Computers & Operations Research 48 (2014) 53–68

Contents lists available at ScienceDirect

Computers & Operations Research

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

The concrete delivery problem

J. Kinable [a,b,*], T. Wauters [b], G. Vanden Berghe [b]

[a] ORSTAT – KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium
[b] KU Leuven, Department of Computer Science, CODes & iMinds – ITEC, Gebr. De Smetstraat 1, 9000 Gent, Belgium

ARTICLE INFO

Available online 24 February 2014

Keywords:
Vehicle routing
Scheduling
Mixed Integer Programming
Constraint Programming
Meta-heuristics

ABSTRACT

From an operational point of view, Ready-Mixed Concrete Suppliers are faced with challenging operational problems such as the acquisition of raw materials, scheduling of production facilities, and the transportation of concrete. This paper is centered around the logistical and distributional part of the operation: the scheduling and routing of concrete, commonly known as the Concrete Delivery Problem (CDP). The problem aims at finding efficient routes for a fleet of (heterogeneous) vehicles, alternating between concrete production centers and construction sites, and adhering to strict scheduling and routing constraints. Thus far, a variety of CDPs and solution approaches have appeared in academic research. However, variations in problem definitions and the lack of publicly available benchmark data inhibit a mutual comparison of these approaches. Therefore, this work presents a more fundamental version of CDP, while preserving the main characteristics of the existing problem variations. Both exact and heuristic algorithms for CDP are proposed. The exact solution approaches include a Mixed Integer Programming (MIP) model and a Constraint Programming model. Similarly, two heuristics are studied: the first heuristic relies on an efficient best-fit scheduling procedure, whereas the second heuristic utilizes the MIP model to improve delivery schedules locally. Computational experiments are conducted on new, publicly accessible, data sets; results are compared against lower bounds on the optimal solutions.

1. Introduction

The Concrete Delivery Problem (CDP), a combination of a scheduling and a routing problem, encompasses the distribution of concrete to a number of construction sites. Each construction site $i \in C$, also referred to as customer, requests q_i tons of concrete. The concrete is transported by a non-homogeneous set of trucks K, each capable of delivering q_k, $k \in K$, tons of concrete. A time window $[a_i, b_i]$ is associated with each construction site $i \in C$, denoting the time interval during which the concrete can be delivered. Deliveries cannot commence before a_i and should be completed before b_i. Due to the hard time windows and capacity limitations of the trucks, it is often impossible to satisfy the demand of all customers (oversubscribed scheduling problem).

The amount of concrete requested by a single customer typically exceeds the capacity of a single truck. Hence, multiple deliveries may be required. Deliveries for the same customer cannot overlap in time, and have to take a maximum time lag γ into consideration; the time between two consecutive deliveries is limited to guarantee proper bonding between the two layers of concrete. Deliveries may not be preempted or split amongst multiple customers. The time required to perform a delivery is truck dependent, and will be denoted by p_k, $k \in K$.

The concrete is produced at several homogeneous production sites (P) which are located some distance away from the customers. The trucks start at a central depot and have to travel to a production plant, after which they have to unload their cargo at one of the construction sites. Loading the concrete and traveling between a pickup and delivery point $i \in P, j \in C$ takes a certain amount of time t_{ij}. Trucks are always filled to their maximum capacity. Whenever the total amount of concrete delivered at a construction site exceeds the customer's demand, the excessive amount becomes waste. A summary of the notation used throughout the paper is provided in Table 1.

The objective of the problem is to maximize the number of satisfied customers, weighted by their demand. A customer $i \in C$ is satisfied if at least q_i tons of concrete have been delivered. Since a concrete distributor has only finite capacity, it is natural that some customers cannot be serviced. Rejected customers are supposed to relax their request or to revert to a different supplier. Concrete distributors also have the option to temporarily increase capacity

* Corresponding author at: ORSTAT – KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium.
E-mail addresses: joris.kinable@kuleuven.be (J. Kinable), tony.wauters@cs.kuleuven.be (T. Wauters), greet.vandenberghe@cs.kuleuven.be (G. Vanden Berghe).

http://dx.doi.org/10.1016/j.cor.2014.02.008
0305-0548

by hiring a subcontractor. Although not considered in this work, it would be straightforward to include services by subcontractors at higher cost in the objective function.

The CDP bares a strong resemblance with two well-studied NP-hard problems: the Capacitated Vehicle Routing Problem (CVRP) with Time Windows and Split Deliveries and the Parallel Machine Scheduling Problem (PMSP) with Time-Windows. Although efficient exact and heuristic algorithms have been presented in the literature for both problems individually, it is not evident how to generalize these methods to CDP. In related works (see Section 2 for details), dedicated approaches are published for several variations of the CDP. Each variation has its own unique constraints, including constraints dealing with different types of concrete, equipment requirements (pumps and vehicle types), and customer demands (origin of the concrete and delivery speed). Next to the fact that some authors conduct their experiments on confidential data, the variations in their problem descriptions render it notoriously difficult to mutually compare their algorithms. This work addresses a more fundamental version of the CDP, which captures the commonalities amongst the existing variations of the problem while dropping some of the more exotic constraints. Nevertheless, the exact and heuristic approaches presented in this work are easily modified to incorporate any of the omitted constraints. To facilitate a future comparison of CDP implementations, we publish a large number of problem instances.

The paper is structured as followed. First, Section 2 provides a detailed overview of related research. Next, in Section 3, three mathematical models for CDP are presented. The first model, based on Mixed Integer Programming (MIP), is derived from models for the CVRP and the PMSP. The second model extends the first model by incorporating the requirement that a vehicle may perform multiple deliveries to the same customer. Finally, the third model, as an alternative to the second model, is based on Constraint Programming. Section 4 proposes two heuristic approaches for the CDP. The first heuristic utilizes an efficient constructive procedure, which schedules customers one-by-one, following a best-fit policy. A meta-heuristic controls the exact order the customers are considered for scheduling by the constructive procedure. The second heuristic relies on the MIP model presented in Section 3. Instead of solving the entire model at once, the heuristic repeatedly fixes one part of the schedule while re-optimizing another part. To compare the various methods discussed in this work, Section 5 outlines two approaches to compute bounds on the optimal solutions. Computational results are presented in Section 6. Finally, Section 7 offers the conclusion.

Table 1
Parameters defining the CDP.

Parameter	Description
P	Set of concrete production sites
C	Set of customers
$0, n+1$	resp. the start and end depots of the trucks
V	$V = P \cup C \cup \{0\} \cup \{n+1\}$
K	Set of trucks
q_i	Requested amount of concrete by customer $i \in C$
q_k	Capacity of truck $k \in K$
p_k	Time required to empty truck $k \in K$
a_i, b_i	Time window during which the concrete for customer i may be delivered
t_{ij}	Time to travel from i to j, $i, j \in V$
γ	Maximum time lag between consecutive deliveries.

2. Related research

A number of related works involving several variations of the CDP exist in the literature. To obtain insight in the diversity in problem specifications, as well as in the solution approaches, we provide an extensive overview of the most recent works in this area. Table 2 summarizes for each of these papers the main problem characteristics and solution approaches. The abbreviations used in the table are explained in Appendix A. Omitted values indicate that a certain property is not applicable, is unknown or is not considered in the paper.

Schmid et al. [8] propose a hybrid heuristic for a CDP with soft time windows, and obligatory deliveries. The heuristic resembles a traditional column generation procedure (CGP, for details refer to e.g. Desaulniers et al. [10]). The columns (delivery patterns), describe feasible orderings of deliveries to satisfy specific customers. More precisely, a pattern describes exactly which vehicles perform deliveries for a customer, and when the deliveries should commence. A MIP approach, comparable to the master problem (MP) in a CGP, selects a delivery pattern for each customer in such a way that the selected patterns together form a feasible solution, thereby satisfying all routing and scheduling constraints. Next, in contrast to a traditional CGP where a pricing problem is utilized to generate new columns for the MP, new columns are created via a Variable Neighborhood Search (VNS) procedure which is initialized with the solution produced by the MIP model. The VNS procedure perturbates and re-optimizes the MIP solution until a new feasible schedule is obtained which is different from the original MIP solution. The resulting schedule is decomposed into delivery patterns and added to the set of existing patterns in the MIP. Finally, the MIP model is resolved and the procedure repeats until a termination criterion is met.

An alternative procedure to Schmid et al. [8] has been published by Schmid et al. [9]. A VNS heuristic, very similar to the one from Schmid et al. [8], is deployed to generate an initial solution. Then, the solution is inserted into a MIP model. Instead of solving the entire MIP model at once, a large number of variables are fixed to the values of the initial solution; only a small portion of the variables remain unfixed. The exact variables to fix are determined

Table 2
Summary of the various CDP problems and solution approaches in related literature. For notation, refer to Appendix A.

Article	Solution method	Time windows	Start/end location	Production depots	Loading, unloading	Fleet	Instrumentation	Deliveries	Objectives
Hertz et al. [1]	m	sv	c	he	d	he		v,l,sp	u, v_s, v_c
Misir et al. [2]	he	hs,p	p	he,s	d	he	✓	s,r,v,l,sp	t,w,d
Silva et al. [3]	he	hd,p	p	ho	d	ho		l,sp	u,o
Asbach et al. [4]	m,he	hd,p	m	he	d	he	✓	s,r,v,l,sp,ss	u,s
Yan and Lai [5]	m	sd,p	p		f	ho		s,r,l,sp	t,d,op
Lin et al. [6]	m	hd,sv,p	p	ho	d	ho		s,r,l,sp	t,d,u,b,s
Naso et al. [7]	m,he	sd,p	p	he,s	d	ho		s,r,l,sp,ss	u,o,d
Schmid et al. [8]	hy,he	sd	p	ho	d	he	✓	s,r,l,sp	t,d
Schmid et al. [9]	m,hy	sd	p	ho	d	he	✓	s,r,l,sp	t,d
This work	m,cp,he,hy	hd	c	ho	f	he		s,r,l,sp	s

by a Very Large Neighborhood Search procedure. The MIP model is solved repeatedly, having different decision variables fixed and unfixed, always starting from the best incumbent solution. When compared mutually, Schmid et al. [9] performs slightly better than Schmid et al. [8] on small and medium sized instances, ranging from 27 to 60 orders.

