidation initiatives. Eur. Transp. - Trasporti Europei (54) 1–23.
Joerss, M., Schröder, J., Neuhaus, F., Klink, C., Mann, F., 2016. Parcel delivery: the future of last mile, pp. 1–32. https://www.mckinsey.com/~/media/mckinsey/industries/travel%20transport%20and%20logistics/our%20insights/how%20customer%20demands%20are%20reshaping%20last%20mile%20delivery/parcel_delivery_the_future_of_last_mile.ashx.
Lammert, M.P., Duran, A., Diez, J., Burton, K., Nicholson, A., 2014. Effect of Platooning on Fuel Consumption of Class 8 Vehicles Over a Range of Speeds, Following Distances, and Mass. SAE International Journal of Commercial Vehicles 7 (2), 2014–01–2438. doi:10.4271/2014-01-2438. http://papers.sae.org/2014-01-2438/.
Larsson, E., Sennton, G., Larson, J., 2015. The vehicle platooning problem: computational complexity and heuristics. Transp. Res. Part C 60, 258–277. doi:10.1016/j.trc.2015.08.019.
Lioris, J., Pedarsani, R., Tascikaraoglu, F.Y., Varaiya, P., 2016. Doubling throughput in urban roads by platooning. IFAC-PapersOnLine 49 (3), 49–54. doi:10.1016/j.ifacol.2016.07.009.
Magnanti, T.L., Wong, R.T., 1984. Network design and transportation planning: models and algorithms. Transp. Sci. 18 (1), 1–55. doi:10.1287/trsc.18.1.1.
Mahmassani, H.S., 2016. 50Th anniversary invited article–autonomous vehicles and connected vehicle systems: flow and operations considerations. Transp. Sci. 50 (4), 1140–1162. doi:10.1287/trsc.2016.0712.
Medina, J., Hewitt, M., Lehuédé, F., Péton, O., 2019. Integrating long-haul and local transportation planning: the service network design and routing problem. EURO J. Transp. Logist. 8 (2), 119–145. doi:10.1007/s13676-017-0114-7.
Meisel, F., Kopfer, H., 2014. Synchronized routing of active and passive means of transport. OR Spectr. 36 (2), 297–322. doi:10.1007/s00291-012-0310-7.
Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies 54, 86–109. doi:10.1016/j.trc.2015.03.005. http://linkinghub.elsevier.com/retrieve/pii/S0968090X15000844.
Murray, C.C., Raj, R., 2020. The multiple flying sidekicks traveling salesman problem: parcel delivery with multiple drones. Transp. Res. Part C 110, 368–398. doi:10.1016/j.trc.2019.11.003.
Nothdurft, T., Hecker, P., Ohl, S., Saust, F., Maurer, M., Reschka, A., Böhmer, J.R., 2011. Stadtpilot: first fully autonomous test drives in urban traffic. In: IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, pp. 919–924. doi:10.1109/ITSC.2011.6082883.
Pedersen, M.B., Crainic, T.G., Madsen, O.B.G., 2009. Models and tabu search metaheuristics for service network design with asset-balance requirements. Transp. Sci. 43 (2), 158–177. doi:10.1287/trsc.1080.0234.
Perboli, G., Tadei, R., Vigo, D., 2011. The two-echelon capacitated vehicle routing problem: models and math-based heuristics. Transp. Sci. 45 (3), 364–380. doi:10.1287/trsc.1110.0368.
Persiel, S., 2017. DHL und Deutsche Post in Braunschweig. https://www.paketda.de/dhl-deutsche-post-braunschweig.html.
Poikonen, S., Golden, B., 2019. Multi-visit drone routing problem. Comput. Oper. Res. 113. doi:10.1016/j.cor.2019.104802.
Poikonen, S., Wang, X., Golden, B., 2017. The vehicle routing problem with drones: Extended models and connections. Networks 70 (1), 34–43. doi:10.1002/net.21746. http://doi.wiley.com/10.1002/net.21746.
SAE International, 2018. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. 10.4271/J3016_201806
Scherr, Y.O., Neumann-Saavedra, B.A., Hewitt, M., Mattfeld, D.C., 2018. Service network design for same day delivery with mixed autonomous fleets. Transp. Res. Procedia 30, 23–32. doi:10.1016/j.trpro.2018.09.004.
Scherr, Y.O., Neumann Saavedra, B.A., Hewitt, M., Mattfeld, D.C., 2019. Service network design with mixed autonomous fleets. Transp. Res. Part E 124, 40–55. doi:10.1016/j.tre.2019.02.001.
Schmeitz, A., Schwartz, R.S., Ravesteijn, D., Verhaeg, G., Altgassen, D., Wedemeijer, H., 2019. Paper number ITS-TP1800 EU AUTOPILOT project: Platooning use case in Brainport, pp. 3–6. https://autopilot-project.eu/wp-content/uploads/sites/16/2019/05/ITS19-AUTOPILOT-Platooning-TNOFINAL.pdf.
Sebe, S.M., Kraus, P., Müller, J.P., Westphal, S., 2019. Cross-provider platoons for same-day delivery. In: VEHITS 2019 - Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems, pp. 106–116. doi:10.5220/0007689601060116.
Skutella, M., 2009. An Introduction to Network Flows over Time. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 451–482.
Taniguchi, E., Dupas, R., Deschamps, J.-C., Qureshi, A.G., 2018. Concepts of an integrated platform for innovative city logistics with urban consolidation centers and transshipment points. In: City Logistics 3. John Wiley & Sons, Inc., pp. 129–146. http://doi.wiley.com/10.1002/9781119425472.ch7.
Ulmer, M.W., Streng, S., 2019. Same-Day delivery with pickup stations and autonomous vehicles. Comput. Oper. Res. 108, 1–19. doi:10.1016/j.cor.2019.03.017.
Urban Freight Lab, 2020. Common MicroHub Research Project: Research Scan. Technical Report. http://depts.washington.edu/sctlctr/urban-freight-lab-0.
Vigo, D., Toth, P., 2014. Vehicle Routing: Problems, Methods, and Applications, Second Edition. Society for Industrial and Applied Mathematics doi:10.1137/1.9781611973594.
Vu, D.M., Hewitt, M., Boland, N., Savelsbergh, M., 2019. Dynamic discretization discovery for solving the time dependent traveling salesman problem with time windows. Transp. Sci. 54 (3), 703–720. doi:10.1287/trsc.2019.0911.
Wang, X., Regan, A.C., 2002. Local truckload pickup and delivery with hard time window constraints. Transp. Res. Part B 36 (2), 97–112. doi:10.1016/S0965-8564(00)00037-9.
Wang, X., Regan, A.C., 2009. On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. Comput. Ind. Eng. 56 (1), 161–164. doi:10.1016/j.cie.2008.04.011.
Wang, Z., Sheu, J.B., 2019. Vehicle routing problem with drones. Transp. Res. Part B. 122, 350–364. doi:10.1016/j.trb.2019.03.005.
Wieberneit, N., 2008. Service network design for freight transportation: a review. OR Spectr. 30 (1), 77–112. doi:10.1007/s00291-007-0079-2.
WSP, 2016. Adapting infrastructure for a driverless future. https://www.wsp.com/en-GL/insights/adapting-infrastructure-for-a-driverless-future.