Journal of Scheduling (2020) 23:363–377
https://doi.org/10.1007/s10951-019-00620-y

Branch-cut-and-price for scheduling deliveries with time windows in a direct shipping network

Timo Gschwind[1] · Stefan Irnich[1] · Christian Tilk[1] · Simon Emde[2]

Published online: 6 September 2019

Abstract
In a direct shipping (or point-to-point) network, individual deliveries are round trips from one supplier to one customer and back to either the same or another supplier, i.e., a truck can visit only one customer at a time before it has to return to a supplier. We consider the multiple sources/multiple sinks case, where a given set of direct deliveries from a set of suppliers to a set of customers must be scheduled such that the customer time windows are not violated, the truck fleet size is minimal, and the total weighted flow time is as small as possible. Direct shipping policies are commonly employed in just-in-time logistics (e.g., in the automotive industry) and in humanitarian logistics. We present an exact branch-cut-and-price algorithm for this problem, which is shown to perform well on instances from the literature and newly generated ones. We also investigate under what circumstances bundling suppliers in so-called supplier parks actually facilitates logistics operations under a direct shipping policy.

Keywords Direct deliveries · Branch-cut-and-price · Weighted flow time · Just-in-time logistics

1 Introduction

Transportation networks for the delivery of physical goods are typically distinguished by their structure as milk runs, cross-docking, or direct shipping (Chopra and Meindl 2015, Chpt. 14). Milk run networks consolidate multiple customers or suppliers on one tour, whereas in a cross-docking network, shipments are routed through a transshipment hub and consolidated there. Such network structures are useful for improving economies of transportation if a shipment is less than a truckload (Du et al. 2007).

In direct shipping networks (also called point-to-point networks), which are the focus of this paper, goods are moved directly from source to sink, without any intermediary step. Such networks are widely used in many industries, such as the automotive industry (Boysen et al. 2015). Direct shipments are also common in many retail supply chains, not necessarily to supply stores directly, but to ship goods from supplier or wholesaler distribution centers to regional dispatch points (also referred to as pool points), where they are then further distributed to local retail stores via milk runs (Chen 2008). Other applications include disaster relief, where humanitarian goods must be moved from national emergency stockpiles to local staging areas (Knott 1987; De Angelis et al. 2007), or the transport of biomass fuel (Rauch et al. 2007).

Apart from being the fastest way to move goods from A to B, direct shipping can also be considered the easiest shipping policy to implement and coordinate. It has been shown to be the most efficient strategy if the economic lot size of the customers is close to the vehicle capacity (Gallego and Simchi-Levi 1990; Barnes-Schuster and Bassok 1997). For these reasons, direct deliveries are commonly used for

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10951-019-00620-y) contains supplementary material, which is available to authorized users.

✉ Timo Gschwind
gschwind@uni-mainz.de

Stefan Irnich
irnich@uni-mainz.de

Christian Tilk
tilk@uni-mainz.de

Simon Emde
emde@bwl.tu-darmstadt.de

1 Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, 55128 Mainz, Germany

2 Fachgebiet Management Science/Operations Research, Technische Universität Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany

just-in-time (JIT), high-velocity, high-bulk, perishable, and specialty goods (Chen 2008).

In this context, we tackle the *direct delivery scheduling problem in a network* (DDSP-N): given a set of *suppliers* (e.g., distribution centers or supplier plants) from which a set of *customers* (e.g., pool points or *original-equipment-manufacturer* (OEM) plants) need to be served via direct delivery (that is, neither suppliers nor customers can be aggregated on milk runs), which truck should make which delivery at what time? Since deliveries are usually made by third-party logistics providers, trucks can switch between suppliers. Moreover, since direct deliveries are particularly common for JIT and high-velocity goods, deliveries have time windows which must not be violated. From the viewpoint of the logistics provider, it is desirable to minimize, on the one hand, the cost of the truck fleet. On the other hand, customer waiting times should not be excessive, as this may lead to significant contractual penalties in the case of delays (Boysen et al. 2015).

The main contributions of this paper are as follows. First, we extend the model by Emde and Zehtabian (2019) to account for shipping networks with more than one supplier. Second, we develop a powerful exact *branch-cut-and-price* (BCP) algorithm to solve this problem, the first exact solution method (apart from a default solver) for the DDSP-N. Finally, through computational experiments, we derive some managerial insight into the usefulness of clustering suppliers (e.g., in “supplier parks”) under a direct shipping policy.

The remainder of this paper is structured as follows. In Sect. 2, we review the pertinent literature. In Sect. 3, we formally define the DDSP-N, review the mixed-integer programming model of Emde and Zehtabian (2019), and present a path-based formulation. Section 4 introduces our custom-tailored BCP schemes, which we test in Sect. 5. Finally, Sect. 6 ends the paper with conclusion and outlook.

2 Literature review

Direct shipping networks, although widespread in practice, have so far received attention almost exclusively from a strategic or supply chain management viewpoint, with the goal of determining optimal long-term supply policies considering both transportation and inventory holding costs. Such studies have been published, for example, by Gallego and Simchi-Levi (1990), Barnes-Schuster and Bassok (1997), and Chopra (2003). Meyer and Amberg (2018) review model-based approaches to selecting delivery strategies and frequencies in the automotive context. Boysen et al. (2015) survey parts logistics in the automotive industry and discuss point-to-point networks in this context in Section 3.2. of their paper.

To the best of our knowledge, the only paper explicitly dealing with direct shipping from a scheduling perspective, where a given set of trips need to be scheduled within a finite planning horizon, is Emde and Zehtabian (2019). The authors introduce the problem and develop several heuristics, restricting their efforts to the one-supplier/ multiple-receiver case. In this paper, we extend their problem by considering shipping networks with more than one supplier, as such cases are often encountered in practice.

Distributing goods from one or more suppliers to multiple customers bears some resemblance to classical (multi-depot) vehicle routing problems (VRP, surveyed, e.g., by Vigo and Toth 2014). However, these models are not immediately applicable to DDSP-N, because trucks do not need to return to a supplier between customer visits, and such models generally consider different objectives.

Uncapacitated routing problems that consider customer waiting times in the form of minimizing the sum of arrival times (without considering release dates) are referred to as *traveling repairman problems* (TRP), also called deliveryman or minimum latency problems. Literature on TRP with multiple vehicles is very limited, however. Fakcharoenphol et al. (2007), Luo et al. (2014), Nucamendi-Guillén et al. (2016), and Angel-Bello et al. (2019) are among the few contributions, assuming a given fixed number of vehicles and disregarding time windows. Most relevant to our research in this context is the paper by Luo et al. (2014), who propose a BCP algorithm for the multiple TRP, where the total distance per route is constrained but time windows are not considered. To solve the pricing problem, they employ a tabu search column generator and a bounded bidirectional label-setting algorithm, with which they can solve most instances with up to 50 customers within a time limit of 3 h. More recently, Bulhoes et al. (2018) have employed BCP to solve the classical TRP. Their main contribution lies in the improved use of ng-paths; specifically, they propose a fully dynamic, arc-based relaxation with strengthened dominance rules. These ideas are not immediately applicable to DDSP-N, however, due to its more complex structure (time windows, multiple vehicles, multi-criteria weighted objective).

Other routing problems with latency objectives, such as the *cumulative capacitated VRP* (e.g., Ribeiro and Laporte 2012) or the school bus routing problem (surveyed by Park and Kim 2010), are also structurally different from DDSP-N, because they assume given customer demands and limited vehicle capacities, which are immaterial for point-to-point deliveries. Algorithmically speaking, a few BCP methods have recently been proposed for the cumulative capacitated VRP (CCVRP). Lysgaard and Wøhlk (2014) observe that classical label-setting schemes for the VRP do not work for the CCVRP because the cost of traversing an edge is not independent of the route on which the edge is traversed. They therefore propose a label-setting algorithm which constructs nonelementary paths in reverse order in pseudo-polynomial time. Rounded capacity inequalities are separated iteratively. Another variant of such a pricing algorithm is used by Rivera et al. (2016), who solve a single-vehicle routing problem with a latency objective. While they do not employ column generation, they reformulate the problem as a resource-constrained shortest-path problem, which they solve via a label-setting algorithm.

Heuristics for the CCVRP have recently been proposed by Ke and Feng (2013), Sze et al. (2017), and Lalla-Ruiz and Voß (2019). Gaur and Singh (2017) propose a column generation-based heuristic in which the pricing problem of finding elementary paths is solved via dynamic programming somewhat similar to the approach used by Lysgaard and Wøhlk (2014). The eventual optimal solution to the restricted master problem is then heuristically rounded to obtain a feasible integer solution. Note, however, that none of these papers considers time windows, and they presuppose a given number of vehicles and uniform priorities, the sole exception being Gaur and Singh (2017), who consider node-specific weights.