Hertz et al. [1] discuss CDP with a strong emphasis on routing and vehicle constraints. Scheduling of the individual deliveries at the customers' side is not taken into account. Next to an exact MIP model, Hertz et al. [1] propose a (heuristic) decomposition of their problem into two MIP subproblems: an assignment and a routing problem. First, the assignment problem is solved, thereby assigning deliveries to vehicles such that all customers are satisfied. Next, for each vehicle, a routing problem is solved, determining the optimal route to perform the deliveries. To improve the solution quality, estimations of the travel costs incurred when assigning a specific delivery to a particular vehicle are added to the assignment problem.

Lin et al. [6] present a MIP model, using a time-indexed notation. Some experiments are conducted and compared against man-made schedules from a Taiwanese RMC production company. No results with respect to computation times, bounds or comparative studies have been included; it is unclear whether the MIP model solved the problem instance(s) to optimality.

A generalization of the present work is published by Asbach et al. [4]. They formally define the problem using a MIP model, and establish that their problem is NP-Hard by reduction to the Euclidean TSP problem. Next, a heuristic is proposed to solve the CDP as solving the MIP model is considered intractable. The heuristic, initialized with a feasible schedule, iteratively un-schedules and re-schedules one or two customers simultaneously. A (greedy) constructive procedure is used to reinsert the unscheduled customers into the schedule, in an attempt to obtain a better solution. This procedure is repeated until a local optimum is reached, or a pre-defined time limit is exceeded. The authors do not publish bounds on the optimal solutions, and the problem instances are inaccessible due to a non-disclosure agreement, rendering a fair comparison to this work virtually impossible.

Another variation of the CDP is presented by Silva et al. [3] where the main focus is shifted to the production centers. A Genetic Algorithm (GA) is used to assign customer orders to concrete production centers. The orders are subdivided into a number of 'jobs'. Trucks located at the production sites are employed to deliver the concrete in a timely manner. Assigning deliveries to trucks is performed via an Ant Colony Optimization algorithm.

Naso et al. [7] expand upon the work by Silva et al. [3], adding a significant number of additional hard and soft constraints to the problem. Again a GA is employed to assign orders originating from construction sites to nearby production centers. Each production center is equipped with a single loading dock, where the produced concrete is loaded into trucks. Since each dock can only handle a single truck at a time, sequencing of the trucks is required. This is performed via a constructive heuristic. Once a feasible loading sequence has been determined for each production center, another constructive heuristic is deployed to determine the schedules and routes for the available fleet of vehicles. This schedule must ensure that a vehicle is ready to transport the concrete to the customers' site as soon as the concrete for the delivery is available at the loading dock. If no vehicle can accommodate the delivery, an external vehicle is hired and a penalty is incurred. Computational experiments are performed on data from a Dutch concrete supplier. The performance of the heuristic is compared against four different scheduling policies utilized by human schedulers; no comparisons against Silva et al. [3] are included in the paper.

Yan and Lai [5] consider a heuristic approach to a CDP consisting of a single production station and a homogeneous fleet. Next to the travel time of vehicles, the authors include overtime of deliveries, and operation time of the production station and construction sites in their objective function. A MIP model is provided, modeled on a time-space network: nodes in the network encode both time and location information, whereas arcs represent translations in the time and space domains. A disadvantage of such a model is the fact that the continuous time domain is discretized in fixed intervals; taking the time intervals too small leads to an excessive number of nodes in the network, whereas large time intervals lead to inaccurate or suboptimal solutions. Solving the model using a MIP solver turned out intractable for their real-world problem instances. Hence, a heuristic decomposition approach is proposed instead. This approach first solves a relaxed version of the MIP model. The resulting solution is then used to simplify the original MIP model by fixing a number of the decision variables to their corresponding values in the relaxed solution. Finally, the simplified MIP model is solved. The authors claim that the resulting decomposition yields very good results when compared against a lower bound obtained by a MIP solver, or solutions produced by human schedulers. However, it must be noted that their dataset is limited to just three instances; it is unclear how any related problems or approaches would fair upon such a decomposition.

Misir et al. [2] propose a new hyper-heuristic to the CDP. The hyper-heuristic is initialized with both a feasible CDP solution and a set of heuristics. At each iteration, the hyper-heuristic selects a heuristic which is used to construct a new CDP solution based on the solution obtained during the previous iteration. The quality of the new solution is established, and the hyper-heuristic decides whether the new solution is accepted or rejected. The authors compare their work against four alternative hyper-heuristics.

In contrast to the previous, mainly Operations Research based approaches, Graham et al. [11] treat CDP from a Machine Learning point of view. A neural network is used to predict the delivery rate of concrete at a production site, based on a set of known parameters such as the capacities of the trucks, their average inter arrival time, designation of the concrete, season, number of accepted and rejected deliveries, etc. Data from previous construction projects is used to train the model. Results from this approach may be used to improve the work flow, planning and resource utilization. As this work conceptually deviates from the other papers discussed in this section, we did not include it in Table 2.

3. Mathematical models

3.1. Integer Programming models

This section introduces two MIP models for CDP. The first one, CDP1, in Section 3.1.1 shows how two well-known models for the Capacitated Vehicle Routing Problem (CVRP) with Time Windows and Split Deliveries, and the Parallel Machine Scheduling Problem (PMSP) with Time Windows and Maximum Time Lags can be combined to model the CDP. Although the resulting model supports split-deliveries, all deliveries for a single customer should be made by different vehicles; it is not possible that a single vehicle performs two or more deliveries for the same customer. A model (CDP2) which overcomes this limitation is presented in Section 3.1.2.

3.1.1. An Integer Programming model for CDP

The concrete trucks start their trip at a central (source) depot and travel between production sites and customers. At the end of the day, the trucks return to an end (sink) depot (which may or may not be the same as the starting depot). This routing problem can be modeled on a directed, weighted graph $G(V,A)$, consisting of vertex set $V=\{0\}\cup C\cup\{n+1\}$, where vertices 0 and $n+1$ are resp. the source and sink depots. Vertices representing the production depots are omitted. Instead, the fact that a vehicle has to reload in between two deliveries has been accounted for in

the arc costs. Optionally, a positive load time for the vehicles can be added to the arcs between two customers. The arc set A is defined as follows:

- The source, sink depots have outgoing resp. incoming edges to/from all other vertices.
- There is an arc (i,j) for all $i,j \in C, i \neq j$.

The arc costs are as follows:

- $c_{0,i} = \min_{p \in P} t_{0,p} + t_{p,i}$ for all $i \in C$.
- $c_{i,j} = \min_{p \in P} t_{i,p} + t_{p,j}$ for all $i,j \in C$.
- $c_{i,n+1} = t_{i,n+1}$.
- $c_{0,n+1} = 0$.

The CDP is very similar two well known combinatorial optimization problems: the CVRP with Time Windows and Split Deliveries, and the PMSP with Time Windows and Maximum Time Lags. However, there is a fundamental difference between the CDP and CVRP with split deliveries: in CVRP with split deliveries, the capacity of a single vehicle must be shared amongst all the customers visited by the vehicle, whereas this is not the case in CDP. Nevertheless, the model for a Capacitated Vehicle Routing Problem as presented by Desaulniers et al. [10] can be modified to meet the routing constraints of CPD as follows:

$$\max. \quad \sum_{i \in C} q_i y_i \tag{1}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} q_k x_{ijk} \geq q_i y_i \qquad \forall i \in C \tag{2}$$

$$\sum_{j \in \delta^+\{0\}} x_{0jk} = \sum_{i \in \delta^-\{n+1\}} x_{i,n+1,k} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{j \in \delta^+(i)} x_{i,j,k} = \sum_{j \in \delta^-(i)} x_{j,i,k} \qquad \forall i \in C, k \in K \tag{4}$$

$$C^i_k + t_{ij} - M(1 - x_{ijk}) \leq C^j_k - p_k \qquad \forall i,j \in V, k \in K \tag{5}$$

$$a_i + p_k \leq C^i_k \leq b_i \qquad \forall i \in V, k \in K \tag{6}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in V, k \in K \tag{7}$$

$$C^i_k \in \mathbb{Z}_{\geq 0} \qquad \forall i \in V, k \in K \tag{8}$$

$$y_i \in \{0,1\} \qquad \forall i \in C \tag{9}$$

In this formulation, denoted as CDP-route, x_{ijk} is a binary variable indicating whether vehicle k travels from i to j. Integer variables C^i_k, $i \in C, k \in K$, record the time that vehicle k finishes its delivery to customer i. Binary variable y_i denotes whether customer $i \in C$ is serviced. For notation purposes, $\delta^-(\cdot)$ resp. $\delta^+(\cdot)$ denote the incoming resp. outgoing neighborhood sets. Constraint (2) ensures that sufficient concrete is delivered at construction site. Constraints (3) and (4) determine the shape of a feasible tour: a tour starts at the source depot, visits a number of pickup and delivery points and finally returns to the sink depot. Constraints (4) are the flow preservation constraints. A customer $i \in C$ cannot be visited before a_i, and deliveries have to be completed before b_i (Constraint (6)). Furthermore, between two consecutive visits, starting, processing and travel times have to be taken into account (Constraint (5)). The remaining constraints restrict the domains of the variables.

Although CDP-route captures many of the requirements of CDP, it cannot express the maximum time lag requirement between multiple deliveries for a single customer. The later concept however can be modeled using a flow based formulation originating in the PMSP problem. To this extent, the model of Bard and Rojanasoonthon [12] is adapted, thereby obtaining CDP-schedule:

$$\max. \quad \sum_{i \in C} q_i y_i \tag{10}$$

$$\sum_{k \in K} \sum_{l \in K_0 \setminus \{k\}} q_k z^i_{kl} \geq q_i y_i \qquad \forall i \in C \tag{11}$$

$$\sum_{l \in K_0} z^i_{k_0 l} = 1 \qquad \forall i \in V \tag{12}$$

$$\sum_{l \in K_0 \setminus \{k\}} z^i_{kl} = \sum_{l \in K_0 \setminus \{k\}} z^i_{lk} \qquad \forall k,l \in K, k \neq l, i \in V \tag{13}$$

$$C^i_k - M(1 - z^i_{kl}) \leq C^i_l - p_l \qquad \forall k,l \in K, k \neq l, i \in V \tag{14}$$

$$C^i_l - p_l \leq C^i_k + \gamma + M(1 - z^i_{kl}) \qquad \forall k,l \in K, k \neq l, i \in V \tag{15}$$

$$a_i + p_k \leq C^i_k \leq b_i \qquad \forall i \in V, k \in K \tag{16}$$

$$z^i_{kl} \in \{0,1\} \qquad \forall i \in V, k,l \in K \tag{17}$$

$$C^i_k \in \mathbb{Z}_{\geq 0} \qquad \forall i \in V, k \in K \tag{18}$$

$$y_i \in \{0,1\} \qquad \forall i \in C \tag{19}$$

The binary variable z^i_{kl} is equal to one if truck $k \in K$ delivers its payload immediately before truck $l \in K$ to customer $i \in C$, zero otherwise. In this model, k_0 represents a dummy truck, $K_0 = K \cup \{k_0\}$.

