Annals of Operations Research
https://doi.org/10.1007/s10479-022-04891-1

ORIGINAL RESEARCH

Point-to-point and milk run delivery scheduling: models, complexity results, and algorithms based on Benders decomposition

Simon Emde[1] · **Shohre Zehtabian**[1] · **Yann Disser**[2]

Accepted: 20 July 2022

Abstract
We consider the problem of scheduling a set of direct deliveries between a depot and multiple customers using a given heterogeneous truck fleet. The trips have time windows and weights, and they should be completed as soon after release as possible (minimization of maximum weighted flow time). Moreover, some trips can optionally be combined in predefined milk runs (i.e., round trip tours), which need not be linear combinations of the constituent direct trips, accounting, e.g., for consolidation effects because the loading dock needs to be approached only once. This problem has applications, e.g., in just-in-time, humanitarian, and military logistics. We adapt a mixed-integer programming model from the literature to this problem and show that deciding feasibility is NP-complete in the strong sense on three levels: assigning trips to trucks, selecting milk runs, and scheduling trips on each individual truck. We also show that, despite this complexity, a state-of-the-art constraint programming solver and a problem-specific approach based on logic-based Benders decomposition can solve even large instances with up to 175 trips in many cases, while the mixed-integer programming model is essentially unsolvable using commercial optimization software. We also investigate the robustness of the maximum flow time objective in the face of unforeseen delays as well as the influence of milk runs.

Keywords Scheduling · Logistics · Benders decomposition · Direct deliveries · Maximum weighted flow time

✉ Simon Emde
siem@econ.au.dk

Shohre Zehtabian
szehtabian@econ.au.dk

Yann Disser
disser@mathematik.tu-darmstadt.de

[1] CORAL—Cluster for Operations Research, Analytics, and Logistics, Department of Economics and Business Economics, Aarhus University, Fuglesangs Allé 4, 8210 Århus V, Denmark

[2] Institut für Mathematik and Graduate School CE, Technische Universität Darmstadt, Darmstadt, Germany

Published online: 08 August 2022

1 Introduction

In many industries, parts and products are shipped directly between one source and multiple sinks. Using such a system, items are sent via a fleet of vehicles, each of which leaves the depot, goes to a single location, and returns to the depot to potentially set out again to another location. Vehicles may either take goods from the depot to a location, or pick goods up from a location to take to the depot, or both.

Direct distribution, where customer orders are not aggregated but are processed individually, is common in many industries. For instance, Queiser (2007) reports that direct shipping from either suppliers or intermediate storage facilities to OEM assembly plants is one of the most common delivery strategies for European and Asian automotive companies. Generally speaking, direct deliveries are proven to be the most cost-efficient distribution policy when the economic lot size of the customers is close to the vehicle capacity (Gallego & Simchi-Levi, 1990; Barnes-Schuster & Bassok, 1997). Moreover, this distribution strategy minimizes pipeline stock because there are no transshipment or detour delays.

More specifically, direct delivery strategies are common, for example, in retail supply chains (e.g., Lin et al., 2009), just-in-time logistics (e.g., Holweg & Miemczyk, 2003), and after-sales customer service (e.g., Kutanoglu & Mahajan, 2009). Other applications of point-to-point delivery discussed in the literature are, e.g., vendor managed inventory (Kleywegt et al., 2002; Li et al., 2008), military resupply in combat situations via drone (McCormack, 2014), and humanitarian logistics using airplanes (De Angelis et al., 2007). Recently, Lmariouh et al. (2019) describe the case of a Moroccan manufacturer of bottled water, shipping its products directly from a central plant to regional depots and wholesalers.

Direct deliveries can, however, be less efficient than multi-stage distribution strategies if vehicle capacities are poorly utilized. In practice, delivery networks therefore often include a mix of direct and milk run deliveries (Meyer & Amberg, 2018). Dispatchers must take care that resources not be wasted and deliveries be made speedily with a small vehicle fleet. In this context, this paper tackles the following problem. Given a set of transport requests, of which only a given subset can be combined into milk runs and most of which entail a single round trip to one customer that takes a given processing time, when should which trip be performed by what vehicle from a given heterogeneous truck fleet? Since customers cannot—or at least prefer not to—accept deliveries at any arbitrary time, each trip is associated with a time window, i.e., an earliest and latest execution time. As the objective, one of the main advantages of direct deliveries is their short transit time. To account for this, optimal schedules should avoid long response times as much as possible, i.e., we minimize the maximum flow time.

Note that the definition of a milk run in this context is different from a tour in a classic vehicle routing setting (VRP). VRP are typically concerned with less-than-truckload shipping, where many customers can lie on a route. Adding or removing a customer to/from a route hence typically affects the driving time on the route and the presence/absence of the fixed service time of the specific customer. Since direct delivery networks are mostly relevant for full or almost-full truckloads, milk runs allow for the consideration of consolidation effects: e.g., if a larger truck is used to combine two shipments, this truck may only have to approach the loading ramp once and only have to check-in at the customer site once (if both shipments have the same destination), thus shortening the trip duration beyond mere savings in driving time. Predefining such possible milk runs may hence be helpful for almost-full-truckload shipping, while for less-than-truckload shipping, it becomes impracticable.

In their seminal papers, Emde and Zehtabian (2019) and Gschwind et al. (2020) introduce the *direct deliveries scheduling problem* (DDSP). The authors propose a MIP model, a branch-cut-and-price algorithm as well as two heuristics. We adapt this problem by considering a given heterogeneous vehicle fleet and the possibility of combining trips into milk runs. Previous studies have considered minimizing the total weighted flow time (or response time) as the objective. However, this may lead to unfair schedules, because individual customers are not protected from excessively slow response times (Anand et al., 2017). Consider for example a problem where three round trips A, B, and C must be scheduled on two different trucks 1 and 2. Assume that trips A and B are released at time zero (i.e., a truck processing these trips can depart from the depot immediately at the beginning of the planning horizon) and trip C is released at time one, all deadlines are four time units later than the release date (i.e., the trucks must return to the depot no later than four time units after the release of the trip they are processing), and the processing times are three, one, and one time units, respectively, on truck 1, while truck 2 is slower and takes an additional time unit of processing time per trip. Moreover, assume that all trips have the same priority, i.e., weight one. Figure 1a shows the solution that minimizes the total flow time. Note that while trips B and C are completed early in their time windows, trip A bumps up against its deadline. Figure 1b, on the other hand, depicts the schedule that minimizes the maximum flow time. In this solution, trips B and C complete a little later, albeit still ahead of their respective deadlines, but trip A also finishes sooner, which may be considered fairer (no customer must wait for more than three time units for their order to be delivered). Moreover, in Fig. 1b, a delay of one time unit would not make any job tardy, unlike in the solution in Fig. 1a, where trip A must not be delayed at all.

Consequently, we adapt the objective function to avoid extreme flow times, and investigate in a numerical study what effect different objectives have on delays in case of unforeseen disturbances. Moreover, we account for milk runs and a heterogeneous vehicle fleet. We dub this problem version *DDSP-het*. We show that DDSP-het is computationally challenging on multiple levels and propose a mixed-integer linear programming model, a constraint programming model, and exact solution methods based on Benders decomposition for this novel problem version, which we enhance with combinatorial cuts, valid inequalities, and a specialized subproblem solution method. Our computational tests show that this procedure solves DDSP-het to optimality or near-optimality quite quickly, clearly outperforming default solvers in many cases.

The rest of this paper is structured as follows. In Sect. 2, we review the pertinent literature. Section 3 formalizes the problem, introduces a MIP and a CP model, and discusses computational complexity. In Sect. 4, we present a logic-based Benders decomposition procedure

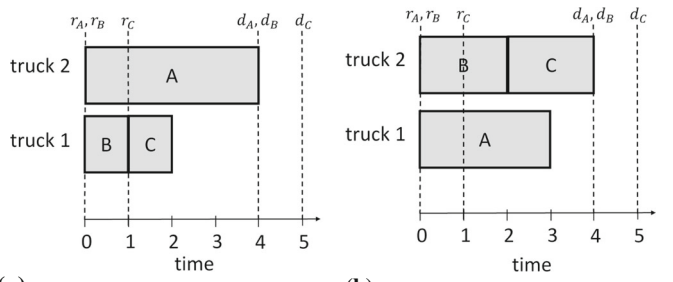


Fig. 1 Gantt charts of two different solutions for the introductory problem with three trips on two trucks

to solve DDSP-het, which we test in Sect. 5, where we also investigate the robustness of DDSP-het solutions and the influence of milk runs. Finally, Sect. 6 concludes the paper.

2 Literature review

Generally speaking, DDSP can be viewed as a machine scheduling problem (surveyed by Pinedo, 2015), where vehicles are parallel machines and trips are jobs. Since the truck fleet is heterogeneous and vehicles can process trips at different speeds, the machines are unrelated. In classic machine scheduling notation (Graham et al., 1979), this is similar to the tuple $[R|r_j, d_j|\max w_j F_j]$. However, there is a fundamental difference in that some trips can be combined into milk runs, whose processing time need neither be the sum of the individual processing times, nor their maximum, thus not conforming to typical batching machine scheduling models (surveyed by Potts & Kovalyov, 2000; Mathirajan & Sivakumar, 2006; Mönch et al., 2011). To the best of our knowledge, this problem has not been solved before. We refer to this problem by the tuple $[R|r_j, d_j, \text{batch}(\mathcal{B})|\max w_j F_j]$, where batch($\mathcal{B}$) indicates that some jobs can be batched into milk runs, whose properties are explicitly defined and need not be either the sum (i.e., serial batching) or the maximum (i.e., parallel batching) of the constituent trips.

Possible (or impossible) combinations of trips can also be expressed through conflict graphs (e.g., Kowalczyk & Leus, 2017) or as tours in a classic vehicle routing problem (VRP, surveyed by Toth & Vigo, 2014). However, milk runs are significantly more flexible than what a conflict graph or the VRP can consider. E.g., it is possible to allow the consolidation of three trips A, B, and C in a milk run, but not any pairwise combination of these trips. This may be relevant, e.g., if A, B, and C are transport requests headed for three facilities of the same customer (e.g., an OEM). The customer may be fine with either individual direct deliveries to the destination facilities or, alternatively, with one complete consolidated shipment to a central receiving location, but not with partially consolidated shipments. Moreover, the processing time of a milk run need not be related to the driving times of the individual trips. E.g., a consolidated shipment may go to a central receiving store, which is different from the individual direct destinations, or the truck need only be loaded once, which saves on service time in a non-linear manner.

Regarding specific applications, direct delivery (or point-to-point) networks are commonly used in industries where high-velocity, high-bulk, just-in-time, or specialty goods need to be shipped by truck. In these cases, "vehicles" are trucks, which process transport jobs by carrying cargo from a source (e.g., supplier plant, cross-dock, or distribution center) to customers (e.g., OEM plants, pool points). For instance, direct deliveries are particularly common in just-in-time logistics (Boysen et al., 2015) and grocery supply chains (Chen, 2008).

While there are some papers looking at direct deliveries from a strategic perspective (e.g., Gallego & Simchi-Levi, 1990; Barnes-Schuster & Bassok, 1997), to the best of our knowledge the only other papers explicitly dealing with direct deliveries from an operational perspective, where a given set of trips must be assigned to truck fleet, are Emde and Zehtabian (2019) and Gschwind et al. (2020). Apart from introducing the problem, the authors' main contributions are two heuristic procedures and an exact branch-cut-and-price algorithm, which are shown to solve large instances (with up to 125 trips) within a time limit of up to one hour of CPU time. The solution methods from these previous papers are not immediately applicable to

DDSP-het, however, because of the different objective functions, heterogeneous truck fleet, and inclusion of milk runs.

The problem of assigning trips to a fleet of vehicles also bears some resemblance to classic vehicle routing. DDSP, however, is a scheduling, not a routing problem. Even combining trips into milk runs in this context bears little resemblance to classic VRPs because the processing times and time windows of the milk runs may not be a linear combination of the constituent trips. In a sense, DDSP-het hence sits somewhere between a scheduling and a routing problem; see also Beck et al. (2003) for a discussion about the relationship between routing with time windows and job shop scheduling.

The most closely related family of routing problems may be the full-truckload vehicle routing problem (FTVRP), which is also concerned with assigning trips between origin and destination pairs to a fleet of (possibly heterogeneous) vehicles. Since the seminal work by Ball et al. (1983), many papers have been published in this stream of research (reviewed by Annouch et al., 2016). The FTVRP, however, is mostly considered in the context of shuttling containers between multiple terminals over longer time horizons (i.e., multiple shifts), often necessitating some type of shift scheduling to be included in the model (e.g., Bai et al., 2015; Xue et al., 2021). Moreover, the objective of FTVRP in most cases involves cost minimization or profit maximization, sometimes minimization of total travel distance or deadheading. To the best of our knowledge, customer response times have never been considered, which is not surprising since FTVRP is typically formulated in the context of container movements, not just-in-time direct deliveries. Consequently, FTVRP often includes a decision as to which transport requests to accept in the first place, i.e., some trips may not be performed at all, which is not possible in DDSP-het. Finally, the concept of a milk run, although common in, e.g., part logistics, has no equivalent in the FTVRP.