DDSP-N also bears some resemblance to the multi-depot vehicle scheduling problem, where a set of timetabled trips must be assigned to a fleet of vehicles (e.g., Carpaneto et al. 1989; Dell'Amico et al. 1993; Ribeiro and Soumis 1994; Kulkarni et al. 2018). Direct deliveries are typically not exactly timetabled like a bus schedule, but merely have time windows which must not be violated.

Finally, scheduling direct deliveries is reminiscent of parallel machine scheduling if we interpret trucks as machines, trips as jobs, and transfer times between suppliers as setup times. Machine scheduling with setup times and related problems have been surveyed by Allahverdi et al. (2008), Allahverdi (2015), and Pinedo (2016). However, minimizing the number of machines (trucks) is an unusual objective in the machine scheduling context (Yu and Zhang 2009). If the number of vehicles/machines were fixed, our problem would be structurally similar to scheduling a set of jobs with given time windows and processing times on a set of identical parallel machines to minimize the total weighted flow time, where switching jobs requires a sequence-dependent setup time. In machine scheduling notation as introduced by Graham et al. (1979), this corresponds to the tuple $[P|r_j, d_j, S_{ij}|\sum w_j F_j]$.

To the best of our knowledge, $[P|r_j, d_j, S_{ij}|\sum w_j F_j]$ has not been solved before. However, BCP methods have been used to solve related parallel machine scheduling problems. The general BCP scheme for scheduling problems proposed by Chen and Powell (1999) and van Den Akker et al. (1999) typically assumes single-machine pricing problems to be solvable at least in pseudo-polynomial time, which is generally not the case for problems with setup times and/or time windows (Lenstra et al. 1977).

Lopes and Valério de Carvalho (2007) solve $[R|a_k, r_j, S_{ij}|\sum w_j T_j]$ via branch-and-price, i.e., they consider unrelated parallel machines with machine ready times, time windows, sequence-dependent setup times, and a total weighted tardiness objective. Note that while this problem is similar to our problem $[P|r_j, d_j, S_{ij}|\sum w_j F_j]$, it is in fact more tractable in the sense that finding a feasible solution is easy because the time windows are soft. Because of the hard deadlines of $[P|r_j, d_j, S_{ij}|\sum w_j F_j]$, even finding a feasible solution on a single machine is already NP-hard. Lopes and Valério de Carvalho (2007) present three alternative dynamic program formulations for the pricing subproblem. In these dynamic programs, schedules are constructed in a forward direction. Only the dynamic program with two-cycle elimination is actually solved. It allows nonelementary schedules only if cycles have a length of at least three jobs. To accelerate the subproblem solution, the authors applied a "primal box" method which restricts the primal search space by introducing a tentative upper bound T for the makespan. With the upper bound, the subproblem becomes (practically) simpler to solve, because schedules are restricted in length. Iteratively, the upper bound T on the makespan is increased, and additional schedules can be generated until it can be proven that no more negative reduced-cost schedules exist in the unrestricted subproblem.

Pessoa et al. (2010) use an arc-time-indexed formulation of $[P||\sum w_j T_j]$ and solve it via BCP, i.e., they price out schedules using a time-discretized network in which vertices are pairs of a job and a job completion time. For the solution of the pricing problem, a forward dynamic program on the time-discretized network is presented. An arc-fixing procedure is also used, in which two complete dynamic programs must be solved, in both forward and backward direction. The backward direction can be handled here, since in a time-discretized network the trade-off curve can be propagated in point-wise fashion. This is certainly computationally expensive, but arc fixing is done in only a few pricing iterations. Dual stabilization techniques, strong branching, and extended capacity cuts are used to further speed up the solution procedure.

Table 1 provides a synopsis of the algorithmic components for all three BCP algorithms, used in the works of Lopes and Valério de Carvalho (2007) and Pessoa et al. (2010), and in our work.

More recently, Tanaka and Araki (2013) address $[1|S_{ij}|\sum w_j T_j]$ by solving a Lagrangian relaxation of the problem via successive sublimation dynamic programming. Constraints are added successively to the relaxation to close the gap between upper and lower bounds. The optimal Lagrangian multipliers are determined by column generation. Other exact and heuristic algorithms have recently been proposed by Gacias et al. (2010) for $[Pm|\text{prec}, r_j, S_{ij}|\sum C_j]$ (branch-and-bound, local search heuristic), Afzalirad and

Table 1 Synopsis of algorithmic components used in the works of Lopes and Valério de Carvalho (2007) and Pessoa et al. (2010), and in our BCP algorithm

	Lopes and Valério de Carvalho (2007)	Pessoa et al. (2010)	Our approach
Problem	$[R\|a_k, r_j, S_{ij}\| \sum w_j T_j]$	$[P\|\| \sum w_j T_j]$	$[P\|r_j, d_j, S_{ij}\| \sum w_j F_j]$
Master problem			
Stabilization	No	Yes	No
Cutting	(None)	Extended capacity cuts	Subset-row inequalities
Branching	Arc flow	Arc-time flow	Time window and arc flow
Subproblem			
Network	Not discretized	Job-time discrete	Not discretized
Relaxation	Nonelementary without 2 cycles	None	ng-path relaxation
Acceleration	Primal box	Arc fixing	Hierarchy of reduced networks

Rezaeian (2016) for $[Rm|r_j, S_{ij}, M_j, \text{prec}, \text{res}|C_{\max}]$ (metaheuristics), and Hamzadayi and Yildiz (2017) for $[Pm, S|S_{ij}|C_{\max}]$ (metaheuristics). All of these papers assume that the number of machines is given.

3 Definition and formulations of the DDSP-N

The DDSP-N can be formally defined as follows: The physical network $\mathcal{D} = (\mathcal{L}, \mathcal{A})$ has the vertex set $\mathcal{L}$ representing the relevant physical locations, i.e., $\mathcal{L} = \{0, 0'\} \cup \mathcal{S} \cup \mathcal{N}$, where 0 and $0'$ are start and destination depots of the trucks, $\mathcal{S}$ is the set of suppliers (=terminals, distribution centers), and $\mathcal{N}$ the set of customers (=OEMs, pool points).

Additionally, we are given the set I of trips that must be performed. Each trip $i \in I$ is defined by the following additional attributes:

s_i:	the supplier $s_i \in \mathcal{S}$ from where goods are shipped;
n_i:	the customer $n_i \in \mathcal{N}$ who receives these goods;
r_i:	the ready times, i.e., the earliest time when a truck can depart from supplier s_i with the shipment associated with the trip;
d_i:	the deadline, i.e., the latest time the service has to be completed at customer n_i;
p_i:	the processing time of the trip, including the truck loading time at the supplier, the driving time to and the unloading time at the customer, plus the time it takes to do any other activities necessary to ready the truck for its next trip (possibly refueling or breaks for the driver);
w_i:	the weight of the trip to be used in the minimum weighted flow time objective (see below).

We assume that a truck can travel along the following connections: between

1. origin depot and all suppliers, i.e., $(0, s) \in \{0\} \times \mathcal{S} \subset \mathcal{A}$,
2. all customers and all suppliers, i.e., $(n, s) \in \mathcal{N} \times \mathcal{S} \subset \mathcal{A}$,
3. all trips with their associated supplier–customer pair, i.e., $\{(s_i, n_i) : i \in I\} \subset \mathcal{A}$, and
4. all customers and the destination depot, i.e., $(n, 0') \in \mathcal{N} \times \{0'\} \subset \mathcal{A}$.

Finally, we define the travel time $\tau(n, s)$ for the connections $(n, s) \in \mathcal{N} \times \mathcal{S}$ between customers and suppliers. For the reader's convenience, we have summarized our notation in the Appendix.

The task of the DDSP-N is to feasibly schedule all trips and assign them to trucks such that (i) the number of trucks employed is minimal and (ii) the weighted flow time of all trips is minimal. Emde and Zehtabian (2019) define the DDSP-N with a hierarchical objective with priority on (i).

To precisely define the *weighted flow time*, we introduce schedule variables $T_i \in \mathbb{R}$ for all trips $i \in I$. The variable T_i describes the point in time when the trip execution is started (with loading at its supplier s_i). These schedule variables are bounded by $r_i \leq T_i \leq d_i - p_i$. If two trips $i, j \in I$ are assigned to the same truck, then either i precedes j, leading to $T_i + p_i + \tau(n_i, s_j) \leq T_j$, or j precedes i, implying $T_j + p_j + \tau(n_j, s_i) \leq T_i$. The weighted flow time objective is given by $\sum_{i \in I} w_i(T_i - r_i)$.