Constraint (11) ensures that sufficient concrete is delivered to each customer. Constraints (12) and (13) sequence the trucks. A truck can only be used once for every customer, and whenever it is used, its delivery must be succeeded by another delivery (possibly a delivery by the dummy truck k_0) (Constraint (12)). The dummy truck must be scheduled (Constraint (12)). Constraints (13) are the flow preservation constraints. Together with Constraints (12), these constraints enforce that all deliveries are scheduled on a ring. Each delivery has exactly one successor and one predecessor. Constraints (14) link the completion time variables C^i_k and the sequence variables z^i_{kl}, thereby enforcing that deliveries do not overlap in time. Additionally, Constraints (15) enforce a maximum lag time between consecutive deliveries. Finally, Constraints (16) ensure that deliveries are scheduled within the customer's time window.

A feasible solution to CDP is obtained at the intersection of the CPD-route and CPD-schedule polytopes. Connecting the two polytopes is accomplished via the linking constraints:

$$\sum_{l \in K_0 \setminus \{k\}} z^i_{kl} = \sum_{j \in N} x_{ijk} \qquad \forall k \in K, i \in C \tag{20}$$

Note that Constraints (11) and (16) in CDP-route are identical to Constraints (2) and (6) in CDP-schedule respectively, and are dropped as a consequence. After removing redundant constraints and rewriting some constraints via the linking constraints, we arrive at a model for CDP which we denote as CDP1.

3.1.2. An Integer Programming model for CDP with revisits

A disadvantage of CDP1 is that a single truck cannot visit the same customer more than once. This restriction is unrealistic as often a small number of concrete trucks drive back and forth between a production depot and a construction site. It is however not obvious how to efficiently lift this restriction in CDP1. Clearly, when the binary variables x_{ijk} are replaced by equivalent integer variables indicating the number of times vehicle k travels from i to j, one can still distinguish the routes, but expressing the scheduling constraints becomes difficult. Two options exist: either the distinct trips made by a single vehicle are enumerated (e.g. vehicle k travels from i to j during trip t), or the deliveries to a specific customer are enumerated. The latter solution is applied in assignment-based formulations for scheduling problems (see e.g. Berghman et al. [13]). An alternative IP formulation for CDP

using this approach is presented by Asbach et al. [4]. This model, simplified and adjusted to our notation, is given below. Let the sets P, C be defined as before. In addition, for each customer $i \in C$, a new ordered set consisting of deliveries, $C^i = \{1, \ldots, n(i)\}$, is defined, where $n(i) = \lceil q_i / \min_{k \in K}(q_k) \rceil$ is an upper bound on the number of deliveries required by customer i. As shorthand notation, c^i_j will be used to denote delivery j for customer i. A time window $[a_u, b_u]$ is associated with each delivery $u \in C^i, i \in C$, which is initialized to the time window of the corresponding customer $i \in C$, i.e. $[a_u, b_u] = [a_i, b_i]$ for all $i \in C, u \in C_i$. Finally, $D = \bigcup_{i \in C} C_i$ is the union of all deliveries.

Let $G(V, A)$ be the directed, weighted graph consisting of vertex set $V = \{0\} \cup D \cup \{n+1\}$. The arc set A is defined as follows:

- the source, sink depots have outgoing resp. incoming edges to/from all other vertices.
- a delivery node c^i_h has a directed edge to a delivery node c^i_j if $h < j$, $i \in C$, $h, j \in C_i$.
- there is a directed edge from c^i_u to c^j_v, $i \neq j$, except if c^j_v needs to be scheduled earlier than c^i_u.

The arc costs are as follows:

- $c_{0,c^i_j} = \min_{p \in P} t_{0,p} + t_{p,i}$ for all $c^i_j \in D$.
- $c_{c^i_u, c^j_v} = \min_{p \in P} t_{i,p} + t_{p,j}$ for all $c^i_u, c^j_v \in D, c^i_u \neq c^j_v$.
- $c_{c^i_j, n+1} = t_{i,n+1}$.
- $c_{0,n+1} = 0$.

The entire model, entitled CDP2, becomes

$$\max. \quad \sum_{i \in C} q_i y_i \tag{21}$$

$$\sum_{j \in \delta^+(0)} x_{0jk} = \sum_{i \in \delta^-(n+1)} x_{i,n+1,k} = 1 \qquad \forall k \in K \tag{22}$$

$$\sum_{j \in \delta^-(i)} x_{j,i,k} = \sum_{j \in \delta^+(i)} x_{i,j,k} \qquad \forall i \in D, k \in K \tag{23}$$

$$S(i,1) \leq 1 \qquad \forall i \in D \tag{24}$$

$$S(j+1,1) \leq S(j,1) \qquad \forall i \in C, j \in \{1, \ldots, n(i)-1\} \tag{25}$$

$$\sum_{j \in C_i} S(j, q_k) \geq q_i y_i \qquad \forall i \in C \tag{26}$$

$$C^i - M(1 - x_{ijk}) \leq C^j - p_k - c_{ij} \qquad \forall (i,j) \in A, i \neq 0, k \in K \tag{27}$$

$$C^i - M(1 - x_{ijk}) \leq C^j - c_{ij} \qquad \forall (0,j) \in A, k \in K \tag{28}$$

$$C^i - S(i, p_k) \geq a_i \qquad \forall i \in D \tag{29}$$

$$C^{j+1} - S(j+1, p_k) - C^j \leq \gamma \qquad \forall i \in C, j \in \{1, \ldots, n(i)-1\} \tag{30}$$

$$C^{j+1} \geq C^j + S(j, p_k) \qquad \forall i \in C, j \in \{1, \ldots, n(i)-1\} \tag{31}$$

$$a_i \leq C^i \leq b_i \qquad \forall i \in V \tag{32}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A, k \in K \tag{33}$$

$$y_i \in \{0,1\} \qquad \forall i \in C \tag{34}$$

Here, $S(i, \alpha) = \sum_{k \in K} \sum_{j \in \delta^+(i)} \alpha x_{ijk}$ for all $i \in D$. Binary variables x_{ijk} denote whether vehicle $k \in K$ travels from i to j, $i, j \in V$. Binary variables C^i record the time that delivery $i \in D$ is completed. In addition, C^{n+1} records the makespan of the schedule. Finally, boolean variables y_i, denote whether customer $i \in C$ is serviced.

Constraints (22)–(24) are the common vehicle routing constraints, defining respectively the starting and ending location of a tour, flow preservation, and the number of times a delivery can be made. Furthermore, Constraints (25) orders the deliveries: a delivery c^i_{j+1} cannot me performed whenever delivery c^i_j has not been made. This constraint, in conjunction with Constraint (30) implements the maximum time lag between consecutive deliveries. The sum of capacities of the vehicles performing the deliveries for customer $i \in C$ should cover the customer's demand (Constraint (26)). Constraints (27)–(32) enforce the necessary scheduling restrictions. A delivery cannot be made outside the customer's time window (Constraints (29) and (32)), travel times need to be accounted for (Constraints (27) and (28)). Finally, Constraints (31) ensure that deliveries to the same customer do not overlap in time.

Although this formulation allows a single vehicle to visit customers more than once, it has the clear disadvantage that each customer is represented by a possibly large set of customer nodes, especially when the bound $n(i)$, $i \in C$, is weak. One solution is to artificially limit the maximum value of $n(i)$, but this potentially cuts off the optimal solution.

3.1.3. A note on complexity

Theorem 1. *The CDP as defined by CDP2 (Section 3.1.2) is NP-Hard.*

Proof. The proof follows by reduction to the Hamiltonian Cycle (HC) problem. Let P be an instance of the HC problem, defined on the simple, undirected, not necessarily complete graph $G(V, E)$ with vertex set V and edge set $E \subseteq V \times V$. Finding a HC in G is NP-Complete [14]. Let P' be an instance to CDP, as defined in CDP2 (Section 3.1.2). Instance P' has only one vehicle k_1, having capacity $q_{k_1} = 1$ and processing time $p_{k_1} = 0$. The vertex set of P', say V', is identical to the vertex set of P, i.e. $V = V'$. Designate an arbitrary vertex in V' as depot; this vertex serves both as starting and ending depot. The remaining vertices become customers, each having a demand of 1. Consequently, each customer can be satisfied with a single delivery. The arc set A of the routing graph $G'(V', A)$, is defined as follows. There are arcs $(i,j), (j,i)$ in A if there is an edge $(i,j) \in E$. All arc weights are set to 1, the time windows for each customer are set to $[0, |V|]$, and the time lag $\gamma = 1$. Note that neither the time windows nor the time lag pose any restrictions on the delivery schedule as all time windows are sufficiently wide, and each customer only requires a single delivery. Finally, it is straightforward to show that if there exists a HC in P, then there must exist a solution to P' satisfying the demands of all customers. As each delivery for each customer may only be made once (Constraint 24), the resulting delivery route for k_1 must be a HC.

3.2. Constraint Programming model

Next to MIP, Constraint Programming (CP) offers an alternative means to model the CDP. Unfortunately, no universal CP language exist; the syntax and expressiveness is solver dependent. In the discussion below, we only use constraints present in IBM's CP optimizer. However, equivalent versions of these constraints exist in several other CP solvers.

Our model utilizes interval variables [15,16]. An interval variable represents an interval during which an activity is performed. An activity, and its corresponding interval variable may be optional, i.e. there is no obligation to schedule the activity. More formally, an interval variable α is a variable where domain $dom(\alpha)$ is a subset of $\{\perp\} \cup \{[s,e) | s, e \in \mathbb{Z}, s \leq e\}$. An interval variable is fixed if its domain is reduced to a singleton, i.e. if α denotes a fixed interval variable:

- $\alpha = \perp$ if the interval is absent; the activity is not scheduled.
- $\alpha = [s,e)$ if the interval is present.

An absent interval variable is ignored by any constraint or expression. For example, a constraint specifying that an interval α needs

to end before the start of an interval β is always satisfied if either of the intervals is absent.