Vehicle scheduling problems (Bodin & Golden, 1981; Bunte & Kliewer, 2009), on the other hand, concern themselves with scheduling already timetabled trips and are often used in the context of public transportation. DDSP trips, however, are not timetabled, they merely have time windows. The most similar model from this general stream of literature may be Tzur and Drezner (2011), who also consider scheduling direct deliveries between a depot and multiple customers. The authors dub their problem the *assignment and scheduling of transportation tasks to vehicles (ASTV)*, which deals with less-than-truckload shipping, i.e., loading constraints on the homogeneous truck fleet must also be satisfied. The objective is to minimize the total cost for vehicles, driving time, and time spent waiting during loading and unloading of commodities. The problem is solved with a decomposition heuristic.

3 Problem description

The direct deliveries scheduling problem with a heterogeneous fleet (DDSP-het) can be described as follows. Given is a set of direct deliveries that have to be performed during the planning horizon (e.g., one day) from one depot to any number of customers by a given heterogeneous fleet of trucks. Each trip takes a certain processing time, which includes the time to load the truck at the depot, drive to the customer, unload, and drive back to the depot, plus any break, refuelling, and/or refitting times, if applicable. Since the truck fleet is heterogeneous, certain trucks may be slower than others or possibly not capable of certain deliveries at all, e.g., for lack of capacity (e.g., Al Theeb et al., 2019). Note that a trip may of course also entail picking up goods from the customer to be transported back to the depot. Moreover, while we mostly consider direct delivery round trips, it may be possible to

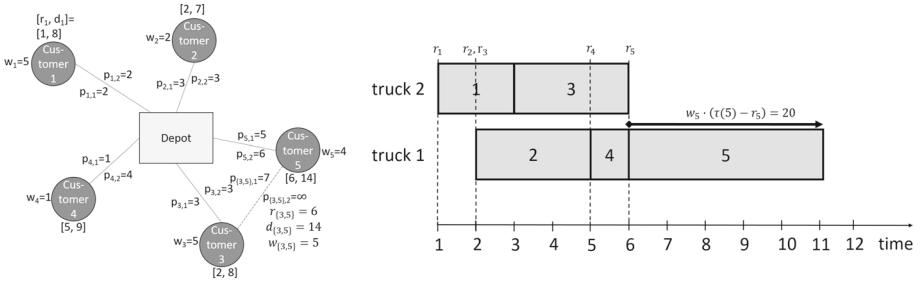
optionally combine some trips into milk runs, where a vehicle makes multiple deliveries in the same run, if the time windows, truck capacities and other considerations allow it. Such a milk run is presumably faster than processing the constituent trips individually, although the decrease in processing time may be different from the mere savings in driving time, because some other steps can also be combined (e.g., the truck needs to approach the loading dock at the depot only once for all combined trips etc.).

Usually, customers will set time windows for deliveries. For instance, some perishable items like baked goods or fresh produce must be delivered such that they can be sold immediately (e.g., Hsu et al., 2007), while OEM manufacturing plants may have tight just-in-time policies (Boysen et al., 2015). As the objective, response times are often considered central (e.g., Hong & Park, 1999). Apart from the immediate effect on customer satisfaction, this also carries the added benefit of making schedules more robust in the face of unplanned disturbances (e.g., unexpectedly heavy traffic). We investigate this further in Sect. 5.2.3.

3.1 Formal description

Let $I = \{1, \ldots, n\}$ be the set of trips from the depot to one customer each, and let the set of different truck types be $C = \{1, \ldots, c\}$. Each trip $i \in I$ takes a processing time of p_{ic} if it is processed by a truck of type $c \in C$, which includes the time to load the vehicle in the depot, drive to the customer, unload the vehicle, and drive back to the depot. This time may differ depending on such things as the size of the truck (heavier and/or older trucks tend to be slower). Some trips may be incompatible with certain truck types altogether (e.g., if the truck capacity is too low), in which case the corresponding processing time can be set to a prohibitively large value ($p_{ic} = \infty$). Moreover, each trip is associated with an earliest start time r_i and a latest completion time d_i, respectively, depending on the customers' time windows. Customer service should be completed as soon as possible within the time window. Consequently, we minimize the maximum weighted difference between completion time and release date over all trips. Of each truck class $c \in C$, there are $\gamma_c \in \mathbb{N}_{>0}$ trucks available, such that the total size of the vehicle fleet is $m = \sum_{c \in C} \gamma_c$. Finally, let $\mathcal{B} \subset \wp(I)$ the set of all subsets of trips that can be combined in a milk run, where $\wp(I)$ denotes the powerset of I. To give an example, if all pairs of trips (and only those) can be combined into milk runs, then $\mathcal{B} = \{\{i, i'\} \mid i, i' \in I, i < i'\}$. If assigned to the same truck, the trips $i \in B$ ($B \in \mathcal{B}$) can be combined into one milk run tour with processing time p_{Bc} for each truck class $c \in C$, release date r_B, deadline d_B, and weight w_B. Note that we do not make any assumptions regarding the parameters of the combined trip, although we would typically expect $p_{Bc} < \sum_{i \in B} p_{ic}$ for at least one $c \in C$. Also note that this implies that the customer sequence inside a milk run is fixed in preprocessing, presumably such that the total driving time is minimal. Other factors that may influence the processing time of a milk run may also be, among other things, the expected registration and waiting time at the customer(s) and the loading time at the depot, which may be shorter if the truck need only approach the loading dock once.

A schedule for DDSP-het is defined by sets $R_1, \ldots, R_m$, where set $R_k \subseteq I \cup \mathcal{B}$ is the set of trips and milk runs processed by truck $k \in \{1, \ldots, m\}$, and a permutation π_k of R_k, denoting the order in which the trips and milk runs are processed, i.e., $\pi_k(j) \in R_k$ denotes the j-th trip or milk run, $j = 1, \ldots, |R_k|$, to be processed by truck k. Note that each set R_k may contain both integers (trips) as well as sets of integers (milk runs combining multiple trips). For ease of notation, let $\eta(k) \in C$ denote the truck class of truck $k \in \{1, \ldots, m\}$, and



(a) Example data. **(b)** Example solution.

Fig. 2 Example problem and Gantt chart of the optimal solution

let $\tilde{I}(i) = \{B \in \mathcal{B} \mid i \in B\}$ be the set of milk runs that contain trip $i \in I$. A feasible schedule has the following properties.

- Each trip $i \in I$ must be assigned exactly once, either on its own or as part of a milk run, i.e., either $\exists! k \in \{1, \ldots, m\} : i \in R_k$ or $\sum_{k=1}^{m} |R_k \cap \tilde{I}(i)| = 1$ must hold.
- Let $\tau(i)$ be the completion time of trip or milk run i. Then $\tau(\pi_k(j)), \forall j = 1, \ldots, |R_k|$, can be calculated recursively as

$$\tau(\pi_k(j)) = \begin{cases} \max\{r_{\pi_k(j)}; \tau(\pi_k(j-1))\} + p_{\pi_k(j)\eta(k)} & \text{if } 2 \le j \le |R_k| \\ r_{\pi_k(j)} + p_{\pi_k(j)\eta(k)} & \text{if } j = 1 \end{cases}.$$

- Each trip is processed within its time window, i.e., for all $i \in R_k, k = 1, \ldots, m$, it must hold that $r_i + p_{i\eta(k)} \le \tau(i) \le d_i$.

The goal of the optimization is minimizing response times, that is, the difference between trip completion time and release date, should be minimal. Emde and Zehtabian (2019) and Gschwind et al. (2020) propose to minimize the total sum of weighted flow times; however, this does not preclude individual customers from having long response times. Consequently, among all feasible schedules, we seek one where the maximum weighted flow time

$$Z = \max_{i \in \bigcup_{k=1}^{m} R_k} \{w_i \cdot (\tau(i) - r_i)\}$$

is minimal.

Example Consider the example problem given in Fig. 2a. In this example, there are 5 trips to be made by two trucks of different types ($C = \{1, 2\}$, $\gamma(1) = \gamma(2) = 1$). Moreover, trips 3 and 5 can optionally be combined in a milk run, i.e., $\mathcal{B} = \{\{3, 5\}\}$. A feasible and optimal solution consists of the two vehicles performing 3 and 2 trips, respectively, i.e., $R_1 = \{2, 4, 5\}$ and $R_2 = \{1, 3\}$, in order $\pi_1 = \langle 2, 4, 5\rangle$ and $\pi_2 = \langle 1, 3\rangle$. The schedule is depicted as a Gantt chart in Fig. 2b. Using this schedule, trip 5 has a flow time of $w_5 \cdot (\tau(5) - r_5) = 4 \cdot (11 - 6) = 20 = Z$. Note that in this example, it is not optimal to combine trips 3 and 5, even though that would have been possible for truck 1 and led to a processing time shorter than the sum of both. Also note that truck 2 is incapable of processing this milk run altogether, possibly due to its capacity being too low.

Table 1 Notation for the MIP model

$\hat{I}$	Set of trips and milk runs (index $i \in \hat{I} = I \cup \mathcal{B}$)
$\hat{n}$	Total number of trips and milk runs ($\hat{n} = \lvert\hat{I}\rvert$)
$\tilde{I}(i)$	Set of milk runs that include trip i ($\tilde{I}(i) = \{B \in \mathcal{B} \mid i \in B\}$)
C	Set of truck classes
M	Big integer, $M = \max_{i \in \hat{I}}\{d_i\}$
$\gamma(c)$	Number of trucks in the fleet of type $c \in C$
p_{ic}	Processing time of trip or milk run i on truck type c
r_i	Earliest start time of trip or milk run i
d_i	Latest completion time of trip or milk run i
w_i	Weight of trip or milk run i
$x^c_{ii'}$	Binary variable: 1, if trip or milk run i' is the direct successor of trip or milk run i on the same vehicle of type c; in case of $x^c_{0,i} = 1$ ($x^c_{i,\hat{n}+1} = 1$), i is the first (last) trip or milk run of a vehicle of type c; 0, otherwise
y_i	Binary variable: 1, if trip or milk run i is processed; 0, otherwise
t_i	Continuous variable: completion time of trip or milk run i

3.2 Mathematical models

To enable the use of default solvers, using the notation from Table 1, DDSP-het can be expressed by the following MIP model [DDSP-OP], adapted from Emde and Zehtabian (2019). To concisely model the problem, we introduce set $\hat{I} = I \cup \mathcal{B}$ of all trips and milk runs as well as $\hat{n} = \lvert\hat{I}\rvert$ as the total number of trips and milk runs.

$$\text{[DDSP-OP] Minimize } \mathcal{Z}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) = \max_{i \in \hat{I}}\{w_i \cdot (t_i - r_i)\} \tag{1}$$

subject to

$$\sum_{i' \in \hat{I} \cup \{\hat{n}+1\}} \sum_{c \in C} x^c_{ii'} = y_i \qquad \forall i \in \hat{I} \tag{2}$$

$$\sum_{i \in \hat{I} \cup \{0\}} \sum_{c \in C} x^c_{ii'} = y_{i'} \qquad \forall i' \in \hat{I} \tag{3}$$

$$\sum_{i \in \hat{I} \cup \{0\}} x^c_{ii'} = \sum_{i \in \hat{I} \cup \{\hat{n}+1\}} x^c_{i'i} \qquad \forall i' \in \hat{I}, c \in C \tag{4}$$

$$\sum_{i' \in \hat{I}} x^c_{0i'} \leq \gamma_c \qquad \forall c \in C \tag{5}$$

$$\sum_{i' \in \tilde{I}(i)} y_{i'} + y_i = 1 \qquad \forall i \in I \tag{6}$$

$$\left(1 - x^c_{ii'}\right) \cdot M + t_{i'} \geq t_i + p_{i'c} \qquad \forall i, i' \in \hat{I}, c \in C \tag{7}$$

$$t_i \leq d_i \qquad \forall i \in \hat{I} \tag{8}$$

$$r_{i'} + \sum_{c \in C} \sum_{i \in \hat{I} \cup \{0\}} p_{i'c} \cdot x^c_{ii'} \leq t_{i'} \qquad \forall i' \in \hat{I} \tag{9}$$

$$x^c_{ii'} \in \{0;1\} \qquad \forall i \in \hat{I} \cup \{0\}, i' \in \hat{I} \cup \{\hat{n}+1\}, c \in C \tag{10}$$

$$y_i \in \{0;1\} \qquad \forall i \in \hat{I} \tag{11}$$

Objective function (1) minimizes the maximum weighted flow time. Note that we linearize the objective function by introducing an auxiliary variable α and a set of auxiliary constraints $\alpha \geq w_i \cdot (t_i - r_i), \forall i \in \hat{I}$. Constraints (2) ensure that each trip i has either one successor (if the trip or milk run is processed) or no successor (if the trip or milk run is not processed). If that successor is dummy trip $\hat{n}+1$, then trip i is the last one performed by the respective vehicle. Constraints (3) have the same effect for the predecessor. Analogously, if $x^c_{0,i'} = 1$, trip or milk run i' is the first trip of a vehicle of type c. Equations (4) ensure that contiguous tours are constructed. Inequalities (5) make it impossible to use more trucks than are available. Constraints (6) enforce that every trip must take place, either as an individual trip or as part of a milk run. Constraints (7) assign each trip a non-overlapping processing time window, while (8) and (9) enforce the time windows. Finally, the domain of the variables is defined by (10) and (11). Note that we can implicitly assume that $t_i \geq 0, \forall i \in \hat{I}$, provided that all release dates are non-negative because of Constraints (9).