3.1 Compact formulation of Emde and Zehtabian (2019)

The single-supplier variant of the DDSP-N was modeled by Emde and Zehtabian (2019). With slight modifications, this formulation also models the multiple-supplier case. The *mixed-integer programming* (MIP) formulation of Emde and

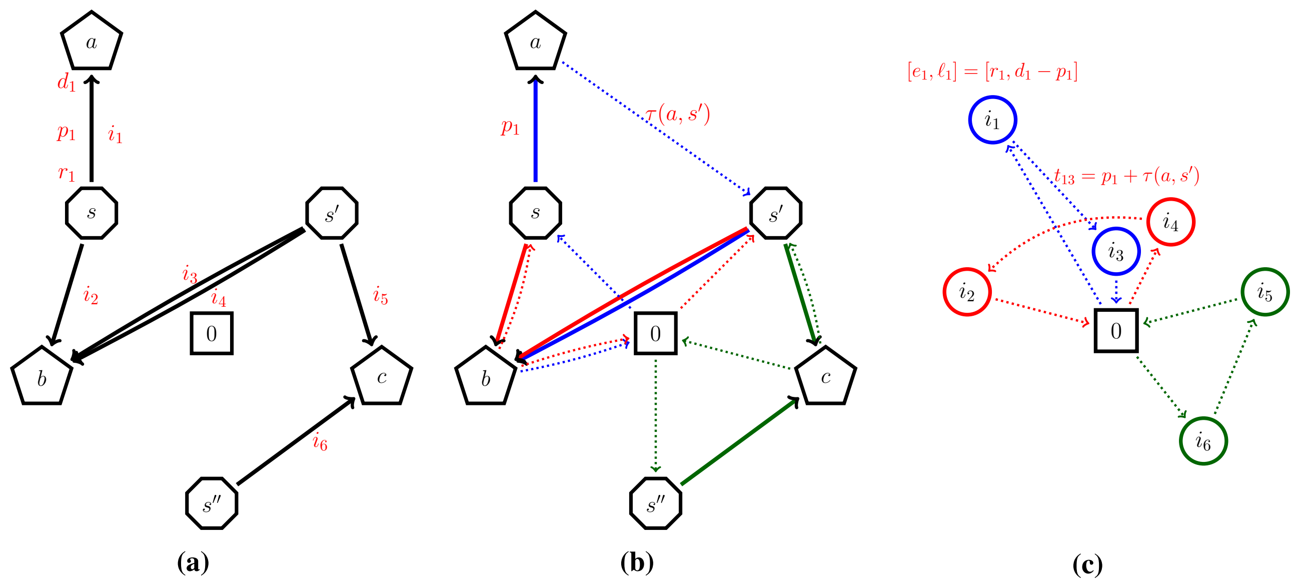

Fig. 1 **a** DDSP-N instance with three suppliers $\{s, s', s''\}$, three customers $\{a, b, c\}$, and six trips $I = \{i_1, i_2, \dots, i_6\}$ with $i_1 = (s, a)$, $i_2 = (s, b)$, $i_3 = i_4 = (s', b)$, $i_5 = (s', c)$, and $i_6 = (s'', c)$; **b** solution with three trucks; **c** trip-based digraph (V, A) with the corresponding solution

Zehtabian uses, in addition to the schedule variables T_i for all $i \in I$, binary connection variables x_{ij} for all $i \in I \cup \{0\}$ and $j \in I \cup \{0'\}$. For two trips $i, j \in I$, the variable x_{ij} is one if i is performed directly before j by the same truck. The variables x_{0j} are one if $j \in I$ is the first trip performed by a truck. Similarly, $x_{i0'}$ equal to one means that $i \in I$ is the last trip performed by a truck. The model now reads as follows:

$$\min \sum_{j\in I} M \cdot x_{0j} + \sum_{i\in I} w_i \cdot (T_i - r_i) \tag{1a}$$

$$\text{s.t.} \sum_{j\in I\cup\{0'\}} x_{ij} = 1 \qquad \forall i \in I \tag{1b}$$

$$\sum_{i\in I\cup\{0\}} x_{ij} = 1 \qquad \forall j \in I \tag{1c}$$

$$\left(1 - x_{ij}\right) \cdot M + T_j \geq T_i + p_i + \tau(n_i, s_j) \qquad \forall (i, j) \in I \times I \tag{1d}$$

$$r_i \leq T_i \leq d_i - p_i \qquad \forall i \in I \tag{1e}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in I \cup \{0\},\ j \in I \cup \{0'\} \tag{1f}$$

The objective function (1a) gives priority to the minimization of the fleet (using a big-M constant). The secondary objective is the minimization of the total weighted flow time. Constraints (1b) and (1c) ensure that each trip has exactly one successor and one predecessor, respectively, which can be either another trip or one of the depots 0 or $0'$. Inequalities (1d) make it impossible for a truck to make two trips concurrently. Finally, (1e) ensure that trips are processed within their time windows and (1f) define the domain of the binary connection variables.

Obviously, constraints (1d) can be eliminated from formulation (1) for those pairs $(i, j) \in I \times I$ where i cannot precede j, i.e., $r_i + p_i + \tau(n_i, s_j) > d_j - p_j$. This strengthens the linear relaxation of formulation (1).

3.2 Path-based formulation

We now model the movement of a truck as a route in a trip-based network. We define the underlying digraph $D = (V, A)$ with vertex set $V = I \cup \{0, 0'\}$ and arc set A. Each trip vertex $i \in I$ has an associated time window $[e_i, \ell_i] := [r_i, d_i - p_i]$ [see also (1e)] and service time p_i. For the origin and destination depot (0 and $0'$), we assume non-constraining time windows $[e_0, \ell_0] = [e_{0'}, \ell_{0'}]$ and define the weights $w_0 := w_{0'} := 0$.

The arc set A is a subset of $(\{0\} \times I) \cup (I \times I) \cup (I \times \{0'\})$, where we define travel times $t_{0j} := 0$ for $(0, j) \in \{0\} \times I$, $t_{ij} := p_i + \tau(n_i, s_j)$ for $(i, j) \in I \times I$, and $t_{i0'} := 0$ for $(i, 0') \in I \times \{0'\}$. Those arcs $(i, j) \in A$ that fulfill $e_i + t_{ij} \leq \ell_j$ are feasible with respect to the travel and service times and time windows defined above; all other arcs can be eliminated. Figure 1 depicts the two modeling possibilities of an DDSP-N instance and two representations of a solution.

A DDSP-N route P is an elementary 0-$0'$-path in D that is feasible with respect to the travel and service times and time windows defined above. Formally, for a route $P = (i_0, i_1, \dots, i_p, i_{p+1})$ (recall that $i_0 = 0$ and $i_{p+1} = 0'$), there must exist a schedule $(T_0, T_1, \dots, T_p, T_{p+1}) \in \mathbb{R}^{p+2}$ satisfying the following two constraints:

$$T_k \in [e_{i_k}, \ell_{i_k}] \qquad \forall k \in \{0, 1, 2, \ldots, p+1\} \tag{2a}$$
$$T_{k-1} + t_{i_{k-1}i_k} \leq T_k \qquad \forall k \in \{1, 2, \ldots, p+1\}. \tag{2b}$$

These are classical time window constraints known from the vehicle routing problem with time windows (VRPTW, Desaulniers et al. 2014).

In the case of feasibility, using the *as-early-as-possible* schedule, i.e., $T_0 := e_{i_0}$ and $T_k := \max\{e_{i_k}, T_{k-1} + t_{i_{k-1}i_k}\}$ for $k = 1, 2, \ldots, p+1$, the accumulated weighted flow time of the route P is

$$w_P := \sum_{k=0}^{p+1} w_{i_k} \cdot (T_k - e_{i_k}) = \sum_{k=1}^{p} w_{i_k} \cdot (T_k - r_{i_k}).$$

Let Ω be the set of all feasible DDSP-N routes, and a_{iP} an indicator of whether trip $i \in I$ is performed by route $P \in \Omega$. The set-partitioning formulation uses variables λ_P for $P \in \Omega$ to select the routes the solution comprises:

$$\min \sum_{P\in\Omega} (M + w_P)\lambda_P \tag{3a}$$
$$\text{s.t.} \sum_{P\in\Omega} a_{iP}\lambda_P = 1 \qquad \forall i \in I \tag{3b}$$
$$L \leq \sum_{P\in\Omega} \lambda_P \leq U \tag{3c}$$
$$\lambda_P \in \{0, 1\} \qquad \forall P \in \Omega \tag{3d}$$

Analogous to (1a), the objective (3a) is hierarchical, minimizing the fleet size first and the total weighted flow time second. The partitioning constraints (3b) ensure that each trip is performed exactly once. The fleet size is bounded from below (above) by L (U) with the help of constraints (3c) (see Sect. 4.3), and (3d) are the binary constraints.