Each interval variable α has a start time $startOf(\alpha)$, an end time $endOf(\alpha)$, and a duration $dur(\alpha)$. Whenever an interval is present, $endOf(\alpha) - startOf(\alpha) \geq dur(\alpha)$ must hold. As shorthand notation, an interval variable α is defined as a tuple: $\alpha = \{r, d, t, o\}$, specifying respectively the earliest start time of the interval, latest end time, minimum duration, and whether the interval is optional or obligatory. The constraints used in this model are summarized in Table 3.

Algorithm 1. CP model for CDP.

```
   Variable definitions:
1  s = {0, 0, 0, oblig.};
2  t = {0, ∞, 0, oblig.};
3  c_i = {r_i, d_i, 0, opt.}        ∀ i ∈ C;
4  d^i_j = {r_i, d_i, 0, opt.}      ∀ i ∈ C, j ∈ {1, ..., n(i)};
5  d^i_{j,k} = {r_i, d_i, p_k, opt.}   ∀ i ∈ C, j ∈ {1, ..., n(i)}, k ∈ K;
   Objective:;
6  Max ∑_{i∈C} q_i · presenceOf(c_i);
   Constraints:;
7  forall i ∈ C
8  |  span(c_i, ⋃_{j∈n(i)} d^i_j);
9  |  forall j ∈ {1, ..., n(i)}
10 |  |alternative(d^i_j, ⋃_{k∈K} d^i_{jk});
11 |  presenceOf(c_i) == (∑_{k∈K} ∑_{j∈n(i)} q_k · presenceOf(d^i_{jk}) ≥ q_i);
12 |  forall {j ∈ {1, ..., n(i) − 1}
13 |  |  endBeforeStart(d^i_j, d^i_{j+1});
14 |  |  startBeforeEnd(d^i_{j+1}, d^i_j, −γ);
15 |  |  presenceOf(d^i_{j+1}) → presenceOf(d^i_j);
16 |  forall j ∈ {1, ..., m(i)}
17 |  |presenceOf(c_i) == presenceOf(d^i_j);
18 |  forall j ∈ {m(i), ..., n(i) − 1}
19 |  |presenceOf(c_i) ∧ (∑_{l∈{1 .. j}, k∈K} q_k · presenceOf(d^i_{lk}) < q_i) → presenceOf(d^i_{j+1});
20 forall k ∈ K
21 |  noOverlapSequence(⋃_{i∈C, k∈K} d^i_{jk} ∪ s ∪ t);
22 |  first(s, ⋃_{i∈C, k∈K} d^i_{jk} ∪ t);
23 |  last(t, ⋃_{i∈C, k∈K} d^i_{jk} ∪ s);
```

The CP model (Algorithm 1) relies on three hierarchical levels of optional interval variables which are linked via the span and alternative constraints (lines 8 and 10):

- A variable c_i, for each customer $i \in C$ (line 3).
- A variable d^i_j for each possible delivery $j \in \{1, \ldots, n(i)\}$ to customer $i \in C$ (line 4).
- A variable $d^i_{j,k}$ for each vehicle $k \in K$ that may perform a specific delivery $j \in \{1, \ldots, n(i)\}$ for customer $i \in C$ (line 5).

The Constraint on line 11 states the relation between a customer and the customer's deliveries: a customer may only be scheduled if sufficient concrete is delivered. The constraints 13–15 take care of the scheduling requirements. Finally, Constraints 21–23 deal with vehicle routing restrictions: for each vehicle, a Hamiltonian path is created which starts and ends at resp. the source depot s and the sink depot t, while respecting travel times between the deliveries.

To improve constraint propagation, two implicit sets of constraints are added. Let $m(i) = \lceil q_i / \max_{k \in K}(q_k) \rceil$ be a lower bound on the number of deliveries required to satisfy customer $i \in C$. The constraint on line 17 states that if customer c_i, $i \in C$, is present, then at least $m(i)$ deliveries must be made. The constraint on line 19 ensures that if the total amount of concrete delivered in the first j deliveries to customer i does not suffice, at least one more delivery is made.

4. Heuristic models

4.1. Steepest Descent and best fit

The first heuristic is based on a steepest descent local search and a best fit constructive procedure. The constructive heuristic (Section 4.1.1) schedules the visits to the costumers one-by-one at the 'best' possible position, determined by several heuristic criteria, e.g. the start time of the visit, and the capacity of the vehicle. The order π the customers are processed by the constructive procedure is controlled by a Steepest Descent heuristic (Section 4.1.2).

4.1.1. Best fit constructive heuristic

Algorithm 2 illustrates the best fit constructive heuristic. The following functions, omitting the implementation details for clarity, are used:

- *satisfied*$(\mathcal{S}, i)$: returns whether customer $i \in C$ is satisfied in solution $\mathcal{S}$, i.e. whether sufficient concrete is delivered for this customer.
- *earliestStartTimeOfVisit*$(\mathcal{S}, k, i)$: searches for the earliest moment in time vehicle $k \in K$ can schedule the next delivery to customer $i \in C$ given the current solution $\mathcal{S}$, and while taking travel times into consideration.
- *wasteOfVisit*$(\mathcal{S}, k, i)$: returns the amount of *waste*, $0 \leq waste < q_k$, a visit by vehicle $k \in K$ would introduce to customer $i \in C$, given the current solution $\mathcal{S}$.
- *startOfPreviousVisit*$(\mathcal{S}, i, t)$: returns the start time of the previous visit to customer $i \in C$ in $\mathcal{S}$. If this is the first visit to customer i then the value t is returned.
- *scheduleVisit*$(\mathcal{S}, i, bestK, start)$: schedules a visit to customer i in solution $\mathcal{S}$ using vehicle *bestK* at time *start*.
- *applyChaining*$(\mathcal{S}, i)$: tries to shift all previous scheduled visits for customer $i \in C$ in solution $\mathcal{S}$ to a later point in time, taking into account the maximum timelag γ and the customer deadline b_i.
- *clean*$(\mathcal{S})$: removes all the visits of unsatisfied customers from solution $\mathcal{S}$.

The algorithm iterates over all costumers in π. For each customer it searches for the best vehicles to schedule the customer's visits. First of all, to schedule a new visit, a feasible start time must be found which respects, (1) the travel times, (2) the maximum time lag γ with respect to the previous visit, and (3) the customer's deadline b_i. The best vehicle to perform a visit is selected by the following (ordered) list of criteria:

1. earliest available vehicle, counting from the moment the delivery may commence;
2. minimize waste, i.e. the amount that the vehicle's capacity surpasses the remaining demand of the customer; and
3. in case of a draw, select the largest vehicle.

These criteria have been obtained experimentally.

Algorithm 2. The best fit construction heuristic.

input: A permutation of customers π
output: A feasible CDP solution $\mathcal{S}$
Start with empty solution $\mathcal{S}$;
foreach $i \in \pi$ **do**/* `For each customer` */
  found $\leftarrow$ *TRUE*;
  while *found* ***and*** $\neg$`satisfied`$(\mathcal{S}, i)$ **do**
    found $\leftarrow$ *FALSE*;
    minStart $\leftarrow$ *MAXVALUE*;
    minWaste $\leftarrow$ *MAXVALUE*;
    bestK $\leftarrow -1$;
    foreach $k \in K$ **do**/* `For each vehicle` */
      start $\leftarrow$ `earliestStartTimeOfVisit`$(\mathcal{S}, k, i)$;
      waste $\leftarrow$ `wasteOfVisit`$(\mathcal{S}, k, i)$;
      prevStart $\leftarrow$ `startOfPreviousVisit`$(\mathcal{S}, i, start)$;
      if $start \leq prevStart + \gamma$ ***and*** $start + p_k \leq b_i$ **then**
        found $\leftarrow$ *TRUE*;
        newBest $\leftarrow$ *FALSE*;
        if $start < minStart$ **then**
          newBest $\leftarrow$ *TRUE*;
        else if $start == minStart$ ***and*** $waste < minWaste$ **then**
          newBest $\leftarrow$ *TRUE*;
        elseif $start == minStart$ ***and*** $waste == minWaste$ ***and*** $q_k > q_{bestK}$ **then**
          newBest $\leftarrow$ *TRUE*;
        if *newBest* **then**
          minStart $\leftarrow$ *start*;
          minWaste $\leftarrow$ *waste*;
          bestK $\leftarrow k$;
    if *found* **then**
      `scheduleVisit`$(\mathcal{S}, i, bestK, minStart)$;
    else
      `applyChaining`$(\mathcal{S}, i)$;
`clean`$(\mathcal{S})$;

Table 3
Description of CP constraints.

Constraint	Description
presenceOf(α)	States that interval α must be present.
span(α, B)	Interval α (if present) spans over all present intervals from the set of intervals B, i.e. the start and end of α coincides with resp. the first present interval and the last present interval in B. If α is not present, then neither are the intervals in B.
alternative(α, B)	If interval α is present, then exactly one of the intervals in set B is present. The start and end of interval α coincides with the start and end of the selected interval from set B.
endBeforeStart(α, β)	$endOf(\alpha) \leq startOf(\beta)$. Automatically satisfied if either of the intervals is absent.
startBeforeEnd(α, β, z)	$startOf(\alpha)+z \leq endOf(\beta)$. Automatically satisfied if either of the intervals is absent.
noOverlapSequence$(B, dist)$	Sequences the intervals in set B. Ensures that the intervals in B do not overlap. Furthermore, the two-dimensional distance matrix *dist* specifies a sequence dependent setup time for each pair of activities. Absent intervals are ignored.
first(α, B)	If interval α is present, it must be scheduled before any of the intervals in B.
last(α, B)	If interval α is present, it must scheduled after any of the intervals in B.

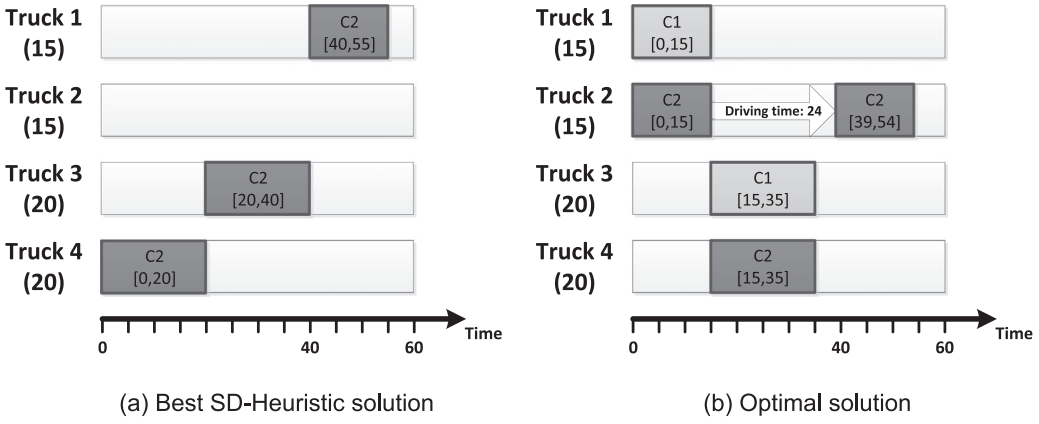

Fig. 1. Example where the SD-heuristic does not find the optimal solution.