Given that many scheduling problems are solved more efficiently by constraint programming solvers than MIP solvers (e.g., Grimes et al. 2009; Malapert et al., 2012), we also formulate a constraint programming (CP) model for DDSP-het. Let ω_i be an optional interval variable, denoting the interval during which trip or milk run $i \in \hat{I}$ is processed. Let δ_{ik} be an optional interval variable denoting the interval of size $p_{i\eta(k)}$ during which trip or milk run i is processed by truck k, which cannot start sooner than r_i and cannot end later than d_i. Note that all interval variables are optional, i.e., it is part of the problem to decide whether a given milk run should be processed at all, and if yes, on what truck. Finally, let κ_k be a sequence variable denoting the order in which the trips assigned to truck k are processed. We get the following CP model.

$$\text{[DDSP-CP] Minimize } \max_{i \in \hat{I}} \{w_i \cdot (\text{endOf}(\omega_i) - r_i)\} \tag{12}$$

subject to

$$\text{noOverlap}(\kappa_k) \qquad \forall k = 1, \ldots, m \tag{13}$$

$$\text{alternative}(\omega_i, \{\delta_{ik} \mid k = 1, \ldots, m\}) \qquad \forall i \in \hat{I} \tag{14}$$

$$\text{presenceOf}(\omega_B) = 1 \Rightarrow \sum_{i' \in B} \sum_{\substack{i'' \in \tilde{I}(i') \cup \{i'\}: \\ i'' \neq B}} \text{presenceOf}(\omega_{i''}) = 0 \qquad \forall B \in \mathcal{B} \tag{15}$$

$$\text{presenceOf}(\omega_i) = 0 \Rightarrow \sum_{B \in \tilde{I}(i)} \text{presenceOf}(\omega_B) = 1 \qquad \forall i \in I \tag{16}$$

Objective (12) minimizes the maximum weighted flow time, where endOf(ω_i) denotes the end of interval ω_i if trip or milk run i is processed (zero, otherwise). Constraints (13) ensure that the trip intervals do not overlap on any truck, while (14) enforce that if a trip or milk run i is processed (i.e., ω_i is present), then it must be scheduled on exactly one truck (i.e., one interval δ_{ik} on exactly one truck k must be present). According to implications (15), if a milk run B is present (i.e., presenceOf(ω_B) $= 1$), then no other milk run containing a trip $i \in B$ or the trips in B themselves can be present. Finally, (16) imply that if trip $i \in I$

is not present, then exactly one milk run containing that trip must be present. Together, (15) and (16) enforce that each trip is processed exactly once, be it by itself or as part of a milk run.

3.3 Time complexity

Regarding worst-case time complexity, it is hard to solve DDSP-het even to feasibility, as we show below. Note that this distinguishes DDSP-het from the direct delivery scheduling problems discussed in Emde and Zehtabian (2019) and Gschwind et al. (2020), for which it is trivial to find a feasible solution.

Proposition 3.1 *Deciding feasibility for DDSP-het is NP-complete in the strong sense, even if* $m = 1$, $\mathcal{B} = \emptyset$.

Proof In this setting, deciding feasibility is equivalent to a single machine scheduling problem with time windows ($[1|r_j|L_{\max}]$), which is well-known to be NP-hard in the strong sense (Lenstra et al., 1977). NP-completeness follows from the fact that solutions to DDSP-het can be checked in polynomial time. □

It is not only the time windows that make DDSP-het hard to solve, however. Even if all trips are released at time 0 and there is a common deadline, deciding feasibility remains hard.

Proposition 3.2 *Deciding feasibility for DDSP-het is NP-complete in the strong sense, even if* $|C| = 1$, $\mathcal{B} = \emptyset$, *and* $r_i = 0$, $d_i = d$ *for all* $i \in I$ *and some* $d \in \mathbb{N}$.

Proof We show NP-hardness by reduction from BinPacking, which is well-known to be NP-hard in the strong sense (Garey & Johnson, 1979). An instance of BinPacking is defined as follows. Given q positive integers $a_1, \ldots, a_q$ and two positive integers $s, U \in \mathbb{N}$, does there exist a partition of the set $\{1, 2, \ldots, q\}$ into s sets $\{A_1, A_2, \ldots, A_s\}$, such that $\sum_{i \in A_j} a_i \leq U$ for all $j \in \{1, \ldots, s\}$?

An instance $(\{a_1, \ldots, a_q\}, s, U)$ of BinPacking is transformed into an instance of DDSP-het as follows. We set $C = \{1\}$, $\gamma_1 = s$, $\mathcal{B} = \emptyset$, and $I = \{1, \ldots, q\}$. Each trip $i \in I$ is released at time $r_i = 0$ and has processing time $p_{i,1} = a_i$. The common deadline of all trips $i \in I$ is $d_i = d = U$. This construction can be carried out in linear time. It is easy to see that every solution $\{A_1, A_2, \ldots, A_s\}$ to the BinPacking instance corresponds to a solution $\{R_1, \ldots, R_s\}$ with $A_k = R_k$ for all $k \in \{1, \ldots, s\}$, and vice-versa, irrespective of the permutations π_k. NP-completeness follows from the fact that solutions to DDSP-het can be checked in polynomial time. □

If there are multiple truck classes, deciding feasibility remains hard even if there are only two types of time windows, and all processing times are either 2 or ∞.

Proposition 3.3 *Deciding feasibility for DDSP-het is NP-complete in the strong sense, even if* $\mathcal{B} = \emptyset$, $p_{ic} \in \{2, \infty\}$ *for all* $i \in I$ *and* $c \in C$, *and* $[r_i, d_i] \in \{[0, 4], [1, 3]\}$ *for all* $i \in I$.

Proof We reduce from the strongly NP-hard 3Sat problem with clauses $C_1, \ldots, C_q$ over a set of variables $\{x_0, \ldots, x_{s-1}\}$, where s denotes the number of variables. We may assume that every variable appears exactly two times positively and exactly two times negatively (2P2N-3Sat or (3, 2B)- Sat, Berman et al., 2007). Given such an instance, we construct the following DDSP-het instance (cf. Fig. 3).

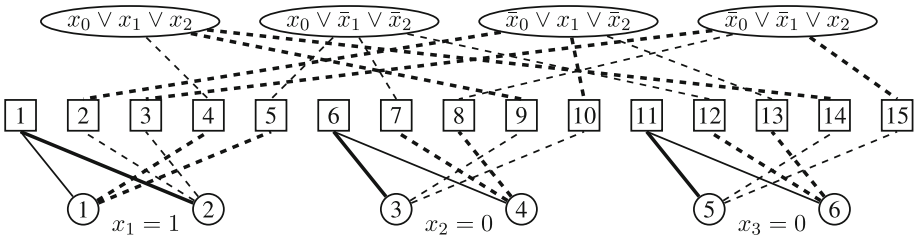


Fig. 3 Example of the reduction from 2P2N-3SAT to DDSP-het with $q = 4$ clauses and $s = 3$ variables in the proof of Proposition 3.3. Square vertices represent the set of trips I, round vertices represent the truck classes C. Each truck can either serve two trips connected to it by a dashed edge or a single trip connected to it by a solid edge. Bold edges indicate a satisfying assignment

We introduce truck classes $C = \{1, \ldots, 2s + q\}$ with $\gamma_c = 1$ truck in each class $c \in C$. We further set $\mathcal{B} = \emptyset$ and $I = \{1, \ldots, 5s\}$. All processing times p_{ic} are taken from $\{2, \infty\}$, and it suffices to specify the set $P_i = \{c \in C : p_{ic} = 2\}$ of truck classes that can serve each trip $i \in I$. For every variable x_i, we set $P_{5i+1} = \{2i + 1, 2i + 2\}$ and $[r_{5i+1}, d_{5i+1}] = [1, 3]$. This means that the unique truck of class $2i + 1$ or $2i + 2$ must be used to serve this trip, and we interpret this decision as setting variable x_i to *false* or to *true*, respectively. Note that, because of $p_{ic} \geq 2$ and $d_i \leq 4$ for all $i \in I$, the truck assigned to serving trip $5i + 1$ cannot serve any other trips.

The trips $5i + 2, 5i + 3$ correspond to the positive literal of variable x_i, and the trips $5i + 4$ and $5i + 5$ correspond to its negative literal, for every $i \in I$. Let $j_{i,1}$ and $j_{i,2}$ denote the indices of the (exactly) two clauses containing the positive literal x_i, and let $\bar{j}_{i,1}$, $\bar{j}_{i,2}$ denote the indices of the (exactly) two clauses containing the negative literal $\bar{x}_i$. We set $P_{5i+2} = \{2i + 2, 2s + \bar{j}_{i,1}\}$, $P_{5i+3} = \{2i + 2, 2s + \bar{j}_{i,2}\}$, $P_{5i+4} = \{2i + 1, 2s + j_{i,1}\}$, $P_{5i+5} = \{2i + 1, 2s + j_{i,2}\}$, and $[r_j, d_j] = [0, 4]$ for all $j \in \{5i + 2, \ldots, 5i + 5\}$.

Our construction ensures that whenever a variable x_i is set to true, the truck of class $2i + 2$ is free to serve the trips $5i + 2$ and $5i + 3$, while the trips $5i + 4$ and $5i + 5$, corresponding to the negative literal $\bar{x}_i$, need to be served by the trucks of classes $\{2s + 1, \ldots, 2s + q\}$. Conversely, if x_i is set to false, the truck of class $2i + 1$ is free to serve the trips $5i + 4$ and $5i + 5$, while the trips $5i + 2$ and $5i + 3$, corresponding to the positive literal x_i, need to be served by the trucks of classes $\{2s + 1, \ldots, 2s + q\}$. Every truck $2s + j$ of these classes corresponds to a clause C_j and is able to serve any (up to) two trips corresponding to the *negations (!)* of literals that appear in clause C_j. In other words, every clause can absorb up to two wrong literals, but not all three. Since each trip corresponding to a literal can be served by exactly one truck, this implies that at least one variable needs to be set correctly per clause.

With this interpretation, it is easy to verify that the constructed instance of DDSP-het is feasible if and only if the original 2P2N- 3SAT instance has a satisfying assignment. Observe that our construction can be carried out in polynomial time. NP-completeness follows from the fact that solutions to DDSP-het can be checked in polynomial time. □

The above results hold even if there are no milk run trips at all. As soon as milk runs are considered, deciding feasibility becomes hard even in the absence of time windows and if the truck fleet size is not limiting.

Proposition 3.4 *Deciding feasibility for DDSP-het is NP-complete in the strong sense, even if* $|C| = 1$, $\gamma_c = \infty$, $p_{B,1} = 0$ *and* $[r_B, d_B] = [0, \infty]$ *for all* $B \in \mathcal{B}$.

Proof We reduce from the strongly NP-hard 3SAT problem with clauses $C_1, \ldots, C_q$ over a set of variables $\{x_1, \ldots, x_s\}$. We again assume, without loss of generality, that every variable appears exactly two times positively and exactly two times negatively (Berman et al., 2007). Given such an instance, we construct the following DDSP-het instance.