4 Branch-Cut-and-Price algorithm

For solving the linear relaxation of the path-based formulation (3), we use a column generation algorithm (Desaulniers et al. 2005). Starting with a subset $\Omega' \subset \Omega$ of the feasible routes, the linear relaxation of formulation (3) defined over Ω' is denoted as the *restricted master program* (RMP). The column generation algorithm alternates between the re-optimization of the RMP and the solution of the column generation pricing problem that adds negative reduced-cost variables to the RMP, if any exist (see next section). If no reduced-cost variable exists, the current linear relaxation is solved to optimality. Branching is required to finally ensure integer solutions.

Section 4.1 presents the pricing problem of formulation (3). To strengthen the linear relaxation of formulation (3), Sect. 4.2 briefly describes known valid inequalities for the RMP and separation strategies for them. In Sect. 4.3, we present our DDSP-N-tailored branching schemes.

4.1 Pricing problem

Let $\pi_i, i \in I$, be the dual prices of the partitioning constraints (3b), and let μ_L and μ_U be the dual prices of fleet size constraints (3c). The task of the pricing problem is to identify at least one feasible route $P \in \Omega$ with negative reduced cost

$$\tilde{c}_P = M + w_P - \mu_L - \mu_U - \sum_{i\in I} a_{iP}\,\pi_i$$

or guarantee that no such route exists. This problem is a variant of the *elementary shortest path problem with resource constraints* (ESPPRC), which can be solved by means of a dynamic programming labeling algorithm (Irnich and Desaulniers 2005). Such an algorithm creates partial paths by moving forward from an origin to a destination vertex in a network. Here, a label stores all necessary information on the resource consumption up to the endpoint of a partial path. Labels are propagated along the network arcs by *resource extension functions* (REFs, Desaulniers et al. 1998; Irnich 2008). Dominance procedures are invoked to identify and discard those partial paths and their labels that cannot lead to an optimal SPPRC solution. The main difference from classical ESPPRCs (as pricing problems of VRP variants) is that there are no routing costs in the DDSP-N. Instead, the pricing problem has to deal with the accumulated weighted flow time cost. The associated reduced cost of an arc $(v, v') \in A$ can be defined as

$$\tilde{c}_{vv'} := -\frac{1}{2}(\pi_v + \pi_{v'}) + \begin{cases} M, & \text{if } v = 0 \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where we define $\pi_0 := \pi_{0'} := \mu_L + \mu_U$. With this definition, $\tilde{c}_P = w_P + \sum_{(v,v')\in A(P)} \tilde{c}_{vv'}$ holds for all routes $P \in \Omega$.

A label in the DDSP-N is associated with a partial path ending at vertex $v \in V$ and has the attributes $(C_v, T_v, (S^i_v))$, where C_v is the accumulated reduced cost (including the weighted flow time), T_v is the time at vertex v (according to the as-early-as-possible schedule), and (S^i_v) is the count of how often a trip $i \in I$ has already been performed by the partial path. The initial label at vertex 0 is $(C_0, T_0, (S^i_0)) = (0, e_0, \mathbf{0})$. An arbitrary label $(C_v, T_v, (S^i_v))$ is extended along an arc $(i, j) \in A$ creating a new label $(C_{v'}, T_{v'}, (S^i_{v'}))$ using the following REFs:

$$C_{v'} = REF^{rdc}(C_v, T_v, (S^i_v)) = C_v + \tilde{c}_{vv'} + w_{v'} \cdot (T_v + t_{vv'} - e_j)^+ \tag{5a}$$
$$T_{v'} = REF^{time}(C_v, T_v, (S^i_v)) = \max\{e_{v'}, T_v + t_{vv'}\} \tag{5b}$$
$$S^i_{v'} = REF^{visit,i}(C_v, T_v, (S^i_v)) = S^i_v + \delta_{v'i}, \quad \forall i \in I \tag{5c}$$

where $\delta_{v'i}$ is the Kronecker delta equal to one if $v' = i$, and 0 otherwise.

The new label $(C_{v'}, T_{v'}, (S^i_{v'}))$ is feasible if all of the following constraints are fulfilled:

$$T_{v'} \leq \ell_{v'} \tag{6a}$$

$$S^i_{v'} \leq 1 \quad \forall i \in I \tag{6b}$$

Note that for any feasible route $P \in \Omega$, the propagation of the initial label along the route's arcs according to (5) yields a final label $(C_{0'}, T_{0'}, (S^i_{0'}))$ at vertex $0'$ with $C_{0'} = \tilde{c}_P$.

Since the REFs (5) are non-decreasing and attributes are bounded only from above by (6), labels can be compared with the standard $\leq$-operator to impose the following valid dominance rule: Let $(C_v, T_v, (S^i_v))$ and $(C'_v, T'_v, (S'^i_v))$ be two different labels associated with the same vertex $v \in V$. Then, $(C_v, T_v, (S^i_v))$ dominates $(C'_v, T'_v, (S'^i_v))$ if $C_v \leq C'_v$, $T_v \leq T'_v$, and $S^i_v \leq S'^i_v$ for all $i \in I$ hold. All dominated labels can be discarded as long as, for each of them, at least one dominating label is kept. In order to accelerate the solution of the pricing problem, we relax the elementary requirement and use the well-known ng-path relaxation developed by Baldacci et al. (2011).

Despite the general success of bidirectional labeling (see, e.g., Righini and Salani 2006; Tilk et al. 2017), we do not employ it here. The reason is that backward labeling is nontrivial for the DDSP-N due to the computation of the weighted flow time w_P. The simple case, as done in the VRPTW, is a backward labeling that propagates an *as-late-as-possible* schedule. This schedule however overestimates the true weighted flow time making the definition of an effective dominance rule impossible. An exact computation of the weighted flow time in backward labeling can be established using a time–cost trade-off function such as the piecewise linear trade-off functions used in Ioachim et al. (1998); Tilk et al. (2018). However, the backward case becomes much more difficult than the forward case, so that we expect a bidirectional labeling to finally not pay off. The computational results discussed in Sect. 5 also show that for the DDSP-N instances studied, the maximum speedup for the overall BCP algorithm is definitely smaller than factor 1.65, probably far less than this.

4.2 Cutting

Different classes of valid inequalities to strengthen the linear relaxation have been successfully applied in BCP algorithms for many VRP variants.

We only use *subset-row inequalities* (Jepsen et al. 2008) defined on sets $U \subset I$ of cardinality three. They are given by $\sum_{P\in\Omega} \lfloor \frac{h_P}{2} \rfloor \lambda_P \leq 1$, where h_P is the number of times route P visits a customer in U. Violated subset-row inequalities can be separated by straightforward enumeration. The addition of subset-row inequalities to the RMP, however, requires some adjustments in the pricing problem algorithm. The value of the dual price of a subset-row inequality defined on a vertex set U has to be subtracted from the reduced cost of a label for every second visit to vertices in U. Therefore, an additional resource is needed for each inequality counting the number of times that a vertex in U is visited. For details on the adjustments see Jepsen et al. (2008).

In pretests, we also experimented with *2-path cuts* (Kohl et al. 1999), which turned out to be not helpful. Hence, we do not use them for solving the DDSP-N.

4.3 Branching

Branching is required to finally ensure integer solutions of formulation (3). Let $\bar{\lambda}_P$ be the values of the corresponding decision variables λ_P, $P \in \Omega$, and let $\bar{x}_{ij} := \sum_{P\in\Omega} b_{ijP} \bar{\lambda}_P$, $(i, j) \in A$, where b_{ijP} indicates whether route P uses arc (i, j). We use two different branching schemes that are computationally evaluated in Sect. 5.2. In the first scheme, we apply a standard two-stage hierarchical branching as it is employed in most BCP approaches for VRP variants. In the first stage, we branch on the number of vehicles, if fractional, by setting the bounds in constraint (3c) to $L := \lceil \sum_{P\in\Omega} \bar{\lambda}_P \rceil$ in one branch and $U := \lfloor \sum_{P\in\Omega} \bar{\lambda}_P \rfloor$ in the other. If the number of vehicles is an integer, the branching in the second stage is employed. There, we branch on arcs by choosing an arc $(i, j) = \arg\max_{(i,j)\in A} \{(1 - |\bar{x}_{ij} - 0.5|) w_j\}$. The zero-branch is implemented by eliminating (i, j) from the network D, while for the one-branch, all ingoing arcs of j and all outgoing arcs of i are eliminated except for the selected arc (i, j).