4.1.2. Steepest descent

The quality of the schedule produced by the best-fit heuristic is largely determined by the order π in which the customers are being processed. Different solutions may be discovered when the best-fit procedure is invoked multiple times for different permutations of π. To this extent, a local search procedure is used which modifies the vector π. The initial ordering of π is determined according to the following criteria:

1. earliest deadline b_i;
2. highest demand q_i;
3. earliest release date a_i.

These criteria and their order have been determined empirically. At each iteration, the steepest descent heuristic performs a full neighborhood search, i.e. all shifts and swaps of customers in π are considered.

4.1.3. Heuristic limitations

A number of instances exist where the heuristic would never find the optimal solution, independent of the order π. An example of such an instance is depicted in Fig. 1. Two customers (1 and 2) with demands $q_1 = 25$ and $q_2 = 45$ are scheduled using four vehicles (two with capacity 15 and two with capacity 20) in time windows $[a_1 = 0, b_1 = 40]$ and $[a_2 = 0, b_2 = 60]$. The maximum time lag $\gamma = 5$. The delivery times are equal to the vehicle capacities, i.e. $q_k = p_k$, for all $k \in K$. The travel times between customers are $t_{11} = 18, t_{12} = 37, t_{21} = 37, t_{22} = 24$; all other travel times are set to 0.

The heuristic fails to find the optimal solution due to the criteria used to determine the next delivery to a specific customer. As elaborated in the subsection 4.1.1, the vehicle performing the next delivery for a customer is selected according to its availability, the amount of waste it produces and its size. Only when the first criterion is replaced by 'select the vehicle which produces the largest amount of waste', the optimal solution for this instance is found. It may be clear however that such a criterion is undesirable for most instances.

4.2. Fix-and-optimize heuristic

The pseudo code for the fix-and-optimize heuristic is provided in Algorithm 3. The following functions are used in this code:

- *satisfied*$(\mathcal{S}, i)$: returns whether customer $i \in C$ is satisfied in solution $\mathcal{S}$, i.e. sufficient concrete is delivered to this customer.
- *calcOverlappingSet*(i): Calculates the set of customers $C' \subset C$ having an overlap between their delivery interval and that of customer $i \in C$. Note that $|C'| \geq 1$ as customer i must be included in C'.
- *fix*$(C', \mathcal{S})$: Fixes the variables for the customers in C' as stated in schedule $\mathcal{S}$. If customer $j \in C'$ is not satisfied in schedule $\mathcal{S}$, set $y_j = 0$, set $y_j = 1$ otherwise. In a similar fashion, the completion variables as well as a number of flow variables are fixed.
- *reOptimize*$(\mathcal{S})$: Solves the resulting MIP model and updates $\mathcal{S}$ accordingly.
- *release*(): Unfixes all fixed variables.

The algorithm is initialized with a feasible schedule $\mathcal{S}$ and a permutation π of customers not satisfied in $\mathcal{S}$. At each iteration

of the algorithm, the fix-and-optimize heuristic attempts to optimize part of the schedule, using the MIP model CDP2 presented in Section 3.1.2. However, instead of solving the entire model at once, only a small part of the problem is optimized, as most of the variables are fixed. First, for a given customer $i \in \pi$, the heuristic determines the set of customers $C' \subseteq C$ having their delivery interval overlap with the interval $[a_i, b_i]$. Next, the heuristic fixes all y_i and C^i variables of the customers not in C' to their corresponding values in $\mathcal{S}$. Moreover, flow variables x_{ijk} that do not affect any of the possible deliveries to customers in C' are fixed as well. Finally, the heuristic solves the resulting MIP model, thereby rescheduling the customers in C', while leaving the deliveries to the other customers unchanged. The resulting schedule must be at least as good as the schedule obtained at the previous iteration of the algorithm.

Note that the above procedure can be used to:

1. improve an existing schedule, i.e. attempt to add unscheduled customers
2. construct a new schedule by starting from an empty schedule

Algorithm 3. A constructive heuristic.

input: A permutation of customers to schedule π, An initial schedule $\mathcal{S}$
output: A feasible CDP solution

```
foreach i ∈ π do
    if ¬satisfied(S, i) then
        O = calcOverlappingSet(i);
        C' = {j ∈ C : j ∉ O};
        fix(C', S);
        reOptimize(S);
        release();
return S
```

5. Bounds

Upper bounds, required to assess the quality of the proposed algorithms, are obtained by solving the LP relaxation of CDP2 (Section 3.1.2). A number of cuts are added to strengthen the bound. These cuts are calculated by solving a number of small subproblems. First, a simple test is conducted to verify whether there exists at least one schedule satisfying a specific customer. The latter is achieved by solving the CDP2 (Section 3.1.2) as follows:

1. First, for a given customer $i \in C$, set $y_i = 1$, and set $y_j = 0$, for all $j \in C, j \neq i$.
2. Solve CDP2. Whenever the model turns out infeasible, there exists no schedule which accommodates customer i.

All customers who could not be accommodated in the schedule are permanently discarded. Next, the above procedure is repeated in a similar fashion, but this time for pairs of customers:

1. For every pair of customers $i, j \in C, i \neq j$, set $y_i = y_j = 1$, and set $y_k = 0$, for all $k \in C, k \notin \{i, j\}$.
2. Solve the MIP model. Whenever the model turns out infeasible, there exists no schedule in which both customers i and j are satisfied. Consequently, the valid inequality $y_i + y_j \leq 1$ may be added to the model.

Note that the above procedure identifies Minimum Infeasible Subsets (MIS) of size two. Clearly, the procedure may be repeated to identify larger Infeasible Subsets, this, however, quickly becomes computationally intractable.

After adding all cuts based on MIS of size two, the LP relaxation of the resulting model is computed, strengthened by a number of cuts automatically generated by IBM's Ilog Cplex solver version 12.5.1.

An alternative means to calculate a valid upper bound involves solving the following simple MIP model:

$$\max. \quad \sum_{i \in C} q_i y_i \tag{35}$$

$$\sum_{i \in S} y_i \leq |S| - 1 \qquad \forall S \in \mathbb{S}, S \subseteq C \tag{36}$$

$$y_i \in \{0, 1\} \qquad \forall i \in C \tag{37}$$

Here, $\mathbb{S}$ is the set of all infeasible subsets. As noted before, computing the entire set $\mathbb{S}$ is intractable. We therefor limit computations to $|S| \leq 3$.

6. Experimental results

6.1. Data sets

Due to the lack of publicly available benchmark data for the CDP, two data sets are created. Both sets, as well as code to generate new instances, are available online at [17]. A summary of the instances is provided in Table 4. Data Set A contains instances with 10–20 *customers*, and 2–5 *vehicles*, whereas Data Set B contains instances with up to 50 *customers* and 20 *vehicles*. The *demands* of customers are always divisible by 5 and are selected at random from the interval [10–75]. In a similar fashion, the *capacities* of the vehicles are selected between [10–25]. It is unrealistic to have a unique capacity for each vehicle, and therefore the number of different vehicles is bounded by the number of *vehicle classes*. The *processing time* of a vehicle $k \in K$ is set proportional to its capacity, i.e. $p_k = q_k$ for all $k \in K$. The width of the time window of a customer $i \in C$ is computed by $\lambda_i q_i$, where λ_i is a scalar uniformly selected from the interval $[1.1, 2.1]$.

Each instance has up to 4 *production stations*. The locations of the start and end depot, construction sites, and production depots, lay in the euclidean plane. The travel time between two locations equals the euclidean distance, rounded upwards. For simplicity, the start and end depot are the same. The distance between each station and the depot is uniformly selected from the interval [1–30]. Furthermore, for each customer, there exists a production station within a distance of [1–25]. Finally, the time lag γ is fixed to 5 for all deliveries.

Table 4
Data sets.

	Set A	Set B
Instances	64	128
Customers	10–20	20–50
Demands	10–75	10–75
Time windows	$q_i \times [1.1, 2.1]$	$q_i \times [1.1, 2.1]$
Time lags	5	5
Vehicles	2–5	6–20
Capacity	10–25	10–25
Vehicle classes	2–3	3
Processing time	$p_k = q_k$	$p_k = q_k$
Stations	1–4	1–4
Cust.-station	1–30	1–30
Depot-station	1–25	1–25

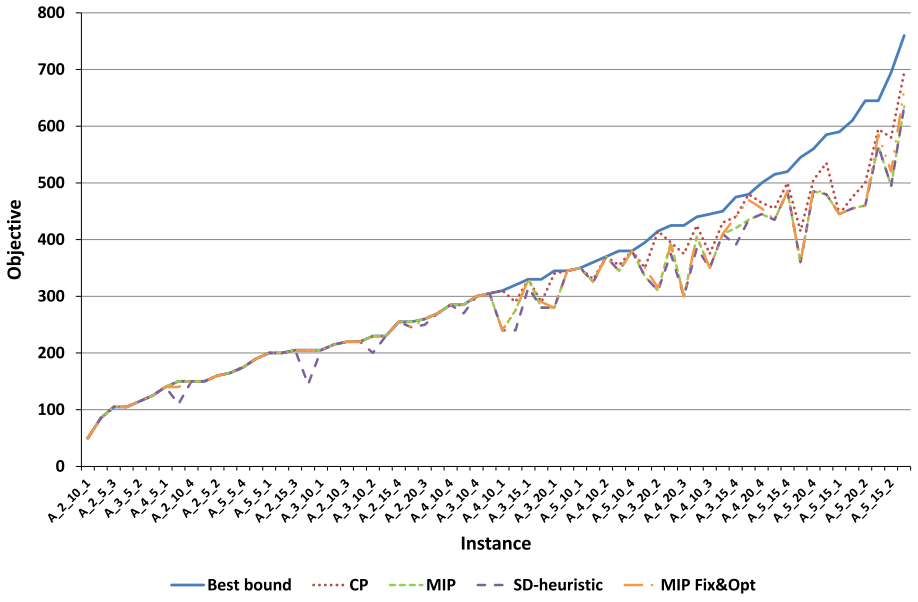

Fig. 2. Comparison of the different methods on Data Set A.