We set $I = \{1, \ldots, s+q\}, C = \{1\}$, and $\gamma_1 = \infty$. For $i \in \{1, \ldots, s\}$, let $\mathcal{C}[i] \subseteq \{1, \ldots, q\}$ denote the set of indices of all clauses that contain the literal x_i, and let $\bar{\mathcal{C}}[i] \subseteq \{1, \ldots, q\}$ denote the set of indices of all clauses that contain the literal $\bar{x}_i$. We define $B_i = \{\{q+i\} \cup \mathcal{C} : \mathcal{C} \subseteq \mathcal{C}[i] \vee \mathcal{C} \subseteq \bar{\mathcal{C}}[i])\}$ for $i \in \{1, \ldots, s\}$ and let the set of milk runs be $\mathcal{B} = \bigcup_{i=1}^{s} B_i$.[1] Finally, we set $p_{i,1} = \infty$ for all $i \in I$, and $p_{B,1} = 0$ and $[r_B, d_B] = [0, \infty]$ for all $B \in \mathcal{B}$. Note that at most one milk run in B_i can be served, since all milk runs share the trip $\{q+i\}$ that may not be served more than once.

Now, a solution to the constructed DDSP-het instance corresponds to a selection of at most one milk run from each set B_i, i.e., to a selection of clauses that can simultaneously be satisfied by each variable x_i, and vice-versa. Observe that the construction can be carried out in polynomial time, since every variable appears in at most four clauses, which bounds the size of the sets B_i. NP-completeness follows from the fact that solutions to DDSP-het can be checked in polynomial time. □

We conclude that DDSP-het combines multiple decisions, each of which is hard in its own right: scheduling trips with time windows, assigning trips to a limited fleet of trucks, and choosing a subset of milk runs. In light of this, we do not expect default solvers working on MIP model [DDSP-OP] to perform well for instances of realistic size, which is confirmed by our computational study (Sect. 5). Moreover, this indicates that even subproblems in a decomposition approach cannot be expected to be easily tractable. We propose problem-specific exact solution methods in the following.

4 Benders decomposition

To improve performance, we propose an exact algorithm based on Benders decomposition (Benders, 1962) in this section. The main advantage of this type of decomposition approach lies in splitting the problem into two smaller, more manageable partial problems, for which specialized algorithms may be used. Benders decomposition based approaches have proven to be quite successful in solving complex scheduling problems (recently, e.g., Li & Womer, 2009; Cao et al., 2010; Verstichel et al., 2015; Emde et al., 2020)

However, straightforward application of a classic Benders decomposition scheme as originally proposed by Benders (1962) is not promising because the presence of big-M constraints weakens the LP relaxation of the master model and/or the Benders cuts, depending on whether they end up in the master or subproblem (Codato & Fischetti, 2006). Therefore, we present a logic-based Benders decomposition of DDSP-het that foregoes the use of these constraints altogether in Sect. 4.1.

4.1 Logic-based Benders decomposition

The presence of big-M Constraints (7) weakens the LP relaxation of model [DDSP-OP], which makes it significantly harder to solve for default solvers. In this section, we will there-

[1] For example, for the formula $(x_0 \vee x_1 \vee x_2) \wedge (x_0 \vee \bar{x}_1 \vee \bar{x}_2) \wedge (\bar{x}_0 \vee x_1 \vee \bar{x}_2) \wedge (\bar{x}_0 \vee \bar{x}_1 \vee x_2)$, we get $B_1 = \{\{5, 1, 2\}, \{5, 1\}, \{5, 2\}, \{5, 3, 4\}, \{5, 3\}, \{5, 4\}\}$, $B_1 = \{\{6, 1, 3\}, \{6, 1\}, \{6, 3\}, \{6, 2, 4\}, \{6, 2\}, \{6, 4\}\}$, and $B_3 = \{\{7, 1, 4\}, \{7, 1\}, \{7, 4\}, \{7, 2, 3\}, \{7, 2\}, \{7, 3\}\}$.

fore generate so-called combinatorial cuts in the spirit of Hooker (2000) and Codato and Fischetti (2006) with the goal of by-passing the big-M constraints altogether. This approach is generally referred to as *logic-based Benders decomposition* (LBBD) (Hooker and Ottosson, 2003). Recently, Lam et al. (2020) proposed a partially automated LBBD approach by combining default MIP and CP solvers. Unlike this approach, we reformulate the master problem and propose valid inequalities as well as a specialized procedure for solving the subproblem.

4.1.1 Reformulation of the master problem

For ease of notation, without loss of generality, we assume that trips are sorted by non-decreasing release date, i.e., we assume that $i < i'$ if $r_i < r_{i'}$. We reformulate the master problem by introducing binary variables v_{ik}, taking value 1 if trip $i \in I$ is assigned to truck $k \in \{1, \ldots, m\}$. We also use auxiliary continuous variable z, which constitutes a lower bound on the maximum weighted flow time.

$$\text{[Master-C] Minimize } Z_{CB}(\boldsymbol{v}, z) = z \tag{17}$$

subject to

$$\sum_{k=1}^{m} v_{ik} = 1 \qquad \forall i \in I \tag{18}$$

$$v_{ik} \in \{0, 1\} \qquad \forall i \in I, k = 1, \ldots, m \tag{19}$$

$$z \geq 0 \tag{20}$$

Objective function (17) minimizes the surrogate objective of the maximum weighted flow time as expressed by auxiliary variable z. This is also called a subproblem relaxation (Hooker, 2007). Constraints (18) ensure that every trip is assigned to exactly one truck, and the remaining constraints define the domain of the variables.

Model [Master-C] is a relaxation of the original problem that merely assigns trips to trucks, but does not sequence them and, consequently does not calculate the objective value. The constraint set of [Master-C] is iteratively extended by cuts as we describe in detail in Sect. 4.1.3. Whenever the solver finds a candidate integer solution for the master model, i.e., an assignment of trips to trucks, it is passed to the subproblem, where the assigned trips are sequenced. Information about the feasibility and optimality is then fed back to the master model by way of cuts. The search continues until all nodes have been fathomed or cut off. The best found solution is optimal.

4.1.2 Valid inequalities

While the procedure outlined above, using the cuts outlined in Sect. 4.1.3, will find the optimal solution eventually, it may take a long time, because model [Master-C] contains no meaningful information about solution quality. The solver will thus most probably generate a lot of mediocre candidate solutions before a sufficient number of rows have been added to guide the search into the promising regions of the search space. To speed up convergence, we therefore propose a number of valid inequalities to be added to model [Master-C].

First off, we can discard milk runs in set $\mathcal{B}$ that are not necessary in an optimal solution. This is clearly the case for any milk run $B \in \mathcal{B}$ where $r_B + \min_{c \in C}\{p_{Bc}\} > d_B$. Similarly,

B need not be considered if $w_B \cdot \min_{c \in C}\{p_{Bc}\} \geq \sum_{i \in B} w_i \cdot (d_i - r_i)$, i.e., if the minimum cost of the milk run is not less than the sum of the maximum cost of the constituent trips.

Regarding valid inequalities, we forbid the assignment of pairs of trips that cannot be processed by the same truck without violating at least one of the time windows, i.e.,

$$\begin{aligned} &\upsilon_{ik} + \upsilon_{i'k} \leq 1, \forall i, i' \in I, k = 1, \ldots, m : \\ &\neg(\exists B \in \mathcal{B} : i \in B \vee i' \in B) \wedge r_i + p_{i\eta(k)} > d_{i'} \\ &- p_{i'\eta(k)} \wedge r_{i'} + p_{i'\eta(k)} > d_i - p_{i\eta(k)}. \end{aligned} \tag{21}$$

If two trips i and i' are compatible, they may contribute to the objective value of the solution if assigned to the same truck. If the trips are processed in order $\langle i, i' \rangle$, then the maximum flow time cannot be less than $g_{ii'} = w_{i'} \cdot (r_i + p_{i\eta(k)} + p_{i'\eta(k)} - r_{i'})$. Similarly, if the processing order is $\langle i', i \rangle$, then the maximum flow time cannot be less than $g_{i'i} = w_i \cdot (r_{i'} + p_{i'\eta(k)} + p_{i\eta(k)} - r_i)$. Note that we can set $g_{ii'} := \infty$ if $r_i + p_{i\eta(k)} + p_{i'\eta(k)} > d_{i'}$ (analogous for $g_{i'i}$). Therefore, we can bound the surrogate objective z by

$$\min\{g_{ii'}, g_{i'i}\} \cdot (\upsilon_{ik} + \upsilon_{i'k} - 1) \leq z, \forall i, i' \in I, k = 1, \ldots, m : \neg\exists B \in \mathcal{B} : i \in B \vee i' \in B. \tag{22}$$

We can extend the idea of the previous inequalities to sets of more than two trips. Let $N(i) = \{i' \in I : r_{i'} \geq r_i\}, \forall i \in I$, be the set of trips whose release date is not sooner than that of trip i. Furthermore, let $N_l(i) \subseteq N(i)$ be the l trips from set $N(i)$ with the earliest release dates. Let $\tilde{p}_{ic} = p_{ic}$ if $\neg\exists B \in \mathcal{B} : i \in B$, i.e., if trip i is not part of any potential milk run. Otherwise, let $\tilde{p}_{ic} = \min\{p_{ic}, \min_{B \in \mathcal{B} : i \in B}\{\frac{p_{Bc}}{|B|}\}\}$. Then, we add the following constraints to [Master-C].

$$r_i + \sum_{i' \in N_l(i)} \tilde{p}_{i'\eta(k)} \cdot \upsilon_{i'k} \leq \max_{i' \in N_l(i)} \{d_{i'}\}, \forall i \in I; k = 1, \ldots, m; l = 1, \ldots, |N(i)|. \tag{23}$$

Note that these constraints are based on the (mild) assumption that milk run trips do not have a later deadline than their constituent individual trips. If that is not the case, the right-hand sides must be adjusted similarly to the processing time $\tilde{p}$.

Inequalities (23) essentially state that the last job in a set of jobs in earliest release date order cannot finish sooner than the earliest release date of those jobs, plus their processing times if assigned to the same machine. The same idea can be exploited for determining the minimum contribution to the objective function by adding

$$\begin{aligned} &\min_{i' \in N_l(i)} \{w_{i'}\} \cdot \left(r_i + \sum_{i' \in N_l(i)} \tilde{p}_{i'\eta(k)} \cdot \upsilon_{i'k} - \max_{i' \in N_l(i)} \{r_{i'}\} \right) \\ &\leq z, \forall i \in I; k = 1, \ldots, m; l = 1, \ldots, |N(i)| \end{aligned} \tag{24}$$

to [Master-C].

Finally, we can break symmetries by including Constraints (25) in the master model, inspired by Bertoli et al. (2018). For truck classes c containing more than one truck ($\gamma(c) > 1$), truck $k+1$ must have at least one higher-index trip than truck k, thus excluding many symmetric solutions with equivalent objective values.

$$\upsilon_{ik} \leq \sum_{\substack{i' \in I: \\ i' > i}} \upsilon_{i',k+1}, \forall i \in I; k = \sum_{c'=1}^{c-1} \gamma(c) + 1, \ldots, \sum_{c'=1}^{c} \gamma(c); c \in C \tag{25}$$

Example (cont.) Consider the example from Sect. 3.1. Trips 4 and 5 can never be together on the same truck of class $c = 2$, because no matter in what order they are processed, a time window will be violated. Hence, one inequality (21) is added as

$$\upsilon_{4,2} + \upsilon_{5,2} \leq 1.$$

Similarly, trips 1, 2, and 3 can never be processed by the same truck of class 2 in any order, therefore the assignment is made impossible by an inequality of type (23) for $N_3(1) = \{1, 2, 3\}$

$$1 + 2 \cdot \upsilon_{1,2} + 3 \cdot \upsilon_{2,2} + 3 \cdot \upsilon_{3,2} \leq 8.$$

By inequalities (24), the objective contribution z for set $N_2(1) = \{1, 2\}$ on truck 1 cannot be less than

$$2 \cdot (1 + 2 \cdot \upsilon_{1,1} + 3 \cdot \upsilon_{2,1} - 2) \leq z,$$

i.e., the objective value cannot be less than 8 if trips 1 and 2 are assigned to truck 1.

4.1.3 Combinatorial cuts

Whenever the branch and cut process solving model [Master-C] finds a candidate integer solution $\bar{\boldsymbol{\upsilon}}$, the subproblem is invoked. From the viewpoint of the subproblem, the assignment of trips to trucks is given, but not the order in which these trips are processed. More specifically, for each $k = 1, \ldots, m$, the trips assigned to truck k are $\hat{R}_k = \{i \in I \mid \bar{\upsilon}_{ik} = 1\}$. The subproblem then consists of finding the optimal permutation π_k for each given set R_k, such that all trips complete within their time windows ($r_i + p_{i\eta(k)} \leq \tau(i) \leq d_i, i \in R_k$) and the maximum flow time $\max_{i \in R_k}\{w_i \cdot (\tau(i) - r_i)\}$ is minimal.