In the second scheme, we apply a three-stage hierarchical branching by using branching on time windows of vertices after branching on the number of vehicles and before branching on arcs. Branching on time windows in column generation approaches for time-constrained routing problems was introduced in Gélinas et al. (1995). The idea is that there may be two or more routes visiting the same vertex at different points in time in a fractional solution of the RMP, while in an integer solution, each vertex is visited at a unique point in time. Such fractional solutions can be cut off by creating two branches that split the time window of a specific vertex in two disjunct parts. Branching on time windows is implemented as follows: We select a vertex i for which the difference between the latest and the earliest visit time at this vertex in the current solution is maximal, i.e., $i = \arg\max\{\max_{P\in\Omega:a_{iP}=1}\{T^P_i\} - \min_{P\in\Omega:a_{iP}=1}\{T^P_i\}\}$, where T^P_i denotes the visit time at vertex i on route $P \in \Omega$. The time window is then split into $[e_i, \bar{t}_i - 1]$ and $[\bar{t}_i, \ell_i]$, where $\bar{t}_i = \lceil \sum_{P\in\Omega} T^P_i \bar{\lambda}_P \rceil$ is the average visit time at vertex i in the fractional solution. Note that this branching rule does not ensure integrality of the route variables, since there can also be two or more routes visiting a vertex at the same time if the solution is fractional. Therefore, we have to apply branching on arcs in a third stage to complete the branching scheme.

To accelerate the overall solution process, we apply *strong branching* (SB) at the second and third stages. The decision on which arc or time window branching is performed is then taken according to the product rule (Achterberg 2007, p. 62). A best-bound-first strategy is applied as node selection rule, because our primary goal is to improve the dual bound.

5 Computational results

In this section, we report computational results for our BCP algorithm. All results are obtained using a standard PC with an Intel(R) Core(TM) i7-5930k processor clocked at 3.5 GHz, 64 GB RAM, and Windows 7 Enterprise. The algorithms are implemented in C++ and compiled into 64-bit single-thread code with Microsoft Visual Studio 2013. The callable library of CPLEX 12.6.2 is used for solving the RMP. We use a hard CPU time limit of 3600 s for all computations. The RMP is initialized with big-M variables, and no primal heuristic is employed. To speed up the solution of the pricing problem, we use the *ng-path relaxation* (Baldacci et al. 2011) with a neighborhood size of ten, as is common for VRPs. Moreover, to speed up the column generation process, we use two arc-reduced networks as heuristic pricers (Irnich and Desaulniers 2005, p. 57), in which only a subset of the arcs is considered, while ensuring that a minimum of 10 and 20 arcs, respectively, enter and exit each node. Because of the hierarchical objective, we first branch on the number of vehicles if fractional. Cuts are then separated only in the resulting nodes. Otherwise, cuts are separated only at the root node. Subset-row inequalities are separated up to a maximum of 60 cuts in total, with a maximum of 10 cuts in each round of separation.

In Sect. 5.1, we introduce the instances we used to test our BCP algorithm. Section 5.2 reports results for the different branching strategies, and a comparison with Emde and Zehtabian (2019) for the single-supplier variant is given. Finally, Sect. 5.3 provides managerial insight into the usefulness of clustering suppliers into supplier parks.

5.1 Instances

We test our algorithm on the single-supplier instances from Emde and Zehtabian (2019). A single-supplier instance is defined by a set of trips I, each having a certain release date and deadline as well as a processing time and a weight in the objective. Emde and Zehtabian (2019) generated two classes with 10 instances each, in which the number of trips is 25 and 125, respectively. The planning horizon is one day with 24 h, and a time unit is 10 min. All other parameters are drawn at random from given distributions.

To deal with the multi-supplier case, we created additional classes of instances in a similar way: Each trip is associated with a random supplier, and travel times between customers and suppliers based on Euclidean distances are given. Therefore, each physical location is defined by an x- and y-coordinate taken uniformly randomly from $[1, 50]$, and the processing times p_i are defined as travel distance plus a service time randomly taken from $[1, 10]$. Time windows are randomly created as follows: r_i is taken randomly from $[1, 144]$ and d_i is randomly taken from $[r_i + p_i + 3, r_i + p_i + 24]$. All processing times as well as release and due dates are integers. Moreover, we assume that each client orders multiple trips during the planning horizon such that there exist 5 to 10 trips to each customer location, which is realistic for JIT distribution systems (Chuah and Yingling 2005). In total, we created 24 new classes consisting of five instances each. The classes differ in the number of trips (50, 100, 150, or 200), the number of suppliers (2, 4, 6), and the length of the planning horizon (1 or 2 days corresponding to 144 or 288 time units, respectively). The 2-day planning horizon corresponds to one time period of 48 consecutive hours, and we also double the time window width of trips, i.e., r_i is taken randomly from $[1, 288]$ and d_i is taken randomly from $[r_i + p_i + 6, r_i + p_i + 48]$. The new instances are available at http://logistik.bwl.uni-mainz.de/benchmarks.php.

5.2 Results

In a first test, we compare the two-stage and the three-stage branching schemes with no, medium or aggressive SB, and with and without subset-row cuts. While medium SB evaluates up to 10 candidates and is applied up to level 20, aggressive SB evaluates up to 40 candidates and is applied up to level 40. Table 2 summarizes the results for all combinations of these parameters. Therein, instances are grouped by the number $|S|$ of suppliers and the planning horizon (PH), resulting in six groups with 20 instances each. The table contains, for each instance group and each of the 12 strategies, the number of solved instances (Opt), the average solution time in seconds (T), and the average number of solved nodes in the branch-and-bound tree (BB).

Comparing the branching schemes, the three-stage branching scheme is clearly superior to the two-stage branching. For all combinations of the other parameters, the BCP algorithm with the three-stage branching scheme is able to solve more instances and the solution time decreases significantly on average. Note that SB only improves the two-stage branching scheme, while it slightly worsens the performance of

Table 2 Comparison of different branching and cutting strategies; each row reports aggregated results for 20 instances

Instances		No strong branching						Medium strong branching						Aggressive strong branching					
|S|	PH	No cuts			Subset-row cuts			No cuts			Subset-row cuts			No cuts			Subset-row cuts		
		Opt	T	BB	Opt	T	BB	Opt	T	BB	Opt	T	BB	Opt	T	BB	Opt	T	BB
Two-stage branching scheme																			
2	1	11	1632	14,488	11	1638	11,191	11	1629	7244	11	1627	4866	12	1532	1599	12	1564	1561
4	1	11	1674	13,298	11	1696	10,027	12	1501	3281	12	1500	2020	14	1266	499	15	1268	396
6	1	10	1844	16,220	11	1662	9589	12	1505	3076	15	1297	1008	16	1078	483	15	1165	357
2	2	9	2297	4068	9	2079	2143	10	1899	2007	10	1882	1187	10	1924	304	10	1938	230
4	2	8	2284	3728	10	2019	1870	9	2148	1275	11	1777	123	11	1789	129	11	1832	62
6	2	7	2659	7288	7	2481	3429	9	2300	1508	8	2299	1055	10	2089	809	11	2073	126
All		56	2065	9848	59	1929	6375	63	1830	3065	67	1730	1710	73	1613	637	74	1640	455
Three-stage branching scheme																			
2	1	18	801	4385	18	853	3755	17	1012	833	17	1003	683	16	1039	375	16	1111	326
4	1	17	769	3358	17	809	2868	17	838	538	17	897	446	17	845	273	17	971	231
6	1	16	775	3179	16	771	2595	16	833	494	16	863	406	16	870	249	16	910	211
2	2	12	1527	1175	13	1391	684	12	1666	229	10	1936	484	12	1675	123	11	1897	93
4	2	11	1724	910	11	1682	565	11	1748	201	10	1925	337	10	1927	159	11	1937	75
6	2	13	1599	1001	13	1512	676	12	1797	274	11	1831	149	10	2009	175	10	2207	135
All		87	1199	2335	88	1170	1857	85	1316	428	81	1409	418	81	1394	226	81	1505	179

Table 3 Detailed results of our best strategy

Instances			Opt	T			V′	V′ + 1	WFT (%)			T^{PP}/T
|C|	PH	#		Min	Avg	Max			Min	Avg	Max	
50	1	15	15	0.1	0.4	1.7	8.5	–	–	–	–	0.39
100	1	15	15	1.4	13.0	99.2	14.5	–	–	–	–	0.47
150	1	15	14	11.9	747.2	3600.0	21.3	0	0.17	0.17	0.17	0.30
200	1	15	7	129.3	2483.4	3600.0	26.7	2	0.10	1.57	5.69	0.23
50	2	15	15	0.5	1.9	5.8	4.7	–	–	–	–	0.43
100	2	15	14	24.7	499.7	3600.0	7.7	0	0.12	0.12	0.12	0.50
150	2	15	7	125.5	2243.0	3600.0	11.1	1	0.12	6.23	17.50	0.44
200	2	15	1	375.2	3388.0	3600.0	14.1	2	0.06	7.98	33.42	0.47
All		120	88	0.1	1172.1	3600.0	13.6	5	0.06	5.52	33.42	0.39

the three-stage branching scheme. This behavior can be attributed to the number of solved nodes in the branch-and-bound tree that decreases by almost factor 15 in the two-stage scheme and by only around factor ten in the three-stage scheme when aggressive SB is applied. The latter percentage does not suffice to compensate the additional computation time induced by the evaluation of branching candidates in the SB procedure. A similar behavior can be observed for the use of subset-row cuts: They are generally beneficial in the two-stage branching scheme. However, in the (superior) three-stage branching scheme, they only slightly improve the performance when no SB is applied, while in combination with SB they degrade the performance.