Table 5
Summary.

Method	Data Set A		Data Set B	
	AVG gap (%)	Time (ms)	AVG gap (%)	Time (ms)
CP	4.2	197,012	12.1	357313
MIP	7.3	13,405,798	–	–
SD-heuristic	9.1	23	16.3	1331
MIP Fix & Opt	7.0	100,765	–	–

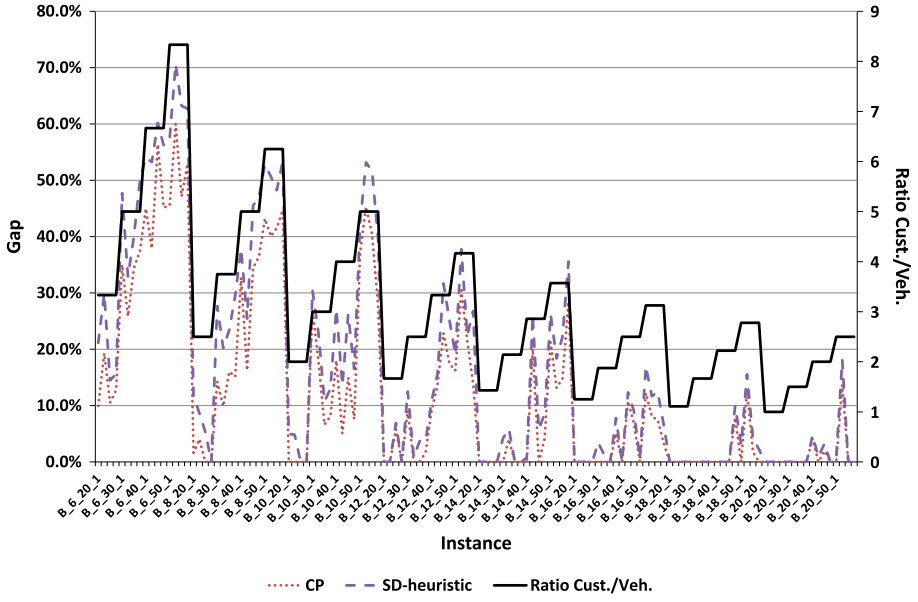

Fig. 3. Comparison of the different methods on Data Set B.

6.2. Experiments

A number of experiments are conducted to assess the quality of the algorithms presented in this work. The results of the Steepest Descent heuristic (*SD-Heuristic*, Section 4.1), the fix-and-optimize MIP heuristic (*MIP fix & opt*, Section 4.2), the exact MIP algorithm (*MIP*, Section 3.1.2) and the CP algorithm (*CP*, Section 3.2) on Data Set A are provided in Table B1 in Appendix B. The first column of

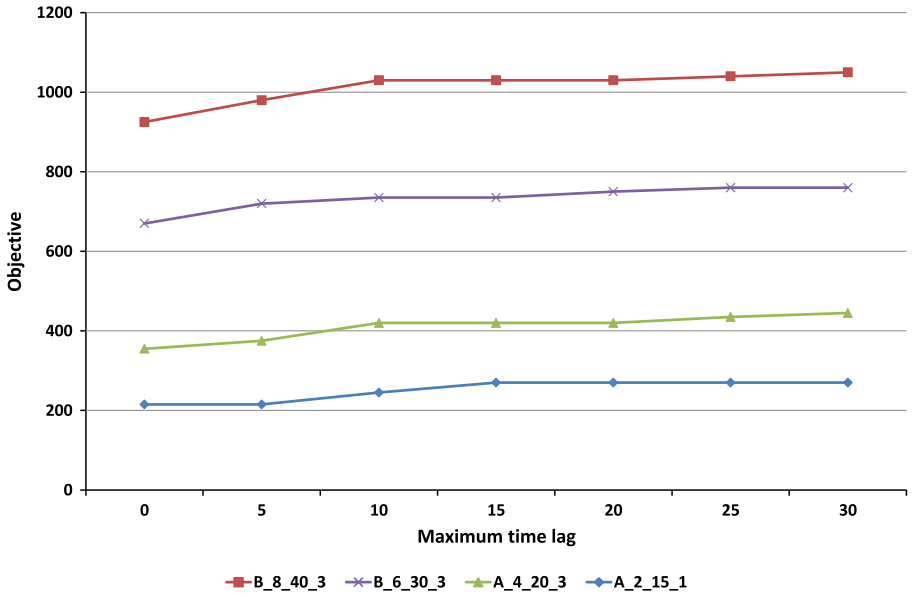

Fig. 4. Influence of the time lag γ. Solutions are obtained via the CP model.

Table B1 provides the instance names, following a "*W_x_y_z*" naming scheme, where *W* reflects the data set, *x* the number of vehicles, *y* the number of customers and *z* the number of stations. The second column provides an upper bound on the optimal solution values; only the best bound obtained from the two procedures discussed in Section 5 is shown. Per solution method, the table shows resp. the objective value, the gap between the objective value and the bound, and the computation time in milliseconds. Each solution method was allotted 5 min computation time per instance. Furthermore, the exact MIP, CP methods, as well as the MIP fix-and-optimize heuristic are initialized by the results obtained from the SD-heuristic; hence, their solutions are at least as good as the solution of the SD-heuristic. Finally, Table 5 displays the average gap obtained by all methods, as well as the average computation time.

The differences in results for the various methods are relatively small when compared on Data Set A (Fig. 2). Clearly, from an objective point of view, CP outperforms all methods (avg. gap 4.2%), followed by the fix-and-optimize heuristic (avg. gap 7.0%) the exact MIP procedure (avg. gap 7.3%) and the SD-heuristic (avg. gap 9.1%). Although the SD-heuristic performs the least in terms of objective value, it usually requires less than 50 ms per instance, whereas the other methods require significantly more time. CP solved 40 out of 64 instances to optimality, the exact MIP method 37, the fix-and-optimize MIP heuristic 35 and the SD-heuristic 30.

The results for Data Set B are shown in Table B2 in Appendix B. The results for the two approaches based on the MIP formulation were omitted from the graphs as these methods were unable to deal with the larger instance sizes. The time limit for the remaining methods is set to 10 minutes per instance. The average gaps obtained are resp. 12.1% for CP, and 16.3% for the SD-heuristic. Out of 128 instances, CP solved 55 instances to optimality, versus 40 instances for the SD-heuristic. Compared to Data Set A, the average gaps have increased significantly, especially for instances with a high ratio between the number of customers and the number of vehicles (Fig. 3). Computing strong bounds for these instances via the approaches elaborated in Section 5 is difficult and very time consuming as enumerating all infeasible subsets of sizes 2 and 3 is computationally expensive for large numbers of customers.

Often, CP finds good solutions in relatively little time, but fails to prove optimality. This is mainly due to the fact that most interval variables in the CP model are declared optional, diminishing the potential amount of constraint propagation. Typically, the smaller the initial domains of the variables (e.g. small time windows for the customers), the faster CP is capable of solving an instance.

Fig. 4 plots the influence of the time lag on several instances. As is visible from the monotonically increasing lines, increasing the time lag improves the overall flexibility of the schedule, thereby enabling better quality solutions.

7. Conclusion

Although several variations of CDP exist in the literature, little attempts have been made to compare the various approaches mutually. In practice, a fair comparison is hindered by differences in problem definitions, as well as by the lack of publicly available benchmark data. This work introduces a more general version of CDP, bearing a strong resemblance with two well-studied scheduling and routing problems: the Parallel Machine Scheduling Problem and the Capacitated Vehicle Routing Problem. The problem is shown to be NP-Hard by reduction to the Hamiltonian Cycle problem. New exact, heuristic and hybrid solution approaches are proposed. Computational experiments are conducted on a library of CDP instances, and compared against bounds on the optimal solutions. The MIP model appears ineffective in solving large problem instances, but can be used to compute bounds instead. The CP model, on the other hand, is highly effective in finding high quality solutions in relatively little time, or to improve existing schedules. When computation time is a limiting factor, the Steepest-Descent heuristic is a viable choice, as it often yields good solutions in less than a second. Finally, when compared to the traditional MIP approach, the fix-and-optimize MIP heuristic produces, on average, better results in less time.

The current work primarily focuses on a static scheduling problem where the set of customers and their demands are known beforehand. Future work could therefore be aimed at an online version of the problem with a rolling scheduling horizon. A similar approach to the fix-and-optimize MIP heuristic may be used to fix part of the schedule which is currently being executed, while reoptimizing the delivery schedule towards the end of the time horizon. On a similar note, different objective functions may be

Table B1
Computational results Data Set A.