To determine the optimal job sequence for each truck k, we need to solve a single machine scheduling problem with release dates and deadlines minimizing the maximum weighted flow time ($[1|r_j, d_j|\max w_j F_j]$), where trips can potentially be combined into milk runs. Note that even without the additional complication of milk runs, just solving $[1|r_j, d_j|\max w_j F_j]$ remains NP-hard in the strong sense as merely finding a feasible solution is equivalent to solving $[1|r_j|L_{\max}]$.

We propose the following dynamic programming (DP) scheme, based on the classic DP for sequencing problems as originally introduced by Held and Karp (1962). The DP is divided into stages, where in each stage one trip or milk run is added to the emerging schedule. Each stage consists of states $(\Xi, \mathcal{C})$, denoting the set Ξ of trips already scheduled and the makespan $\mathcal{C}$ of the (partial) schedule. Starting from dummy stage $l = 0$ with dummy state $(\Xi, \mathcal{C}) = (\emptyset, 0)$, a successor state $(\Xi', \mathcal{C}') = (\Xi \cup \{i\}, \max\{\mathcal{C} + p_{ik}; r_i + p_{ik}\})$ or $(\Xi', \mathcal{C}') = (\Xi \cup B, \max\{\mathcal{C} + p_{Bk}; r_B + p_{Bk}\})$ in stage $l + 1$ is reached by appending one trip $i \in \hat{R}_k \backslash \Xi$ or one milk run $B \in \mathcal{B} : B \subseteq \hat{R}_k \backslash \Xi$ to the emerging schedule. We only consider a transition from state $(\Xi, \mathcal{C})$ to state $(\Xi', \mathcal{C}')$ if it can still lead to a feasible solution. Specifically, the following criteria must be satisfied.

First, all remaining trips can still be processed in no-wait earliest deadline (EDD) order, i.e.,

$$\mathcal{C}' + \sum_{j=1}^{\ell} \tilde{p}_{\pi^{\mathrm{EDD}}_{\Xi'}(j)\eta(k)} \leq d_{\pi^{\mathrm{EDD}}_{\Xi'}(\ell)}, \forall \ell = 1, \ldots, |\hat{R}_k| - |\Xi'|,$$

where $\pi^{\mathrm{EDD}}_{\Xi'}(j)$ denotes the j-th trip in EDD order from set $\hat{R}_k \backslash \Xi'$. Note that if the remaining trips cannot be processed in EDD order, they cannot be processed within their time windows

at all (Lawler, 1973), and state (Ξ', C') can consequently be pruned. Also note that processing time $\tilde{p}$ is adjusted to account for potential milk runs in $\hat{R}_k \setminus \Xi'$, similar to Eq. (23).

Second, state (Ξ', C') must not be dominated by another state. A state is dominated if another state exists that covers (at least) the same trips, whose last trip ends no later, and whose partial objective value is no greater. Formally, a state (Ξ', C') is dominated by another state (Ξ'', C'') if $\Xi' \subseteq \Xi''$ and $C' \geq C''$ and $H(\Xi', C') \geq H(\Xi'', C'')$, where $H(\Xi, C)$ denotes the (partial) objective value of state (Ξ, C).

Let $\Gamma(\Xi', C')$ be the set of states from which a transition to state (Ξ', C') exists. For ease of notation, let I^* be either the trip that is appended in this transition (if $|\Xi' \setminus \Xi| = 1$) or the milk run B that is appended (if $\Xi' \setminus \Xi = B$). Then the (partial) objective value of state (Ξ', C') can be calculated recursively as

$$H(\Xi', C') = \min_{(\Xi, C) \in \Gamma(\Xi', C')} \{\max\{H(\Xi, C), w_{I^*} \cdot (C' - r_{I^*})\}\},$$

where $H(\emptyset, 0) := 0$. The optimal objective value is $\min_{C \in \mathbb{N}}\{H(\hat{R}_k, C)\}$; the corresponding optimal trip and milk run assignment R_k, trip sequence π_k, and completion times τ are determined by backward recovery along the optimal path.

The number of states in the DP is bounded by $O(2^n \cdot \max_{i \in \hat{I}}\{d_i\})$. Each state can have $O(n + |\mathcal{B}|)$ successors, and the objective value can be calculated in $O(1)$. Consequently, the total computational effort is bounded by $O((n + |\mathcal{B}|) \cdot 2^n \cdot \max_{i \in \hat{I}}\{d_i\})$.

DP may not return any feasible solution at all if all states have been pruned before ever reaching a final state $(\hat{R}_k, C)$. In this case, the trip assignment $\hat{R}_k$ simply does not allow a trip sequence which does not violate time windows. Consequently, we generate the minimum infeasible subsets of $\hat{R}_k$ by checking for each subset $\bar{R} \in \wp(\hat{R}_k)$ (where $\wp(\hat{R}_k)$ denotes the powerset of $\hat{R}_k$) whether there is at least one permutation of the trips in $\bar{R}$ that does not violate a time window. Note that this may also require checking any potentially milk runs if some trips in $\bar{R}$ can be combined. If no such permutation exists, then the trips in $\bar{R}$ can never feasibly be processed by the same truck and we add them to the set of infeasible trip combinations $\mathcal{R}_k := \mathcal{R}_k \cup \bar{R}$. Finally, after all infeasible trip combinations have been added to $\mathcal{R}_k$, we prune set $\mathcal{R}_k$ by removing all sets $\bar{R} \in \mathcal{R}_k$ that are true supersets of other sets $\bar{R}' \in \mathcal{R}_k$, i.e., we delete set $\bar{R} \in \mathcal{R}_k$ if $\exists \bar{R}' \in \mathcal{R}_k : \bar{R} \supset \bar{R}'$. Then we add cuts

$$\sum_{i \in \bar{R}} (1 - \upsilon_{i\eta(k)}) \geq 1, \forall \bar{R} \in \mathcal{R}_k; k = 1, \ldots, m : \text{ DP outputs no feasible solution for } \hat{R}_k. \tag{26}$$

Example (cont.) Assume that $\hat{R}_2 = \{1, 2, 3, 4\}$. The set of all infeasible subsets is $\{\{1, 2, 3\}, \{1, 2, 3, 4\}, \{1, 2, 4\}, \{2, 3, 4\}, \{3, 4\}, \{1, 3, 4\}\}$. After pruning, the minimum infeasible subsets are $\mathcal{R}_k = \{\{1, 2, 3\}, \{1, 2, 4\}, \{3, 4\}\}$, leading to cuts

$$\begin{aligned}(1 - \upsilon_{1,2}) + (1 - \upsilon_{2,2}) + (1 - \upsilon_{3,2}) &\geq 1\\ (1 - \upsilon_{1,2}) + (1 - \upsilon_{2,2}) + (1 - \upsilon_{4,2}) &\geq 1\\ (1 - \upsilon_{3,2}) + (1 - \upsilon_{4,2}) &\geq 1\end{aligned}$$

Note that the powerset $\wp(\hat{R}_k)$ may contain a prohibitively large number of elements if $|\hat{R}_k|$ is large. Therefore, we can restrict the size of subsets $\bar{R}$ to some sufficiently low number of elements ζ, i.e., we only consider subsets that satisfy $|\bar{R}| \leq \zeta$. That cuts down on the computational effort, but carries the risk that no infeasible subset is found if ζ is too small. In that case, a cut of type (26) is added where $\mathcal{R}_k := \{\hat{R}_k\}$, i.e., the whole schedule for truck

k is made infeasible. In our computational experiments, this turned out to be unnecessary, and we left $\zeta = \infty$.

If the solution is feasible, it may still be suboptimal. Let $i^* = \arg\max_{i \in \bigcup_{k=1}^{m} R_k} \{w_i \cdot (\tau(i) - r_i)\}$ be the trip or milk run with the greatest weighted flow time. Let $k^* \in \{1, \ldots, m\}$ such that $i^* \in R_{k^*}$ be the truck processing trip or milk run i^* in the current solution. Moreover, let $j^* \in \{1, \ldots, |R_{k^*}|\}$ such that $\pi_{k^*}(j^*) = i^*$ be the sequence position of trip i^*. Finally, let $j^{**} = \arg\max_{j=1,\ldots,j^*} \{\tau(\pi_{k^*}(j)) - p_{\pi_{k^*}(j)\eta(k^*)} = r_{\pi_{k^*}(j)}\}$ be the last trip before the critical trip i^* to start without delay at its release date. To lower the objective value, at least one of the trips in subsequence $\langle \pi_{k^*}(j^{**}), \ldots, \pi_{k^*}(j^*) \rangle$ must change its vehicle. Hence, we add cut

$$\sum_{j=j^{**}}^{j^*} (1 - \upsilon_{\pi_{k^*}(j)\eta(k^*)}) \geq 1. \tag{27}$$

Finally, if a new feasible best solution has been found, it is stored, and the following cut is added

$$z \leq UB - \epsilon, \tag{28}$$

where UB is the objective value of the new best solution and ϵ is a sufficiently small number ($\epsilon = 1$ if all parameters are integer).

Note that these cuts make the current incumbent solution infeasible. The branch and cut scheme solving [Master-C] will thus terminate when there are no more unfathomed nodes left to explore. The best feasible solution found up to that point is then guaranteed to be optimal. An outline of the entire procedure is provided in Algorithm 1.

input: instance of DDSP-het
$UB := \infty$;
Start solving IP model [Master-C] with valid inequalities from Sect. 4.1.2 using CPLEX;
while *there are feasible solutions in the search space* **do**
  $\bar{\upsilon}$:= a candidate integer solution of model [Master-C];
  Solve the subproblem for the given $\bar{\upsilon}$ via dynamic programming;
  if *the subproblem has no solution* **then**
    Determine minimum infeasible subets of trip assignments;
    Add cuts of type (26) to [Master-C];
  else
    Add cuts of type (27) to [Master-C];
    if $UB >$ *objective value of the current solution* **then**
      Save new best solution;
      UB := objective value of the current solution;
      Add a cut of type (28) to [Master-C];
return *the optimal solution with objective value* UB;

Algorithm 1: Logic-based Benders decomposition for DDSP-het.

Example (cont.) Consider the example from Sect. 3.1. Assume that the current master solution is $\bar{\upsilon}_{1,1} = \bar{\upsilon}_{3,1} = \bar{\upsilon}_{4,1} = \bar{\upsilon}_{2,2} = \bar{\upsilon}_{5,2} = 1$, all other $\bar{\upsilon}$ variables are 0. This implies that trips $\hat{R}_1 = \{1, 3, 4\}$ are assigned to truck 1 and trips $\hat{R}_2 = \{2, 5\}$ to truck 2. The resulting DP graph for the first truck is depicted in Fig. 4. The optimal path $\pi_1 = \langle 1, 3, 4 \rangle$ is bold. The solution is feasible. Trip $i^* = 3$ is the trip with the greatest weighted flow time (tied with trip 5). Trip 3 is at sequence position $j^* = 2$. The last trip to be on-time before trip 3 is trip 1 in sequence position $j^{**} = 1$. Consequently, at least one of these two trips needs to be moved to a different truck for the objective value to improve, i.e., $(1 - \upsilon_{1,1}) + (1 - \upsilon_{3,1}) \geq 1$.

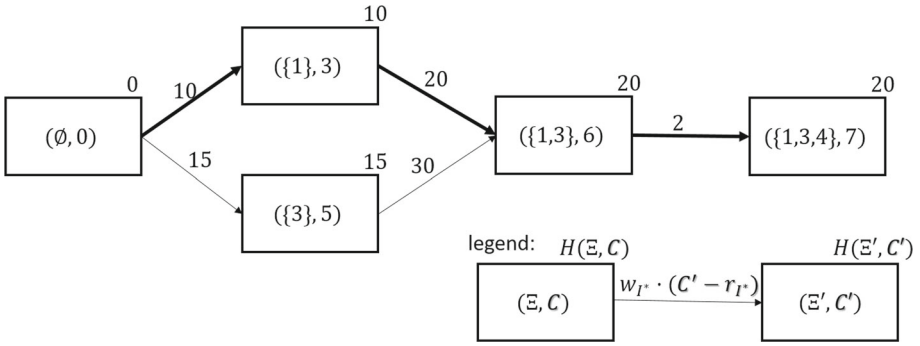


Fig. 4 DP graph in the example for $R_1 = \{1, 3, 4\}$

5 Computational study

We test the models and solution methods in this section. First, we describe the test setup and instances in Sect. 5.1. In Sects. 5.2.1 and 5.2.2, we analyze the performance of the models and algorithms. We then look into the effect of minimizing the maximum flow time as opposed to the total flow time (Sect. 5.2.3) and, finally, investigate the effect of combining trips in milk runs in Sect. 5.2.4.