In all settings, there is no significant difference in the number of solved instances or in the average solution time when the number of suppliers varies. On the other hand, the number of solved instances decreases significantly and the solution times increase significantly in all settings for the instance groups with a two-day planning horizon.

In Table 3, we give more detailed results on our best algorithm (with subset-row cuts, three-stage branching scheme without SB). In addition, we now group the instances by the number of customers and the length of the planning horizon, since the number of suppliers does not affect the computation time significantly. The table contains for each instance group the number of instances (#), the number of solved

Table 4 Comparison with Emde and Zehtabian (2019) on single-supplier instances

|C|	#	BCP		CPLEX EZ		Heuristic EZ		
		Opt	T	Opt	T	Opt	$V' + 2/1$	WFT (%)
25	10	10	0.1	10	6.0	3	0/0	4.1
125	10	10	199.0	0	1800.0	0	1/9	–
All	20	20	99.6	10	903.0	3	1/9	4.1

instances, the minimum, average, and maximum computation time needed to solve an instance to proven optimality, and the average number of vehicles used in the best known solutions (V′). Moreover, the table presents, for the unsolved instances, two types of gaps between the overall lower bound (at termination) and the best known solution: (i) the number of times that the number of vehicles in the lower bound differs from the best known solution by exactly one $V' + 1$, and (ii) the minimum, average, and maximum percentage gap in weighted flow time (WFT (%)). The best known solutions are taken over all tests and pretests we have performed. Note that the difference in the number of vehicles is at most one in all unsolved instances. We compute the weighted flow time gap only over all unsolved instances and only over those in which the number of vehicles in the lower bound is equal to the number of vehicles in the best known solution. Additionally, the percentage of time spent with pricing relative to the total computation time is given in column T^{PP}/T.

The table shows that the number of customers significantly affects the solution time. Regarding the gaps, the number of vehicles in the lower bound differs only five times from the best known solution, and the gap in weighted flow time is around 5 % on average. Note that on average, only 39 % of the total computation time is spent in pricing. This can be attributed to the low practical difficulty of the pricing problems, when solved with the forward labeling algorithm of Sect. 4.1.

Next, we test our algorithm on the single-supplier instances of Emde and Zehtabian (2019). Table 4 summarizes the results. The table contains for both instance classes the number of solved instances and the average solution times of our BCP algorithm as well as of the solution of the compact model in CPLEX carried out by Emde and Zehtabian (2019). Moreover, we compare the best heuristic solutions found in Emde and Zehtabian (2019) regarding the difference in the number of vehicles and the gap in terms of weighted flow time compared with the optimal solution. Summarizing, our BCP algorithm is able to solve all single-supplier instances to optimality, with an average computation time of 0.1 s for the 25 trip instances and 199.0 s for the 125 trip instances, while CPLEX solves all 25 trip instances in an average of 6.0 s but cannot solve any of the 125 trip instances in 1800 s of computation time. Moreover, the detailed results show that the computation times of our BCP algorithm never exceed 485 s on the single-supplier instances. Table 4 also reveals that the heuristic of Emde and Zehtabian (2019) is able to meet the minimum number of vehicles for all 25 trip instances and finds 9(1) solutions with 1(2) more vehicles than the minimum number of vehicles for the large 125 trip instances. The average gap in weighted flow time over the instances where they are able to find the optimal number of vehicles is 4.1 %.

5.3 Effect of supplier centralization

So far, we have assumed that suppliers are randomly dispersed on the map. However, in reality, this is not necessarily the case. Especially in JIT industries, where short delivery cycles are key, suppliers often form clusters close to their customers, that is, so-called supplier parks (e.g., Larsson 2002; Pfohl and Gareis 2005). Such supplier parks are supposed to facilitate JIT logistics, but for the companies involved, relocating may be expensive and makes it impossible to benefit from wage level and taxation differences across country or regional borders. To investigate whether pooling suppliers in regional clusters is actually beneficial when employing direct deliveries, we conduct the following series of experiments.

We created 3600 additional instances with 100 trips each that can be divided into two classes of 1800 instances each.

First, we generate 100 *base instances* for each class and each number $|S| \in \{2, 4, 6\}$ of suppliers in the same fashion as before, except that the suppliers are placed more centrally in both classes, i.e., placed randomly at coordinates $(x_{S_i}, y_{S_i}) \in [15, 35] \times [15, 35]$, while customer coordinates (x_{C_i}, y_{C_i}) are taken randomly from $[1, 50] \times [1, 50]$, i.e., the suppliers' locations are somewhat optimized with respect to their customers. Additionally, we distinguish the `Random` instance class, where trips are assigned randomly to a supplier, and the `Optimized` class, where trips are assigned to the nearest supplier (regarding Euclidean distances) with a higher probability of $0.5 + \frac{1}{|S|}$.

For all created base instances and all values $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, we build additional instances to model the clustering of the suppliers. These instances differ only in the supplier coordinates: For all $i \in S$, we compute new coordinates $(x_{S_i}, y_{S_i}) := (\lceil(1-\alpha)x_{S_i} + \alpha x_{CM}\rceil, \lceil(1-\alpha)y_{S_i} + \alpha y_{CM}\rceil)$, where CM is the center of mass of all customer locations, i.e., $CM := (x_{CM}, y_{CM}) = (\lfloor\sum_{i\in I} x_{C_i}/|I|\rfloor, \lfloor\sum_{i\in I} y_{C_i}/|I|\rfloor)$. As a result, $\alpha = 0$ reproduces the base instance, $0 < \alpha < 1$ increases the concentration of suppliers, and $\alpha = 1$ has all suppliers at the same location. The procedure of concentrating suppli-

Fig. 2 Clustering of suppliers with values $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$ for a base instance with four suppliers

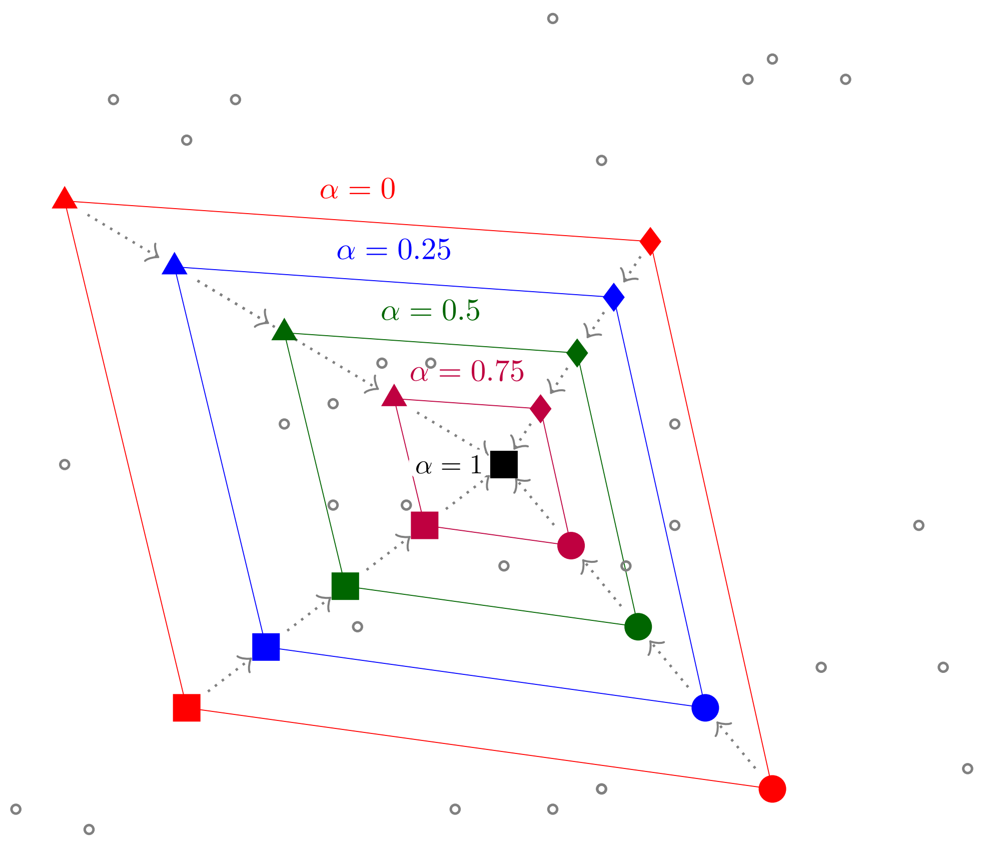

Table 5 Impact of supplier centralization on computation times

Class	α-Value						All
	0	0.2	0.4	0.6	0.8	1.0	
`Optimized`	107.0	102.0	62.3	57.5	54.6	11.0	65.7
`Random`	34.9	50.3	45.1	39.5	24.2	9.4	33.9
All	70.9	76.1	53.7	48.5	39.4	10.2	49.8

ers for a base instance with four suppliers is depicted in Fig. 2.