		CP			MIP			SD-heuristic			MIP Fix-Opt		
Instance	UB	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
A_2_5_1	85	85	0	414	85	0	1055	85	0	9	85	0	375
A_2_5_2	160	160	0	53	160	0	139	160	0	5	160	0	44
A_2_5_3	105	105	0	37	105	0	39	105	0	3	105	0	33
A_2_5_4	105	105	0	300,000	105	0	107	105	0	10	105	0	531
A_2_10_1	50	50	0	212	50	0	125	50	0	24	50	0	268
A_2_10_2	150	150	0	300,000	150	0	4587	110	27	21	140	7	15,227
A_2_10_3	220	220	0	255	220	0	378	220	0	52	220	0	261
A_2_10_4	150	150	0	300,000	150	0	1443	150	0	50	150	0	1847
A_2_15_1	215	215	0	431	215	0	618	215	0	89	215	0	1326
A_2_15_2	320	290	9	300,000	275	14	300,000	240	25	15	275	14	90,674
A_2_15_3	205	205	0	300,000	205	0	2167	205	0	12	205	0	139,759
A_2_15_4	255	255	0	300,000	255	0	4902	255	0	11	255	0	49,266
A_2_20_1	255	255	0	300,000	255	0	10,256	245	4	52	245	4	294,900
A_2_20_2	270	270	0	300,000	270	0	7624	270	0	42	270	0	201,615
A_2_20_3	260	260	0	300,000	260	0	10,135	250	4	27	260	0	155,839
A_2_20_4	380	355	7	300,000	345	9	300,000	345	9	39	345	9	162,334
A_3_5_1	205	205	0	175	205	0	776	145	29	1	205	0	102
A_3_5_2	115	115	0	187	115	0	212	115	0	0	115	0	24
A_3_5_3	125	125	0	16	125	0	98	125	0	0	125	0	5
A_3_5_4	190	190	0	18	190	0	107	190	0	1	190	0	8
A_3_10_1	205	205	0	353	205	0	274	205	0	3	205	0	2230
A_3_10_2	230	230	0	2637	230	0	4039	200	13	2	230	0	23,278
A_3_10_3	305	305	0	300,000	305	0	6025	305	0	3	305	0	53,302
A_3_10_4	300	300	0	300,000	300	0	543	300	0	5	300	0	394
A_3_15_1	330	330	0	300,000	330	0	11,191	315	5	44	330	0	102,587
A_3_15_2	425	395	7	300,000	395	7	300,000	375	12	14	390	8	4239
A_3_15_3	330	290	12	300,000	280	15	300,000	280	15	9	290	12	52,873
A_3_15_4	475	440	7	300,000	420	12	300,000	390	18	16	440	7	254,550
A_3_20_1	345	340	1	300,000	280	19	300,000	280	19	32	280	19	253,366
A_3_20_2	415	415	0	300,000	310	25	300,000	310	25	23	315	24	255,852
A_3_20_3	360	330	8	300,000	325	10	300,000	325	10	56	325	10	243,884
A_3_20_4	480	480	0	300,000	435	9	300,000	435	9	35	470	2	149,430
A_4_5_1	140	140	0	15	140	0	79	140	0	0	140	0	5
A_4_5_2	150	150	0	22	150	0	296	150	0	0	150	0	16
A_4_5_3	165	165	0	16	165	0	95	165	0	0	165	0	7
A_4_5_4	230	230	0	21	230	0	265	230	0	0	230	0	10
A_4_10_1	310	310	0	300,000	240	23	300,000	240	23	5	240	23	300,000
A_4_10_2	370	370	0	300,000	370	0	51,041	370	0	1	370	0	312
A_4_10_3	445	375	16	300,000	350	21	300,000	350	21	3	350	21	300,000
A_4_10_4	285	285	0	300,000	285	0	657	285	0	2	285	0	40
A_4_15_1	545	415	24	300,000	360	34	300,000	360	34	17	360	34	296,608
A_4_15_2	610	475	22	300,000	455	25	300,000	455	25	31	455	25	192,264
A_4_15_3	450	430	4	300,000	410	9	300,000	410	9	20	410	9	29,357
A_4_15_4	515	455	12	300,000	435	16	300,000	435	16	29	435	16	282,456
A_4_20_1	585	535	9	300,000	480	18	300,000	480	18	57	480	18	196,031
A_4_20_2	440	425	3	300,000	405	8	300,000	385	13	58	405	8	147,700
A_4_20_3	425	375	12	300,000	300	29	300,000	300	29	89	300	29	300,000
A_4_20_4	500	465	7	300,000	445	11	300,000	445	11	39	455	9	215,212
A_5_5_1	200	200	0	17	200	0	113	200	0	0	200	0	7
A_5_5_2	200	200	0	22	200	0	187	200	0	0	200	0	11
A_5_5_3	220	220	0	24	220	0	234	220	0	0	220	0	14
A_5_5_4	175	175	0	300,000	175	0	390	175	0	0	175	0	49
A_5_10_1	350	350	0	62	350	0	1593	350	0	3	350	0	71
A_5_10_2	345	345	0	44	345	0	835	345	0	1	345	0	36
A_5_10_3	285	285	0	300,000	285	0	22,468	270	5	3	285	0	158
A_5_10_4	380	380	0	56	380	0	1300	380	0	2	380	0	41
A_5_15_1	590	445	25	300,000	445	25	300,000	445	25	19	445	25	300,000
A_5_15_2	695	580	17	300,000	495	29	300,000	495	29	15	520	25	241,135
A_5_15_3	395	350	11	300,000	335	15	300,000	335	15	20	350	11	104,745
A_5_15_4	520	500	4	300,000	485	7	300,000	485	7	37	485	7	168,244
A_5_20_1	760	695	9	300,000	635	16	300,000	635	16	49	665	13	253,379
A_5_20_2	645	500	22	300,000	460	29	300,000	460	29	67	460	29	298,333
A_5_20_3	645	595	8	300,000	565	12	300,000	565	12	63	585	9	136,384
A_5_20_4	560	505	10	300,000	485	13	300,000	485	13	121	490	13	172,364

Table B2
Computational results Data Set B.

		CP			SD-heuristic		
Instance	UB	Obj	Gap (%)	Time	Obj	Gap (%)	Time
B_6_20_1	805	725	9.9	600,000	635	21.1	431
B_6_20_2	855	690	19.3	600,000	600	29.8	103

Table B2 (*continued*)

		CP			SD-heuristic		
Instance	UB	Obj	Gap (%)	Time	Obj	Gap (%)	Time
B_6_20_3	760	680	10.5	600,000	650	14.5	48
B_6_20_4	705	615	12.8	600,000	595	15.6	52
B_6_30_1	1300	845	35.0	600,000	680	47.7	281
B_6_30_2	1140	845	25.9	600,000	765	32.9	160
B_6_30_3	1060	695	34.4	600,000	635	40.1	262
B_6_30_4	1000	625	37.5	600,000	500	50.0	354
B_6_40_1	1545	850	45.0	600,000	710	54.0	1359
B_6_40_2	1635	1015	37.9	600,000	765	53.2	534
B_6_40_3	1570	685	56.4	600,000	625	60.2	2266
B_6_40_4	1450	795	45.2	600,000	635	56.2	850
B_6_50_1	1890	1030	45.5	600,000	805	57.4	1869
B_6_50_2	2250	900	60.0	600,000	660	70.7	1687
B_6_50_3	1740	920	47.1	600,000	640	63.2	1767
B_6_50_4	2080	980	52.9	600,000	775	62.7	2551
B_8_20_1	935	920	1.6	600,000	825	11.8	44
B_8_20_2	865	830	4.0	600,000	790	8.7	55
B_8_20_3	655	655	0.0	77,476	620	5.3	36
B_8_20_4	820	820	0.0	312	820	0.0	80
B_8_30_1	1085	930	14.3	600,000	785	27.6	250
B_8_30_2	1115	1005	9.9	600,000	890	20.2	280
B_8_30_3	1155	970	16.0	600,000	885	23.4	555
B_8_30_4	1320	1120	15.2	600,000	930	29.5	410
B_8_40_1	1665	1120	32.7	600,000	1035	37.8	1351
B_8_40_2	1415	1185	16.3	600,000	1055	25.4	773
B_8_40_3	1495	980	34.4	600,000	815	45.5	1087
B_8_40_4	1730	1100	36.4	600,000	915	47.1	1024
B_8_50_1	1980	1125	43.2	600,000	935	52.8	5810
B_8_50_2	1935	1160	40.1	600,000	955	50.6	4615
B_8_50_3	1960	1145	41.6	600,000	1015	48.2	3623
B_8_50_4	1835	1015	44.7	600,000	855	53.4	4024
B_10_20_1	805	805	0.0	24,814	765	5.0	36
B_10_20_2	825	825	0.0	415,312	785	4.8	28
B_10_20_3	730	730	0.0	361	730	0.0	30
B_10_20_4	765	765	0.0	383	765	0.0	29
B_10_30_1	1215	885	27.2	600,000	845	30.5	459
B_10_30_2	1355	1115	17.7	600,000	1045	22.9	447
B_10_30_3	1210	1130	6.6	600,000	1080	10.7	327
B_10_30_4	1235	1135	8.1	600,000	1075	13.0	383
B_10_40_1	1475	1210	18.0	600,000	1075	27.1	1086
B_10_40_2	1580	1500	5.1	600,000	1365	13.6	1295
B_10_40_3	1605	1365	15.0	600,000	1185	26.2	1235
B_10_40_4	1455	1345	7.6	600,000	1215	16.5	1353
B_10_50_1	2265	1420	37.3	600,000	1300	42.6	3394
B_10_50_2	1900	1040	45.3	600,000	890	53.2	5223
B_10_50_3	2005	1195	40.4	600,000	970	51.6	3597
B_10_50_4	1925	1390	27.8	600,000	1160	39.7	4719
B_12_20_1	770	770	0.0	260	770	0.0	15
B_12_20_2	770	770	0.0	502	770	0.0	33
B_12_20_3	945	890	5.8	600,000	880	6.9	64
B_12_20_4	850	850	0.0	576	850	0.0	71
B_12_30_1	1320	1200	9.1	600,000	1155	12.5	324
B_12_30_2	1185	1185	0.0	913	1175	0.8	168
B_12_30_3	950	950	0.0	46,560	915	3.7	209
B_12_30_4	1185	1165	1.7	600,000	1140	3.8	327
B_12_40_1	1475	1340	9.2	600,000	1315	10.8	481
B_12_40_2	1510	1305	13.6	600,000	1275	15.6	903
B_12_40_3	1640	1265	22.9	600,000	1120	31.7	1713
B_12_40_4	1550	1280	17.4	600,000	1160	25.2	2258
B_12_50_1	1755	1470	16.2	600,000	1420	19.1	3408
B_12_50_2	2000	1390	30.5	600,000	1245	37.8	7886
B_12_50_3	1825	1435	21.4	600,000	1415	22.5	6029
B_12_50_4	1940	1670	13.9	600,000	1420	26.8	4249
B_14_20_1	830	830	0.0	666	830	0.0	15
B_14_20_2	695	695	0.0	536	695	0.0	15
B_14_20_3	840	840	0.0	633	840	0.0	40
B_14_20_4	755	755	0.0	570	755	0.0	17
B_14_30_1	1190	1190	0.0	3690	1140	4.2	275
B_14_30_2	1370	1315	4.0	600,000	1290	5.8	261
B_14_30_3	1005	1005	0.0	551	1005	0.0	109
B_14_30_4	1205	1205	0.0	1579	1205	0.0	207
B_14_40_1	1395	1395	0.0	4612	1385	0.7	355
B_14_40_2	1725	1380	20.0	600,000	1275	26.1	1929
B_14_40_3	1550	1550	0.0	64,809	1460	5.8	433
B_14_40_4	1705	1635	4.1	600,000	1560	8.5	1289

Table B2 (*continued*)