5.1 Test data and setup

Since DDSP-het is based on the direct delivery scheduling problem as introduced by Emde and Zehtabian (2019) and Gschwind et al. (2020), we adopt their instance generation scheme to generate the fundamental trips. As in the previous publications, we consider instances with $n \in \{25, 125\}$ trips. Assuming a 24 hour planning horizon, subdivided into periods of 10 minute length, we set the release date of trip i to $r_i = \text{rnd}^U(\{1, \ldots, 144\})$, the deadline to $d_i = r_i + p_{i,1} + \text{rnd}^U(\{3, \ldots, 48\})$, and the weight to $w_i = 1$, where rnd^U denotes a uniformly distributed random integer from the set given in the argument.

The original instances assume a homogeneous vehicle fleet, whereas we consider $|C| \in \{3, 10\}$ truck classes. For each trip $i \in I$ and truck class $c \in C$, we set the processing time to $p_{ic} = \text{rnd}^U(\{6, \ldots, 36\})$, similar to the original papers for the homogeneous case. We set the total number of trucks over all classes to $m = \max\{|C|, \lceil n/5.5 \rceil\}$, and then distribute this total number randomly among the truck classes such that $\sum_{c \in C} \gamma(c) = m$ and $\gamma(c) \geq 1$, $\forall c \in C$.

Since we fundamentally consider a direct delivery logistics system, the assumption is that only small groups of trips can be combined as milk runs. Specifically, we consider all combinations of two or three trips, i.e., $\mathcal{B} = \{\{i, i'\} \mid i, i' \in I, i < i'\} \cup \{\{i, i', i''\} \mid i, i', i'' \in I, i < i' < i''\}$. Let $B = \{i, i'\}$ ($B = \{i, i', i''\}$) be such a pair (triplet) of trips. The parameters of combined trip B are set to $p_{Bc} = \left\lceil \phi \cdot \text{rnd}^U(\{p_{ic}, \ldots, \sum_{i' \in B} \max_{c' \in C}\{p_{i'c'}\}\}) \right\rceil$, $r_B = \min_{i \in B}\{r_i\}$, $d_B = \min_{i \in B}\{d_i\}$, $w_B = \sum_{i \in B}\{w_i\}$, where $\phi > 0$ is a factor determining the time savings when combining trips, which we set to one unless otherwise noted. Only such combined trips are created where $r_B + \max_{c \in C}\{p_{Bc}\} \leq d_B$.

This instance generation scheme does not guarantee that a feasible solution even exists for a given instance. Therefore, if none of our solution schemes finds a feasible solution, the instance is scrapped, and a new one is generated in its stead.

For each combination of truck class count and trip count, we generate 10 instances, plus two other sets of 10 instances with $n = 125$ and $n = 175$ trips, respectively, where $|C| = 2$, i.e., 60 random instances in total. Instances are named according to the following scheme: $Rn\_|C|\_j$, where n is the number of trips, $|C|$ the number of truck classes, and j a running index. Finally, we also adapt 24 famous R1 instances from Solomon (1987), originally proposed for the VRP with time windows. All instances are available at https://github.com/ECdQGt/DDSP-het.

5.2 Computational results

We implement all algorithms in C# 8. As a default solver, we use ILOG CPLEX 12.9. All tests are run on an x64 PC equipped with an Intel Core i5-8265U CPU and 8 GB of RAM. All codes (including the default solver) are executed on a single thread. Otherwise, all parameters are left in their default settings unless otherwise noted.

In Sect. 5.2.1 we test the usefulness of the valid inequalities proposed in Sect. 4.1.2. We compare several different solution methods and investigate the computational performance in Sect. 5.2.2. Finally, in Sects. 5.2.3 and 5.2.4, we compare different objective functions to check whether they can serve to minimize delays in case of unforeseen disturbances of the schedule and the influence of milk runs, respectively.

5.2.1 Effect of valid inequalities

As we describe in Sect. 4.1.2, we add valid inequalities to the master model of LBBD to speed up convergence. To test whether the proposed inequalities have the intended effect, we run LBBD with and without these inequalities on 5 instances generated as described in Sect. 5.1 with $n = 20$ trips and $|C| = 2$ truck classes. The results, averaged over the five instances, are in Table 2. The table shows the number of combinatorial Benders cuts added, the total time spent solving the subproblem, and the total CPU time (including master and subproblem).

All inequalities contribute to lowering the CPU times. The most effective valid inequalities are apparently the time window constraints (23), which stop many infeasible candidate solutions from being generated. The subproblem relaxation constraints (22) and (24), setting the surrogate objective value z, also have a strong effect on runtimes, while the influence of the symmetry breaking constraints (25) is relatively weak but still noticeable. The best result, however, is achieved when all constraints are active.

Table 2 Solution performance of LBBD with and without valid inequalities

Valid inequalities	# cuts	CPU sec (sub)	CPU sec (total)
None	2869.8	32.7	33.4
Only (21) and (22)	491.0	2.4	2.6
Only (23)	422.4	0.2	0.7
Only (24)	467.6	4.4	4.4
Only (25)	2799.0	22.7	23.3
All	21.6	0.1	0.2

5.2.2 Algorithmic performance

We test five different solution approaches on the small instances with $n = 25$ trips. First, CPLEX solving the *single MIP model* from Sect. 3.2. Second, the classic Benders decomposition scheme as implemented in the automated Benders decomposition logic that ships with CPLEX since version 12.7 (IBM , 2016). We use the fully automated level, meaning that CPLEX decides on its own how best to decompose model [DDSP-OP], and do not otherwise change any parameters or settings. Third, the *constraint programming (CP)* model introduced in Sect. 3.2 solved by ILOG CP Optimizer 12.9. Fourth, *LBBD using restarts*, i.e., the master model is re-solved from scratch every time new cuts have been added via the subproblem. Note that this is the classic approach as proposed by Benders (1962). Finally, *LBBD using lazy constraints*, where cuts are added iteratively to the branch and cut tree as it grows. This general approach is sometimes called *branch and Benders cut* (Rahmaniani et al., 2017) or *branch and check* (Thorsteinsson, 2001). We set a time limit of 1800 CPU seconds for the approaches based on Benders decomposition and 3600 s for the single model approaches (MIP and CP).

The results are in Table 3. All approaches can solve all small instances to optimality within the time limit. However, classic Benders cuts without lifting (e.g., Tang et al., 2013) or other acceleration techniques (e.g., Saharidis & Ierapetritou, 2013) do not work well in the presence of big-M constraints (Codato & Fischetti, 2006). The automatic Benders decomposition integrated into CPLEX exhibits CPU times that are weak compared to the other approaches.

CPLEX solving the single model performs well on the very smallest instances ($n = 25$, $|C| = 3$), but quickly loses ground to CP and LBBD when the instances grow larger. Comparing CP and LBBD, the performance is similar, which is maybe not surprising, given that the basic algorithmic concepts (iteratively paring down the search space via logic-based considerations) are somewhat related. Regarding the two LBBD versions, it is interesting to note that the lazy version of LBBD often generates more cuts than the restart version, because the latter can profit from the additional cuts when solving the root node relaxation in later iterations, thus requiring fewer iterations overall. Generally speaking, however, this does not appear to offset the additional effort wasted on revisiting the same branches of the branch-and-bound tree over and over again, as both versions are about equally fast.

Since lazy LBBD, CP, and CPLEX solving the single model are the three most successful approaches on the small instances, we restrict ourselves to these three solution methods for the large instances ($n = 125$). However, it turns out that CPLEX is incapable of solving the single model to feasibility for any large instance within the 60-minute time limit. At least in some instances—specifically, those with a low number $|C| \leq 3$ of truck classes – it manages to find relatively tight lower bounds, however. Table 4 therefore only shows the best lower bound for CPLEX as well as the best upper bound for CP and LBBD using lazy constraints.

Overall, CP performs well on the instances with high truck heterogeneity ($|C| \geq 3$). It works particularly well on the large instances with $|C| = 10$ truck classes, but takes a long time on the instances with a medium number of truck classes ($|C| = 3$), exhibiting runtimes in excess of 30 minutes in the majority of cases. On the low-heterogeneity instances ($|C| = 2$), CP struggles to find tight upper bounds for several instances and is clearly outperformed by LBBD, which finds a greater number of optimal solutions and tighter bounds in less time on average. For all instances, LBBD finds at least a good feasible solution, with only instance R125_10_42 showing a substantial optimality gap. On the other hand, the solution times for the $|C| \leq 3$ instances are shorter for LBBD than CP in the majority of cases. Looking at the

Table 3 Results on small instances ($n = 25$)

Instance	$\mathcal{Z}$	single model	CPLEX Benders		CP	LBBD (restarts)		LBBD (lazy)	
		CPU s.	# cuts	CPU s.	CPU s.	# cuts	CPU s.	# cuts	CPU s.
R25_3_0	27	1.3	14	6.6	0.2	2	0.5	19	0.7
R25_3_1	25	0.5	4	0.5	0.1	4	0.6	14	0.1
R25_3_2	30	0.8	17	7.8	0.1	2	0.2	12	0.2
R25_3_3	32	0.6	19	10.4	0.3	2	0.4	21	0.6
R25_3_4	26	2.1	14	10.8	0.2	2	0.4	27	0.5
R25_3_5	25	0.3	4	1.3	0.1	4	0.2	10	0.3
R25_3_6	25	1.2	12	3.3	0.1	2	0.2	11	0.5
R25_3_7	29	1.2	16	4.6	0.2	2	0.5	26	0.4
R25_3_8	27	0.5	5	1.8	0.2	4	0.3	21	0.2
R25_3_9	32	1.2	4	2.7	0.1	2	0.5	23	0.5
Avg.		1.0	10.9	5.0	0.2	2.6	0.4	18.4	0.4
R25_10_10	15	2.1	3	12.2	0.6	4	0.5	11	0.3
R25_10_11	13	2.2	7	3.4	0.2	2	0.3	13	0.4
R25_10_12	16	3.2	0	8.2	0.3	4	0.6	13	0.3
R25_10_13	15	2.6	6	14.0	0.2	2	0.3	17	1.4
R25_10_14	16	3.3	5	8.8	0.0	6	0.6	14	0.3
R25_10_15	17	4.6	14	12.5	0.5	2	0.3	10	0.2
R25_10_16	12	1.9	1	2.4	0.2	13	2.4	21	0.4
R25_10_17	11	1.5	8	4.0	0.0	2	0.3	12	0.7
R25_10_18	15	3.6	2	12.0	0.3	4	0.5	13	0.3
R25_10_19	15	2.9	6	21.5	0.3	4	0.4	11	2.1
Avg.		2.8	5.2	9.9	0.3	4.3	0.6	13.5	0.7

Table 4 Results on large instances ($n = 125$); optimal objective values are bold

Instance	Single model	CP		LBBD			
	LB	UB	CPU s.	UB	# cuts	CPU s. (sub)	CPU s.
R125_2_20	34	135	3600.4	35	213	0.2	1808.9
R125_2_21	32	**34**	1734.1	**34**	199	0.2	1442.8
R125_2_22	31	154	3600.5	**34**	208	17.5	318.5
R125_2_23	32	86	3600.5	**35**	287	9.2	1385.5
R125_2_24	32	**32**	2881.6	37	350	0.9	1811.7
R125_2_25	34	**34**	3294.2	**34**	99	0.2	284.8
R125_2_26	33	126	3602.1	**33**	118	0.2	1717.2
R125_2_27	32	**32**	1729.1	34	193	0.9	1807.4
R125_2_28	34	96	3600.7	**34**	120	0.1	415.9
R125_2_29	32	**33**	2465.9	**33**	104	0.2	1127.2
Avg.	32.6	76.2	3010.9	34.3	189.1	3.0	1212.0
R125_3_30	29	**31**	3451.6	32	132	2407.7	3245.2
R125_3_31	30	**32**	1765.1	33	450	457.1	1804.3
R125_3_32	35	**35**	1966.6	**35**	83	199.7	463.5
R125_3_33	31	**31**	2661.3	**31**	423	136.3	1002.5
R125_3_34	29	**33**	1929.5	**33**	68	19.5	410.8
R125_3_35	25	**32**	732.9	**32**	53	39.5	280.3
R125_3_36	33	**33**	2260.2	**33**	106	158.7	1786.1
R125_3_37	31	**31**	1940.4	**31**	87	1173.6	1469.9
R125_3_38	30	**30**	912.1	**30**	145	67.6	578.9
R125_3_39	24	**32**	1793.1	33	294	25.7	1805.7
Avg.	29.7	32.0	1941.3	32.3	184.1	468.5	1284.7
R125_10_40	0	**16**	57.5	**16**	221	64.7	1575.2
R125_10_41	0	**17**	96.3	**17**	20	0.7	1626.6
R125_10_42	2	**20**	99.2	31	56	9.8	1801.7
R125_10_43	1	**19**	79.1	21	117	0.7	1801.4
R125_10_44	1	**15**	30.7	18	263	224.5	1801.2
R125_10_45	0	**18**	42.6	**18**	61	195.1	567.6
R125_10_46	0	**17**	42.5	19	105	144.0	1801.3
R125_10_47	4	**20**	15.3	22	185	167.9	1801.1
R125_10_48	1	**19**	79.4	**19**	218	173.5	1613.1
R125_10_49	1	**18**	61.7	**18**	137	120.6	1693.1
Avg.	1.0	17.9	60.4	19.9	138.3	110.1	1608.2

detailed results, it is striking that LBBD spends a good part of the time solving the subproblem in many instances. This leads the procedure to exceed the time limit in one case (R125_3_30), because the time limit is only checked in the master solution process. Overall, we draw the conclusion that CP is a viable approach to solve DDSP-het with high truck heterogeneity, although LBBD is faster and finds tighter upper bounds on large instances with relatively few different truck classes. A reason for this behavior may be that it is easier for the master