All of the 3600 instances were solved to optimality in less than 50 s on average. Table 5 shows the impact of supplier centralization on computation times. Therein, the average computation time for instances grouped according to their class and their α-value is reported. Note first that the computation times for the `Random` instances are smaller by almost factor two on average. Moreover, for larger α values, the average computation times decrease (with a single exception), which means that instances with more clustered suppliers are less difficult to solve.

Figure 3 shows how often all deliveries could be made with one fewer vehicle than in the baseline scenario ($\alpha = 0$), i.e., where all suppliers are located at their original (random) sites, and how often an extra vehicle was necessary to visit all customers compared with that scenario. Figure 4 visualizes the same information with regard to the weighted flow time objective.

While there is some expected random fluctuation, surprisingly, our tests reveal that supplier concentration negatively affects both fleet size and weighted flow time on average. The more centralized the suppliers are, the more the logistics performance suffers. The performance loss is particularly strong in the case of optimized customer–supplier assignments (class `Optimized`), because the advantage of matching customers with nearby suppliers is lost if all suppliers are in the same spot anyway. However, even in the `Random` class, more trucks are required and weighted flow time rises when suppliers are grouped together. At first glance, this seems counterintuitive, since the average distance between suppliers and customers drops when siting suppliers at the center of mass of their customers. However, the return trips become longer in many cases. Since trucks do not need to return to the same supplier from whence they set off, after visiting a far-distant customer, a truck may be able to return to a nearby supplier for its next trip as long as suppliers are widely dispersed. If suppliers are clustered in the same location, trucks must always make a full round trip back to the same area from which they came. This leads to the somewhat surprising conclusion that supplier parks can actually be detrimental to JIT logistics performance when direct shipping policies are used.

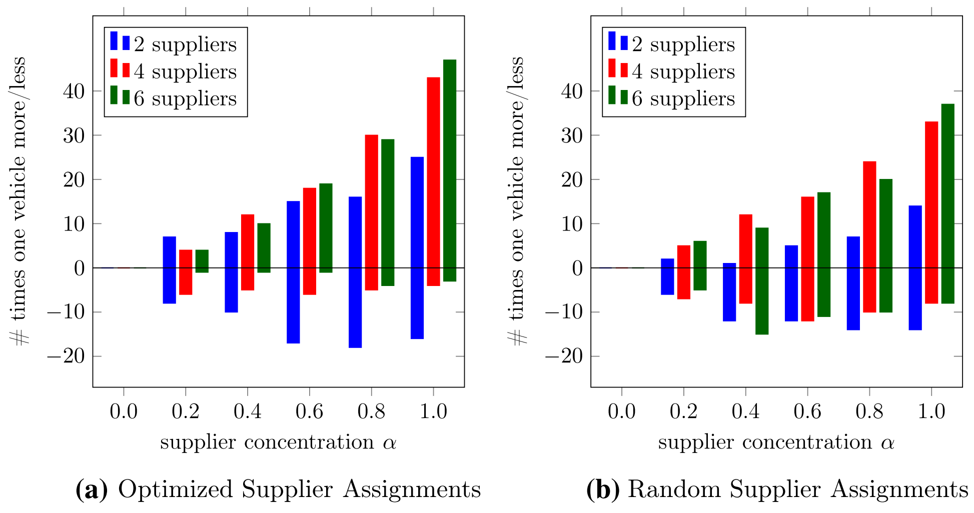

Fig. 3 Difference in number of vehicles

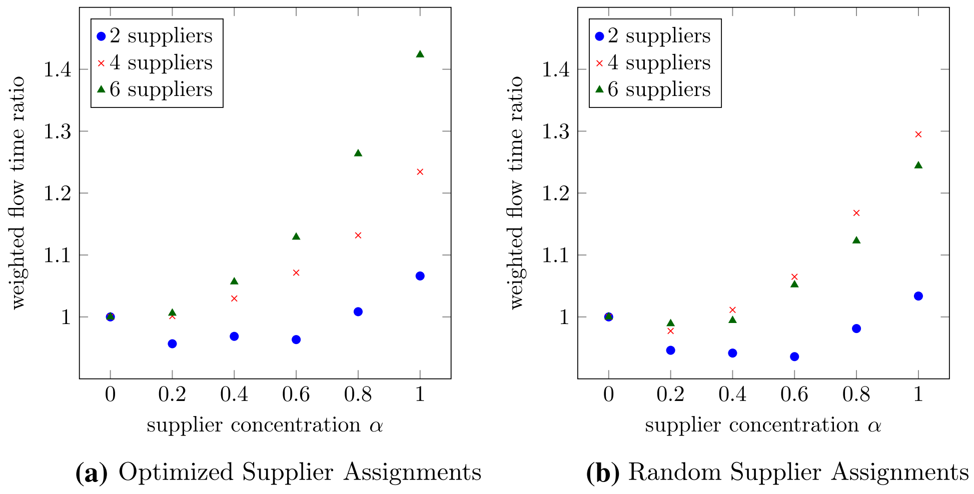

Fig. 4 Difference in weighted flow time

6 Conclusion and outlook

In this paper, we introduced the direct delivery scheduling problem in networks (DDSP-N), which is concerned with assigning and timetabling a given set of trips to vehicles such that the vehicle fleet is as small as possible and the customer waiting times are minimal. We adapted the compact model of Emde and Zehtabian (2019), presented a path-based formulation, and solved the latter via branch-cut-and-price (BCP). Our algorithm solved all instances from the literature in less than 1 min on average. On newly generated problems, BCP solved to optimality most DDSP-N instances with up to 100 customers in acceptable time. For larger data sets, it found tight bounds within 1 h of CPU time.

Our computational tests also revealed that, somewhat counterintuitively, the common just-in-time practice of gathering multiple suppliers in so-called supplier parks may not actually facilitate logistics operations in a direct shipping network. Of course, real-world just-in-time shipping does not consist only of direct deliveries; however, they do often make up a sizable share of total deliveries. For instance, Klug (2018) reports that about 30 % of parts are delivered directly via truck to the BMW plant in Leipzig (Germany). Given such figures, it is worth investigating whether supplier centralization is necessarily as helpful as is widely believed.

As point-to-point shipping is often used for high-velocity goods, where even slight delays can be problematic, future research should study the robustness of direct delivery networks by incorporating uncertain travel times into a stochastic model. Our model considers this only implicitly by minimizing waiting times. Moreover, industrial distribution networks often contain a mix of direct deliveries and other shipping strategies. Optimizing delivery policy and route choices in a mixed network is a worthwhile topic for future work.

Appendix: Notation

Sets	
$\mathcal{L}$	Set of physical locations
$\mathcal{S}$	Set of suppliers
$\mathcal{N}$	Set of customers
I	Set of trips
$\mathcal{A}$	Set of connections between physical locations
Attributes of a trip $i \in I$	
s_i	The supplier $s_i \in \mathcal{S}$ from where goods are shipped
n_i	The customer $n_i \in \mathcal{N}$ who receives these goods
r_i	Ready time of the trip
d_i	Deadline of the trip
p_i	Processing time of the trip
w_i	Weight of the trip
Parameters	
$\tau(n, s)$	Travel time between the physical location of customer n and supplier s
0	Start depot
$0'$	Destination depot
Variables in the compact formulation	
x_{ij}	Binary connection variable for trips $i, j \in I$
T_i	Starting time variable of trip i
Sets and parameters for the trip-based network	
V	Set of vertices in the trip-based network
A	Set of arcs in the trip-based network
$[e_v, \ell_v]$	Time window of vertex $v \in V$
p_v	Service time at vertex $v \in V$
$t_{vv'}$	Travel time between vertices $v, v' \in V$
Routes and their attributes	
Ω	Set of all DDSP-N routes
P	A DDSP-N route
w_P	Weighted flow time of route P
a_{vP}	Number of times vertex $v \in V$ is visited by route P
$\tilde{c}_P$	Reduced cost of path $P \in \Omega$
Variables and bounds in the set-partitioning formulation	
λ_P	Route variable
L, U	Lower/upper bound on the number of vehicles
Parameters in the pricing problem	
π_i	Dual price of the partitioning constraint (3b) for trip $i \in I$
μ_L, μ_U	Dual price of the lower/upper fleet size constraint (3c)
$\tilde{c}_{vv'}$	Reduced cost of arc $(v, v') \in A$
Resources in the pricing problem	
$(C_v, T_v, (S_v^i))$	A label at vertex $v \in V$
C_v	The accumulated reduced cost (including the weighted flow time)
T_v	The time at vertex v
(S_v^i)	Count of how often a trip $i \in I$ has already been performed

References

Achterberg, T. (2007). *Constraint integer programming*. Ph.D. thesis, Technische Universität Berlin, Fakultät II—Mathematik und Naturwissenschaften, Berlin, Germany.