		CP			SD-heuristic		
Instance	UB	Obj	Gap (%)	Time	Obj	Gap (%)	Time
B_14_50_1	2285	1800	21.2	600,000	1685	26.3	3747
B_14_50_2	2015	1750	13.2	600,000	1645	18.4	8700
B_14_50_3	2095	1785	14.8	600,000	1630	22.2	5462
B_14_50_4	2095	1500	28.4	600,000	1350	35.6	9086
B_16_20_1	905	905	0.0	882	905	0.0	17
B_16_20_2	805	805	0.0	748	805	0.0	24
B_16_20_3	915	915	0.0	908	915	0.0	21
B_16_20_4	875	875	0.0	812	875	0.0	42
B_16_30_1	1305	1305	0.0	120,612	1260	3.4	317
B_16_30_2	1175	1175	0.0	6503	1160	1.3	534
B_16_30_3	1105	1105	0.0	1363	1105	0.0	114
B_16_30_4	1090	1035	5.0	600,000	1005	7.8	569
B_16_40_1	1340	1340	0.0	1132	1340	0.0	138
B_16_40_2	1580	1410	10.8	600,000	1385	12.3	1419
B_16_40_3	1600	1490	6.9	600,000	1455	9.1	2024
B_16_40_4	1615	1615	0.0	3596	1605	0.6	1518
B_16_50_1	2090	1835	12.2	600,000	1735	17.0	2086
B_16_50_2	1930	1775	8.0	600,000	1705	11.7	1169
B_16_50_3	2010	1860	7.5	600,000	1760	12.4	2163
B_16_50_4	1980	1915	3.3	600,000	1850	6.6	2831
B_18_20_1	820	820	0.0	935	820	0.0	24
B_18_20_2	740	740	0.0	799	740	0.0	16
B_18_20_3	775	775	0.0	704	775	0.0	15
B_18_20_4	840	840	0.0	900	840	0.0	17
B_18_30_1	1080	1080	0.0	1727	1080	0.0	159
B_18_30_2	1205	1205	0.0	2009	1205	0.0	66
B_18_30_3	1155	1155	0.0	1683	1155	0.0	77
B_18_30_4	1125	1125	0.0	1784	1125	0.0	60
B_18_40_1	1670	1670	0.0	4303	1670	0.0	633
B_18_40_2	1635	1635	0.0	2409	1635	0.0	363
B_18_40_3	1610	1610	0.0	3895	1610	0.0	539
B_18_40_4	1655	1525	7.9	600,000	1490	10.0	1493
B_18_50_1	1795	1795	0.0	586,397	1740	3.1	3333
B_18_50_2	1930	1685	12.7	600,000	1630	15.5	9162
B_18_50_3	2005	1970	1.7	600,000	1920	4.2	2101
B_18_50_4	1795	1795	0.0	10,385	1755	2.2	1498
B_20_20_1	875	875	0.0	593	875	0.0	17
B_20_20_2	770	770	0.0	452	770	0.0	19
B_20_20_3	980	980	0.0	1296	980	0.0	19
B_20_20_4	765	765	0.0	916	765	0.0	16
B_20_30_1	1250	1250	0.0	2247	1250	0.0	71
B_20_30_2	1325	1325	0.0	2627	1325	0.0	178
B_20_30_3	1205	1205	0.0	2337	1205	0.0	91
B_20_30_4	1245	1245	0.0	2330	1245	0.0	84
B_20_40_1	1695	1625	4.1	600,000	1615	4.7	1535
B_20_40_2	1725	1725	0.0	25,317	1710	0.9	765
B_20_40_3	1540	1510	1.9	600,000	1490	3.2	499
B_20_40_4	1530	1530	0.0	4043	1530	0.0	182
B_20_50_1	2075	2075	0.0	309,753	2055	1.0	1896
B_20_50_2	1825	1575	13.7	600,000	1495	18.1	5757
B_20_50_3	1825	1825	0.0	7912	1825	0.0	1251
B_20_50_4	1890	1890	0.0	2895	1890	0.0	1159

investigated. For example, deliveries for customers which were not serviced by the concrete distributor due to a lack of available vehicle capacity may be postponed to a later point in time, while being treated with higher priority.

Appendix A. Literature summary notation

Solution method:

- Mixed Integer Programming (*m*) — The paper proposes a MIP model.
- Constraint Programming (*c*) — The paper proposes a CP model.
- Heuristic (*he*) — The paper proposes a heuristic approach.
- Hybrid (*hy*) — The paper proposes a hybrid approach, involving both exact approaches and local search techniques.

Time windows and limits:

- Soft delivery time windows (*sd*) — Time windows may be violated, or only a preferred start time of the deliveries is provided.
- Hard delivery time windows (*hd*) — Deliveries must be made within a defined time window.
- Hard delivery start time (*hs*) — Deliveries may not commence before a predefined time.

- Vehicle usage time (*sv*) — Vehicles may only be used for a certain amount of time, or may not be available during certain periods, e.g. due to maintenance.
- Concrete Perish Time (*p*) — The amount of time concrete may reside in the truck is limited. Consequently, customers may not be reachable from distant depots, or deliveries are aborted before the truck is empty as the truck's processing time exceeds the time limit.

Start, end location of vehicles:
- Central depot (*c*) — All vehicles start and/or return to a central depot.
- Production center (*p*) — All vehicles start and/or return to a (specific) production center.
- Mixed (*m*) — A vehicle may start and/or end at a production center or at a depot.

Production depots:
- Homogeneous (*ho*) — All production depots are identical.
- Heterogeneous (*hs*) — Differences in production depots, e.g. they produce different types of concrete, or cannot service all customers.
- Scheduling (*s*) — Vehicle reloads must be scheduled, e.g. a vehicle may have to wait while another vehicle is being loaded.

Loading and unloading:
- Fixed rate (*f*) — (un)loading takes a constant amount of time.
- Dynamic rate (*d*) — (un)loading time depends on the specific customer, vehicle, type of concrete, amount of concrete in the truck and/or the production station.

Fleet:
- Homogeneous (*ho*) — All vehicles are identical.
- Heterogeneous (*he*) — Vehicles differ in capacity or equipment carried.

Instrumentation:
- For some deliveries, specialized equipment must be present. Either the delivery truck has this equipment, or an additional vehicle having this equipment must be scheduled.

Deliveries, restrictions:
- Synchronization (*s*) — Deliveries for the same customer must be synchronized as they may not overlap in time, and/or have to take a minimum or a maximum time lag into consideration.
- Revisits (*r*) — A single truck may perform multiple deliveries for the same customer, not necessary in a consecutive order.
- Vehicle requirements (ν) — Not all vehicles can perform all deliveries, e.g. because a vehicle cannot transport the type of concrete required by the customer, or a vehicle may be too large for the construction site.
- Reload (*l*) — Vehicles must reload after each delivery.
- Shared delivery (*sh*) — A vehicle may split its content over multiple customers without reloading in between deliveries.
- Split delivery (*sp*) — Customer may require multiple deliveries by different vehicles.
- Single source (*ss*) — For some customers, all concrete delivered must originate from the same production site.

Objectives (incl. weighted versions and composite objectives):
- Minimize vehicle usage (*u*) — The frequency a vehicle is used, or the time that a vehicle is used, incl. travel time, loading and unloading.
- Minimize wastage (*w*) — Amount of concrete delivered to a customer that surpasses the requested amount.
- Minimize delay (*d*) — Deviation from soft time window, deviation from requested start time, or vehicle overtime.
- Minimize outsourcing (*o*) — When the schedule fails accommodate deliveries for a particular customer, the deliveries may be outsourced.
- Minimize operating costs (*op*) — Operating costs incurred at the production site or the construction sites, e.g. the (weighted) time difference between the start and end time of the production of resp. the first and the last batch of concrete.
- Maximize utilization balance (*b*) — Ensure that vehicles are employed equally often.
- Minimize travel time or distance (*t*).
- Minimize number of vehicles (v_s) used in the solution.
- Minimize number of vehicles (v_c) used per customer.
- Maximize number of satisfied customers (*s*).

Appendix B. Computational results

See Tables B1 and B2.

References

[1] Hertz A, Uldry M, Widmer M. Integer linear programming models for a cement delivery problem. Eur J Oper Res 2012;222(3):623–31.

[2] Misir M, Vancroonenburg W, Verbeeck K, Vanden Berghe G. A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete. In: Di Gaspero L, Schaerf A, Stützle T, editors, Proceedings of the 9th metaheuristics international conference, July 2011. p. 289–98.

[3] Silva C, Faria JM, Abrantes P, Sousa JMC, Surico M, Naso D. Concrete delivery using a combination of GA and ACO. In: 44th IEEE conference on decision and control, 2005 and 2005 European control conference. CDC-ECC '05, 2005, pp. 7633–8.

[4] Asbach L, Dorndorf U, Pesch E, Analysis. modeling and solution of the concrete delivery problem. Eur J Oper Res 2009;193(3):820–35.

[5] Yan S, Lai W. An optimal scheduling model for ready mixed concrete supply with overtime considerations. Autom Constr 2007;16(6):734–44.

[6] Lin P-C, Wang J, Huang S-H, Wang Y-T. Dispatching ready mixed concrete trucks under demand postponement and weight limit regulation. Autom Constr 2010;19(6):798–807.

[7] Naso D, Surico M, Turchiano B, Kaymak U. Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete. Eur J Oper Res 2007;177(3):2069–99.

[8] Schmid V, Doerner KF, Hartl RF, Savelsbergh MWP, Stoecher W. A hybrid solution approach for ready-mixed concrete delivery. Transp Sci 2009;43(1):70–85.

[9] Schmid V, Doerner KF, Hartl RF, Salazar-González J-J. Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. Comput Oper Res 2010;37(3):559–74.

[10] Desaulniers G, Desrosiers J, Solomon M, editors. A primer in column generation. US: Springer; 2005.

[11] Graham LD, Forbes DR, Smith SD. Modeling the ready mixed concrete delivery system with neural networks. Autom Constr 2006;15(5):656–63.

[12] Bard JF, Rojanasoonthon S. A branch-and-price algorithm for parallel machine scheduling with time windows and job priorities. Nav Res Log (NRL) 2006;53(1):24–44.

[13] Berghman L, Leus R, Spieksma F. Optimal solutions for a dock assignment problem with trailer transportation. Ann Oper Res 2011:1–23.

[14] Garey MR, Johnson DS. Computers and intractability: A guide to the theory of NP-completeness. New York, USA: W. H. Freeman; 1979.

[15] Laborie P, Rogerie J. Reasoning with conditional time-intervals. In: FLAIRS conference, 2008, pp. 555–60.

[16] Laborie P, Rogerie J, Shaw P, Vilím P. Reasoning with conditional time-intervals. Part ii: An algebraical model for resources. In: FLAIRS conference, 2009.

[17] Kinable J, Wauters T. CDPLib. ⟨https://sites.google.com/site/cdplib/⟩, 2013.