Table 5 Results on very large instances ($n = 175$) for LBBD; CP and MIP run out of time or memory before finding any solution

Instance	UB	# cuts	CPU s. (sub)	CPU s.
R175_2_50	36	166	0.3	1818.9
R175_2_51	47	90	0.1	1820.2
R175_2_52	34	181	1.9	1835.0
R175_2_53	35	127	0.1	1824.1
R175_2_54	35	211	0.3	1825.0
R175_2_55	37	174	0.4	1838.0
R175_2_56	49	142	0.1	1830.1
R175_2_57	38	143	0.7	1830.7
R175_2_58	48	178	0.3	1823.4
R175_2_59	40	155	0.3	1840.0
Avg.	39.9	156.7	0.5	1828.5

problem to balance the workload among the trucks if they are relatively homogeneous, thus avoiding long trip sequences in the subproblem.

To test the limits of the solution methods, we also generate a set of very large instances with $n = 175$ trips. The results are in Table 5. Neither the CP nor the MIP solver manage to solve any of these instances even to feasibility due to running out of memory or time. LBBD, on the other hand, finds decent solutions within the half-hour time limit, at least for the low-heterogeneity instances ($|C| = 2$). On our test system with 8 GB of RAM, no method—including LBBD—can consistently solve very large instances with $|C| \geq 3$ truck classes. Nonetheless, the results in Table 5 suggest that LBBD scales comparatively well, at least for instances where the truck fleet is not too varied.

We also adapt the classic R1 instances with 25 and 50 customer locations from Solomon (1987), originally proposed for the VRP with time windows. For each customer location, we create a trip with a processing time equal to two times the Euclidean distance between the depot and the customer plus the given service time. We copy the time windows from the original instances, except that we normalize them such that the earliest release date is 0. Moreover, we generate milk runs as described in Sect. 5.1. The results are in Table 6. Both CP and LBBD are capable of solving the majority of these instances to optimality (CP solves 17, LBBD solves 19). However, LBBD is faster on most instances and finds much better upper bounds on many of those instances that are not solved to optimality by CP.

To get a clearer picture of the solution process of LBBD, we plot in Fig. 5 the evolution of the best lower and upper bounds of LBBD solving an example instance with $n = 50$ and $|C| = 3$ truck classes. The objective value of the master model serves as a lower bound on the maximum weighted flow time, because the master model is a relaxation of the original problem. The more combinatorial Benders cuts are added, the tighter the relaxation. The best upper bound is updated whenever a new best feasible solution is found when solving the subproblem.

Looking at the figure, the lower bound is quite strong from the outset: even in the root node, the lower bound is already 29, where the optimal objective value is 31. The evolution of the upper bound is somewhat slower. It takes about 5 s before the first feasible solution is found. The upper bound then improves quickly; after about 20 s, the optimal solution with objective value 31 has been found. However, it takes another approximately 25 s to prove its optimality. More broadly speaking, a very good solution is found after about a third of the runtime, suggesting that truncated versions of LBBD may deliver fair heuristic solutions.

Table 6 Results on Solomon R1 instances ($n = 25$ and $n = 50$) for LBBD and CP

Instance	CP		LBBD			
	UB	CPU s.	UB	# cuts	CPU s. (sub)	CPU s.
R101_025	**82**	0.0	**82**	13	< 0.1	0.1
R102_025	**82**	43.7	**82**	7	< 0.1	0.4
R103_025	**82**	43.4	**82**	8	< 0.1	0.9
R104_025	83	3600.1	83	268	< 0.1	1807.6
R105_025	**82**	0.0	**82**	6	< 0.1	0.2
R106_025	**82**	8.5	**82**	10	< 0.1	0.5
R107_025	**82**	16.6	**82**	8	< 0.1	1.3
R108_025	86	3600.0	86	21	< 0.1	1807.1
R109_025	**82**	0.2	**82**	12	< 0.1	0.4
R110_025	**82**	9.7	**82**	22	< 0.1	0.3
R111_025	**82**	47.6	**82**	11	< 0.1	0.5
R112_025	84	3600.0	84	55	< 0.1	1803.6
Avg.	82.6	914.1	82.6	36.8	< 0.1	451.9
R101_050	**98**	0.0	**98**	10	< 0.1	1.3
R102_050	**98**	455.4	**98**	28	< 0.1	4.1
R103_050	288	3600.6	**98**	5	< 0.1	6.5
R104_050	603	3600.3	99	89	< 0.1	1807.4
R105_050	**98**	0.1	**98**	11	< 0.1	2.8
R106_050	**98**	468.0	**98**	10	< 0.1	3.6
R107_050	**98**	1914.9	**98**	12	< 0.1	5.2
R108_050	543	3600.3	98	42	< 0.1	1808.8
R109_050	**95**	5.2	**95**	57	< 0.1	11.5
R110_050	**98**	819.6	**98**	50	< 0.1	3.8
R111_050	**98**	1857.0	**98**	19	< 0.1	4.0
R112_050	387	3600.2	**98**	22	< 0.1	4.0
Avg.	216.8	1660.1	97.8	29.6	< 0.1	305.2

5.2.3 Comparison of objectives

Apart from improving customer satisfaction and reducing work-in-process, one of the reasons for minimizing the weighted flow time cited in the literature on direct delivery scheduling is improving the robustness of the schedule (Emde & Zehtabian, 2019). The thinking is that starting trips early in their time window (which is encouraged by a flow time objective) makes it less likely that they will violate their deadlines in case of unforeseen delays. To investigate the veracity of this claim, Emde and Zehtabian (2019) propose the following simulation experiment, which they use to show a positive effect of their flow time objective on total delays.

Given a DDSP instance and a (feasible) solution, randomly increase the processing time of 20% of the trips in the instance by $\text{rnd}^U(\{1, \ldots, 12\})$. Following the same trip partition and sequence as in the given solution, the delays (i.e., increased processing time) may propagate because subsequent trips cannot be started as planned. Count the total weighted tardiness

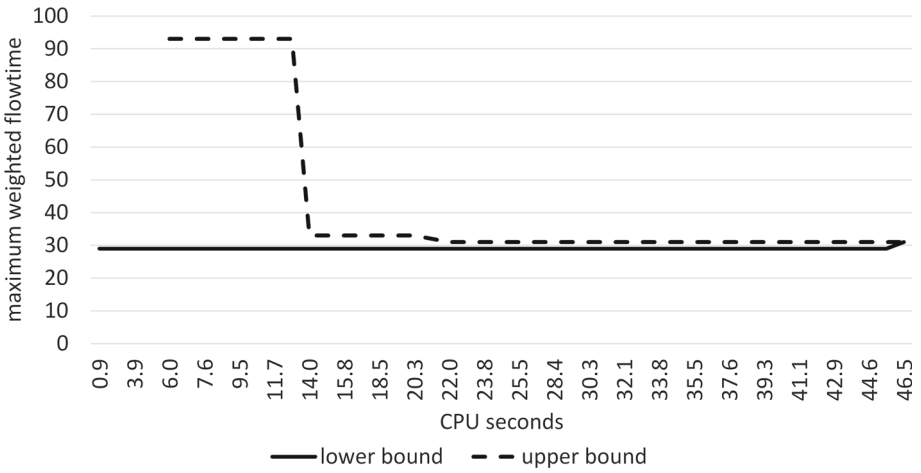
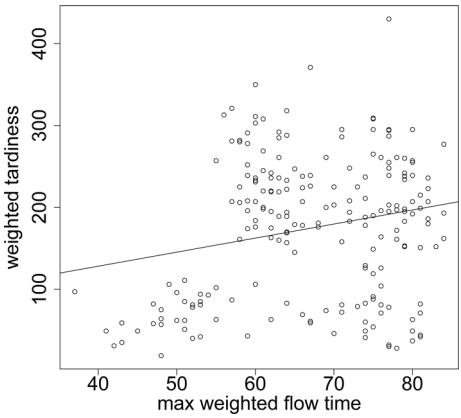


Fig. 5 Development of lower and upper bound over time for LBBD ($n = 50$, $|C| = 3$)

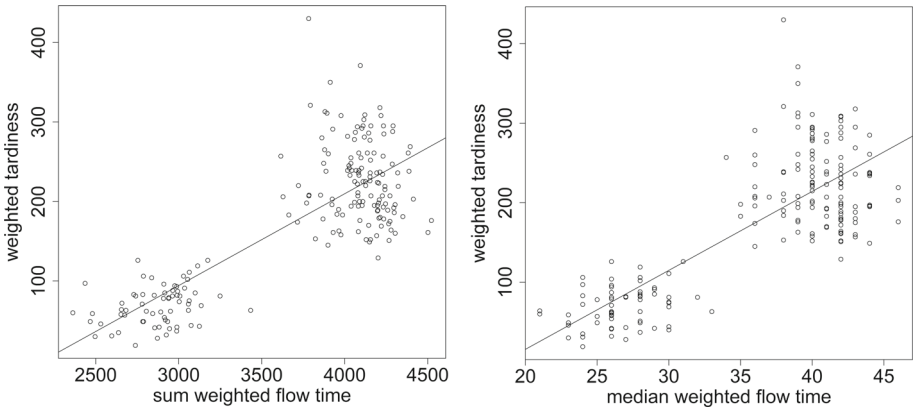
Fig. 6 Correlation between maximum weighted flow time and total tardiness



of all trips, i.e., the difference between the actual completion time including (propagated) delays and the deadline.

Unlike Emde and Zehtabian (2019) and Gschwind et al. (2020), we do not minimize the total weighted flow time but the maximum weighted flow time. To investigate whether this objective also protects against unforeseen disturbances, we adapt the simulation procedure outlined above. We generate 100 DDSP-het instances with $n = 100$ trips and $|C| = 1$ truck class. For each of these instances, we create a feasible solution by solving the single model [DDSP-OP] without an objective function, i.e., minimize $\mathbf{0}$, s.t. (2)–(11). For each solution, we then calculate three values: the maximum weighted flow time as we define it in this paper; the total weighted flow time as it is used in Emde and Zehtabian (2019) and Gschwind et al. (2020); and the median weighted flow time.

Figures 6 and 7a, b plot the maximum, total, and median weighted flow time, respectively, against the total tardiness from the simulation experiment for all 100 instances. All three objectives correlate with the measured tardiness in a highly significant manner (99% confidence level). The differences in the Pearson correlation coefficients are relatively minor, with minsum having the strongest correlation, and the median and maximum weighted flow



(a) Total weighted flow time. **(b)** Median weighted flow time.

Fig. 7 Correlation between objectives and total tardiness

time correlating a bit less strongly. Still, the correlation is highly significant in all cases. This suggests, at the very least, that minimizing the maximum weighted flow time can serve as an adequate objective when considering the robustness against unexpected time window violations, apart from the other benefits discussed in the introduction.

5.2.4 Effect of milk runs

While we fundamentally consider full-truckload direct deliveries, we also assume that it is possible to combine some trips in milk runs. Unlike tours in classic routing problems like the VRP, the total processing time of a milk run need not be a simple linear combination of its constituent arc weights. Since we deal with industrial deliveries, it is quite reasonable to assume that there are significant economies of scale, going beyond mere saved driving time, when multiple deliveries are combined, e.g., because the truck needs to approach the loading dock only once etc. To investigate the effect this consolidation of multiple shipments has, we generate another set of ten instances with $n = 25$ trips and $|C| = 1$ truck type. We generate these instances exactly as outlined in Sect. 5.1, except that we vary the factor ϕ between 0.25 and 1.5, determining how much the processing time of the milk run is compressed compared to the total processing time of the constituent trips: if $\phi = 0.25$, a milk run is a lot faster than processing its individual trips; if $\phi = 1.5$, milk run processing time tends to be longer than processing the trips individually.