Afzalirad, M., & Rezaeian, J. (2016). Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. *Computers & Industrial Engineering*, *98*, 40–52.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, *246*(2), 345–378.

Allahverdi, A., Ng, C., Cheng, T. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985–1032.

Angel-Bello, F., Cardona-Valdes, Y., & Alvarez, A. (2019). Mixed integer formulations for the multiple minimum latency problem. *Operational Research*, *19*(2), 369–398. https://doi.org/10.1007/s12351-017-0299-4.

Baldacci, R., Mingozzi, A., & Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, *59*(5), 1269–1283.

Barnes-Schuster, D., & Bassok, Y. (1997). Direct shipping and the dynamic single-depot/multi-retailer inventory system. *European Journal of Operational Research*, *101*(3), 509–518.

Boysen, N., Emde, S., Hoeck, M., & Kauderer, M. (2015). Part logistics in the automotive industry: Decision problems, literature review and research agenda. *European Journal of Operational Research*, *242*(1), 107–120.

Bulhoes, T., Sadykov, R., & Uchoa, E. (2018). A branch-and-price algorithm for the minimum latency problem. *Computers & Operations Research*, *93*(1), 66–78.

Carpaneto, G., Dell'Amico, M., Fischetti, M., & Toth, P. (1989). A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, *19*(5), 531–548.

Chen, L. (2008). Product & customer profiling for direct store delivery (DSD). Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Chen, Z.-L., & Powell, W. B. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, *11*(1), 78–94.

Chopra, S. (2003). Designing the distribution network in a supply chain. *Transportation Research Part E: Logistics and Transportation Review*, *39*(2), 123–140.

Chopra, S., & Meindl, P. (2015). *Supply chain management: Strategy, planning, and operation* (6th ed.). Upper Saddle River, NJ: Prentice Hall.

Chuah, K. H., & Yingling, J. C. (2005). Routing for a just-in-time supply pickup and delivery system. *Transportation Science*, *39*(3), 328–339.

De Angelis, V., Mecoli, M., Nikoi, C., & Storchi, G. (2007). Multiperiod integrated routing and scheduling of world food programme cargo planes in Angola. *Computers & Operations Research*, *34*(6), 1601–1615.

Dell'Amico, M., Fischetti, M., & Toth, P. (1993). Heuristic algorithms for the multiple depot vehicle scheduling problem. *Management Science*, *39*(1), 115–125.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., & Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic & G. Laporte (Eds.), *Fleet management and logistics, chapter 3* (pp. 57–93). Boston: Kluwer Academic Publisher.

Desaulniers, G., Desrosiers, J., & Solomon, M. (Eds.). (2005). *Column generation*. New York, NY: Springer.

Desaulniers, G., Madsen, O., & Ropke, S. (2014). The vehicle routing problem with time windows. In D. Vigo & P. Toth (Eds.), *Vehicle routing, chapter 5* (pp. 119–159). Philadelphia, PA: Society for Industrial and Applied Mathematics.

Du, T., Wang, F., & Lu, P.-Y. (2007). A real-time vehicle-dispatching system for consolidating milk runs. *Transportation Research Part E: Logistics and Transportation Review*, *43*(5), 565–577.

Emde, S., & Zehtabian, S. (2019). Scheduling direct deliveries with time windows to minimize truck fleet size and customer waiting times. *International Journal of Production Research*, *57*(5), 1315–1330.

Fakcharoenphol, J., Harrelson, C., & Rao, S. (2007). The k-traveling repairmen problem. *ACM Transactions on Algorithms (TALG)*, *3*(4), 40.

Gacias, B., Artigues, C., & Lopez, P. (2010). Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research*, *37*(12), 2141–2151.

Gallego, G., & Simchi-Levi, D. (1990). On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer r-systems. *Management Science*, *36*(2), 240–243.

Gaur, D. R., & Singh, R. R. (2017). A heuristic for cumulative vehicle routing using column generation. *Discrete Applied Mathematics*, *228*, 140–157.

Gélinas, S., Desrochers, M., Desrosiers, J., & Solomon, M. (1995). A new branching strategy for time constrained routing problems with applications to backhauling. *Annals of Operations Research*, *61*, 91–109.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of discrete mathematics* (Vol. 5, pp. 287–326). Amsterdam: Elsevier.

Hamzadayi, A., & Yildiz, G. (2017). Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. *Computers & Industrial Engineering*, *106*, 287–298.

Ioachim, I., Gélinas, S., Desrosiers, J., & Soumis, F. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, *31*, 193–204.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, *30*(1), 113–148.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, & M. Solomon (Eds.), *Column generation, chapter 2* (pp. 33–65). Berlin: Springer.

Jepsen, M., Petersen, B., Spoorendonk, S., & Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, *56*(2), 497–511.

Ke, L., & Feng, Z. (2013). A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, *40*(2), 633–638.

Klug, F. (2018). *Logistikmanagement in der Automobilindustrie: Grundlagen der Logistik im Automobilbau*. Heidelberg: Springer. https://doi.org/10.1007/978-3-662-55873-7.

Knott, R. (1987). The logistics of bulk relief supplies. *Disasters*, *11*(2), 113–115.

Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., & Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, *33*(1), 101–116.

Kulkarni, S., Krishnamoorthy, M., Ranade, A., Ernst, A. T., & Patil, R. (2018). A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem. *Transportation Research Part B: Methodological*, *118*, 457–487.

Lalla-Ruiz, E., & Voß, S. (2019). A popmusic approach for the multi-depot cumulative capacitated vehicle routing problem. *Optimization Letters,* 1–21.

Larsson, A. (2002). The development and regional significance of the automotive industry: Supplier parks in Western Europe. *International Journal of Urban and Regional Research*, *26*(4), 767–784.

Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In *Annals of discrete mathematics* (Vol. 1, pp. 343–362). Elsevier.

Lopes, M. J. P., & Valério de Carvalho, J. M. (2007). A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. *European Journal of Operational Research*, *176*(3), 1508–1527.

Luo, Z., Qin, H., & Lim, A. (2014). Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. *European Journal of Operational Research*, *234*(1), 49–60.

Lysgaard, J., & Wøhlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research*, *236*(3), 800–810.

Meyer, A., & Amberg, B. (2018). Transport concept selection considering supplier milk runs-an integrated model and a case study from the automotive industry. *Transportation Research Part E: Logistics and Transportation Review*, *113*, 147–169.

Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., & Moreno-Vega, J. M. (2016). A mixed integer formulation and an efficient metaheuristic procedure for the *k*-travelling repairmen problem. *Journal of the Operational Research Society*, *67*(8), 1121–1134.

Park, J., & Kim, B.-I. (2010). The school bus routing problem: A review. *European Journal of Operational Research*, *202*(2), 311–319.

Pessoa, A., Uchoa, E., de Aragão, M. P., & Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, *2*(3–4), 259–290.

Pfohl, H.-C., & Gareis, K. (2005). Supplier parks in the German automotive industry: A critical comparison with similar concepts. *International Journal of Physical Distribution & Logistics Management*, *35*(5), 302–317.

Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems*. Berlin: Springer.

Rauch, P., Gronalt, M., & Häuslmayer, H. (2007). Überregionales Logistik- und Versorgungsnetzwerk für Holz-Biomasse. *Schriftenreihe Berichte aus Energie- und Umweltforschung*, *51*.

Ribeiro, C. C., & Soumis, F. (1994). A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, *42*(1), 41–52.

Ribeiro, G. M., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, *39*(3), 728–735.

Righini, G., & Salani, M. (2006). Symmetry helps: Bounded bidirectional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, *3*(3), 255–273.

Rivera, J. C., Afsar, H. M., & Prins, C. (2016). Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem. *European Journal of Operational Research*, *249*(1), 93–104.

Sze, J. F., Salhi, S., & Wassan, N. (2017). The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. *Transportation Research Part B: Methodological*, *101*, 162–184.

Tanaka, S., & Araki, M. (2013). An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *Computers & Operations Research*, *40*(1), 344–352.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., & Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, *261*(2), 530–539.

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., & Meisel, F. (2018). Branch-and-price-and-cut for the active-passive vehicle-routing problem. *Transportation Science*, *52*, 300–319.

van Den Akker, J. M., Hoogeveen, J. A., & van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research*, *47*(6), 862–872.

Vigo, D., & Toth, P. (Eds.). (2014). *Vehicle routing*. Philadelphia, PA: MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics.

Yu, G., & Zhang, G. (2009). Scheduling with a minimum number of machines. *Operations Research Letters*, *37*(2), 97–101.