Figure 8 shows the average objective value plotted against the consolidation factor $\phi \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$. Expectedly, it becomes easier to ensure responsive customer service if milk run processing is fast (low ϕ). However, interestingly, consolidation is somewhat beneficial even if the consolidation factor is high, i.e., if milk runs do not save much time, or may even take extra time. The reason is that milk run trips may have wider time windows than some of their constituent trips, allowing alternative (and sometimes better) assignments of trips to trucks. Overall, the effect of allowing milk runs is quite substantial even for larger ϕ, suggesting that enabling the combination of trips may allow significantly more flexibility for logistics providers.

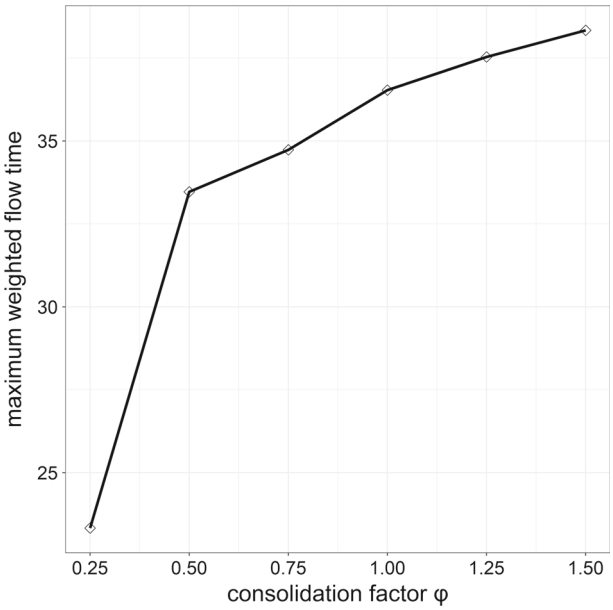


Fig. 8 Effect of consolidation on the usefulness of combining trips into milk runs

6 Conclusion

We study the problem of scheduling a given set of direct delivery trips between a depot and customers using a given heterogeneous truck fleet. Trucks differ in their processing speed and may also be incapable of processing certain trips. Moreover, some trips can optionally be combined in predefined milk runs, which is a novel feature of DDSP-het that has no direct equivalent in the scheduling or routing literature because the properties of a milk run (like processing time and time window) need not be a simple linear combination of its constituent trips. We adapt a MIP model from the literature, propose a new constraint programming model, investigate the computational complexity, and develop solution methods based on Benders decomposition.

In terms of complexity, we show that even deciding feasibility of this problem is NP-hard in the strong sense on all three levels: assigning trips to trucks, selecting milk runs, and scheduling trips on each individual truck. Despite this complexity, our computational tests indicate that both a default constraint programming solver and our logic-based Benders decomposition approach perform quite well, where CP performs better on instances with very heterogeneous trucks, while LBBD works better on instances with few different truck classes. Moreover, LBBD scales better, solving even very large instances at the very least to feasibility, whereas the MIP and CP solvers run out of time or memory before finding even a feasible solution. Classic Benders decomposition as well as a default solver working on the single model perform substantially less well.

The models, insights, and algorithms we develop for DDSP-het may fundamentally also be applicable to related scheduling problems. For instance, it would be straightforward to apply our valid inequalities (21) to (unrelated) parallel machine scheduling problems with time window constraints, while inequalities (22) are also applicable to any regular minmax objective, not just the weighted flow time. Similarly, the relatively good performance of the

constraint programming solver may also serve as an inspiration for researchers and practitioners who want to solve similar problems in mostly acceptable time without implementing problem-specific algorithms.

Our objective of minimizing the maximum weighted flow time deviates from the minsum objectives typically considered in the literature for similar problems. Replicating a simulation experiment originally proposed by Emde and Zehtabian (2019), we show that our minmax objective offers a similar level of protection against deadline violations as the minsum objective. Moreover, our tests indicate that the possibility of combining trips into milk runs may be beneficial because of the added flexibility even if the total processing time of the milk run is not much shorter than individually processing the constituent trips.

Future research can extend the problem further by combining a heterogeneous fleet with the multiple depots case. Given the relatively long solution times in some of the large instances, developing efficient heuristic approaches may also be a fruitful endeavor. Such methods should then be compared to the performance of truncated LBBD runs. Moreover, it may be possible to combine classic Benders cuts with logic-based combinatorial cuts to profit from both the dual information of a given integer solution as well as the problem-specific logic embedded in the LBBD cuts.

References

Al Theeb, N., Al-Araidah, O., & Aljarrah, M. H. (2019). Optimization of the heterogeneous vehicle routing problem with cross docking logistic system. *Logistics Research, 12,* 4.

Anand, S., Bringmann, K., Friedrich, T., Garg, N., & Kumar, A. (2017). Minimizing maximum (weighted) flow-time on related and unrelated machines. *Algorithmica, 77,* 515–536.

Annouch, A., Bouyahyaoui, K., Bellabdaoui, A. (2016). A literature review on the full truckload vehicle routing problems. In *2016 3rd international conference on logistics operations management (GOL)* (pp. 1–6). IEEE.

Bai, R., Xue, N., Chen, J., & Roberts, G. W. (2015). A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. *Transportation Research Part B: Methodological, 79,* 134–148.

Ball, M. O., Golden, B. L., Assad, A. A., & Bodin, L. D. (1983). Planning for truck fleet size in the presence of a common-carrier option. *Decision Sciences, 14,* 103–120.

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

Beck, J. C., Prosser, P., & Selensky, E. (2003). Vehicle routing and job shop scheduling: What's the difference?. In *ICAPS* (pp. 267–276).

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik, 4,* 238–252.

Berman, P., Karpinski, M., & Scott, A. D. (2007). Computational complexity of some restricted instances of 3-SAT. *Discrete Applied Mathematics, 155,* 649–653.

Bertoli, F., Kilby, P., & Urli, T. (2018). Vehicle routing problems with deliveries split over days. *Journal on Vehicle Routing Algorithms, 1,* 1–17.

Bodin, L., & Golden, B. (1981). Classification in vehicle routing and scheduling. *Networks, 11,* 97–108.

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,* 107–120.

Bunte, S., & Kliewer, N. (2009). An overview on vehicle scheduling models. *Public Transport, 1,* 299–317.

Cao, J. X., Lee, D. H., Chen, J. H., & Shi, Q. (2010). The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. *Transportation Research Part E: Logistics and Transportation Review, 46,* 344–353.

Chen, L. (2008). *Product & customer profiling for direct store delivery (DSD)*. Doctoral dissertation, Massachusetts Institute of Technology.

Codato, G., & Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research, 54,* 756–766.

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,* 1601–1615.
Emde, S., Polten, L., & Gendreau, M. (2020). Logic-based Benders decomposition for scheduling a batching machine. *Computers & Operations Research, 113,* 104777.
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,* 1315–1330.
Gallego, G., & Simchi-Levi, D. (1990). On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer R-systems. *Management Science, 36,* 240–243.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman.
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, 5,* 287–326.
Grimes, D., Hebrard, E., Malapert, A. (2009). Closing the open shop: Contradicting conventional wisdom. In *International conference on principles and practice of constraint programming* (pp. 400–408). Springer, Berlin.
Gschwind, T., Irnich, S., Tilk, C., & Emde, S. (2020). Branch-cut-and-price for scheduling deliveries with time windows in a direct shipping network. *Journal of Scheduling, 23,* 363–377.
Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics, 10,* 196–210.
Holweg, M., & Miemczyk, J. (2003). Delivering the '3-day car'-the strategic implications for automotive logistics operations. *Journal of purchasing and supply management, 9,* 63–71.
Hong, S. C., & Park, Y. B. (1999). A heuristic for bi-objective vehicle routing with time window constraints. *International Journal of Production Economics, 62,* 249–258.
Hooker, J. (2000). *Logic-based methods for optimization: Combining optimization and constraint satisfaction* (Vol. 2). Wiley.
Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research, 55,* 588–602.
Hooker, J. N., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming, 961,* 33–60.
Hsu, C. I., Hung, S. F., & Li, H. C. (2007). Vehicle routing problem with time-windows for perishable food delivery. *Journal of Food Engineering, 80,* 465–475.
IBM. (2016). What's in CPLEX Optimization Studio 12.7? Retrieved March 4, 2020 from https://developer.ibm.com/docloud/blog/2016/11/11/whats-in-cos-12-7/.
Kleywegt, A. J., Nori, V. S., & Savelsbergh, M. W. (2002). The stochastic inventory routing problem with direct deliveries. *Transportation Science, 36,* 94–118.
Kowalczyk, D., & Leus, R. (2017). An exact algorithm for parallel machine scheduling with conflicts. *Journal of Scheduling, 20,* 355–372.
Kutanoglu, E., & Mahajan, M. (2009). An inventory sharing and allocation method for a multi-location service parts logistics network with time-based service levels. *European Journal of Operational Research, 194,* 728–742.
Lam, E., Gange, G., Stuckey, P., Van Hentenryck, P., & Dekker, J. J. (2020). Nutmeg: a MIP and CP hybrid solver using branch-and-check. *SN Operations Research Forum, 1*(3). https://doi.org/10.1007/s43069-020-00023-2
Lawler, E. L. (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science, 19,* 544–546.
Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics, 1,* 343–362.
Li, H., & Womer, K. (2009). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP Benders decomposition algorithm. *Journal of Scheduling, 12,* 281–298.
Li, J. A., Wu, Y., Lai, K. K., & Liu, K. (2008). Replenishment routing problems between a single supplier and multiple retailers with direct delivery. *European Journal of Operational Research, 190,* 412–420.
Lin, L., Gen, M., & Wang, X. (2009). Integrated multistage logistics network design by using hybrid evolutionary algorithm. *Computers & Industrial Engineering, 56,* 854–873.
Lmariouh, J., El Hachemi, N., Jamali, A., Bouami, D., & Rousseau, L. M. (2019). An integrated production and distribution problem with direct shipment: A case from Moroccan bottled-water market. *International Journal of Operational Research, 34,* 144–160.
Malapert, A., Cambazard, H., Guéret, C., Jussien, N., Langevin, A., & Rousseau, L. M. (2012). An optimal constraint programming approach to the open-shop problem. *INFORMS Journal on Computing, 24*(2), 228–244.

Mathirajan, M., & Sivakumar, A. I. (2006). A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *The International Journal of Advanced Manufacturing Technology, 29,* 990–1001.

McCormack, I. M. (2014). *The military inventory routing problem with direct delivery*. MSc thesis, Air Force Institute of Technology, OH. https://scholar.afit.edu/etd/684

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.

Mönch, L., Fowler, J. W., Dauzere-Peres, S., Mason, S. J., & Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling, 14,* 583–599.

Pinedo, M. (2015). *Scheduling* (5th ed.). Springer.

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research, 120,* 228–249.

Queiser, H. (2007): Anlieferkonzepte in der Automobilindustrie – Ein internationaler Vergleich. Dokumentation – Zukunft AutomobilMontage. Retrieved February 25, 2020 from https://www.4flow.de/single-ansicht-news/article/anlieferkonzepte-in-der-automobilindustrie-ein-internationaler-vergleich.html

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research, 259,* 801–817.

Saharidis, G. K., & Ierapetritou, M. G. (2013). Speed-up Benders decomposition using maximum density cut (MDC) generation. *Annals of Operations Research, 210,* 101–123.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35,* 254–265.

Thorsteinsson, E. S. (2001). Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In *International conference on principles and practice of constraint programming* (pp. 16–30). Springer.

Tang, L., Jiang, W., & Saharidis, G. K. (2013). An improved Benders decomposition algorithm for the logistics facility location problem with capacity expansions. *Annals of Operations Research, 210,* 165–190.

Toth, P., & Vigo, D. (Eds.). (2014). *Vehicle routing: Problems, methods, and applications* (2nd ed.). Society for Industrial and Applied Mathematics.

Tzur, M., & Drezner, E. (2011). A lookahead partitioning heuristic for a new assignment and scheduling problem in a distribution system. *European Journal of Operational Research, 215,* 325–336.

Verstichel, J., Kinable, J., De Causmaecker, P., & Berghe, G. V. (2015). A combinatorial Benders' decomposition for the lock scheduling problem. *Computers & Operations Research, 54,* 117–128.

Xue, N., Bai, R., Qu, R., & Aickelin, U. (2021). A hybrid pricing and cutting approach for the multi-shift full truckload vehicle routing problem. *European Journal of Operational Research, 292,* 500–514.