# A Logic Based Benders' Approach to the Concrete Delivery Problem

Joris Kinable[1,2,3] and Michael Trick[1]

[1] Tepper School of Business, Carnegie Mellon University,
5000 Forbes Ave Pittsburgh, PA 15213, USA
jkinable@andrew.cmu.edu, trick@cmu.edu
[2] ORSTAT - KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium
[3] KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC,
Gebr. De Smetstraat 1, 9000 Gent, Belgium

**Abstract.** This work presents an exact Logic Based Benders' decomposition for the Concrete Delivery Problem (CDP). The CDP is a complex, real world optimization problem involving the allocation and distribution of concrete to construction sites. The key scheduling issue for the CDP is the need for successive deliveries to a site to be sufficiently close in time. We decompose the CDP into a master problem and a subproblem. Based on a number of problem characteristics such as the availability of vehicles, geographical orientation of the customers and production centers, as well as the customers' demand for concrete, the master problem allocates concrete to customers. Next, the subproblem attempts to construct a feasible schedule, meeting all the routing and scheduling constraints. Infeasibilities in the schedule are communicated back to the master problem via a number of combinatorial inequalities (Benders' cuts). The master problem is solved through a Mixed Integer Programming approach, whereas the subproblem is solved via a Constraint Programming model and a dedicated scheduling heuristic. Experiments are conducted on a large number of problem instances, and compared against other exact methods presented in related literature. This algorithm is capable of solving a number of previously unsolved benchmark instances to optimality and can improve the bounds for many other instances.

**Keywords:** Vehicle Routing, Scheduling, Logic Based Benders' Decomposition, Integer and Constraint Programming.

## 1 Introduction

Many of today's real world optimization challenges do not involve just a single problem, but often comprise a multitude of interconnected problems. Although many of these optimization problems can be formulated in a single Mixed Integer (MIP) or Constraint Programming (CP) model, solving them to optimality is only possible for moderately small problem instances. The dependencies between the subproblems produce an excessive number of conditional constraints (big-M constraints), which have a significant impact on the quality of the model.

H. Simonis (Ed.): CPAIOR 2014, LNCS 8451, pp. 176–192, 2014.

Furthermore, MIP and CP solvers often make poor branching decisions, simply because the solvers are unaware of the underlying problem structure. Therefore, a major challenge lies in the design of efficient decomposition procedures for these problems.

In this work, a logic based Benders' decomposition for the Concrete Delivery Problem (CDP) is presented. The CDP, recently presented by Kinable et al. (2013), comprises the allocation and distribution of concrete to customers, under a number of routing and scheduling constraints, while maximizing the amount of concrete delivered. Concrete is transported from production centers to the customer's construction sites by a set of heterogeneous vehicles. Often, multiple deliveries for the same customer are required as the customer's demand exceeds the capacity of a single truck. Consequently, delivery schedules for different trucks need to be synchronized as deliveries for the same customer may not overlap in time. Furthermore, successive deliveries must not differ in time too much since the concrete from an early delivery must still be liquid when a second arrives.

The logic based Benders' procedure presented in this paper decomposes the CDP into a master problem and a subproblem. The master problem (MP) allocates concrete to customers, while taking a number of resource restrictions into consideration. The subproblem (SP) attempts to find a feasible delivery schedule for the concrete trucks. Whenever such a schedule does not exist, a feasibility cut is generated and added to the master problem. The master problem and subproblem are solved iteratively, until a provable optimal (and feasible) schedule is obtained.

In Kinable et al. (2013) several solution approaches for the CDP were investigated, including two exact approaches based on Mixed Integer and Constraint Programming, as well as a number of heuristic approaches. The best performance was obtained with a hybrid approach, using a dedicated scheduling heuristic, and a CP model to improve the heuristic solutions. Although good results were reported, the approach provided little insight as to the quality of the solutions. Moreover, alternative approaches to compute bounds on the optimal solution, including a Linear Programming approach could not close the optimality gap for most instances. The approach presented in this paper addresses these issues, as bounds on the optimal solution are available through the master problem.

The CDP bears strong resemblance to a number of routing and scheduling problems, including the Pickup-and-Delivery problem with Time-Windows and Split Deliveries and the Parallel Machine Scheduling Problem with Time Lags. Although the Benders' decomposition in this work is discussed in the context of CDP, we must note that the techniques presented are not uniquely confined to this application.

The remainder of this paper is structured as follows. First in Section 2, the CDP is defined in detail. Section 3 provides a literature review. Next, Section 4 presents the Benders' decomposition, defining the master and subproblem in more detail, as well as their interaction. Experiments are conducted in Section 5, thereby comparing the Benders' decomposition against other exact methods previously appeared in literature. Finally, Section 6 offers the conclusions.

**Table 1.** Parameters defining the CDP

| Parameter | Description |
|---|---|
| $P$ | Set of concrete production sites |
| $C$ | Set of construction sites, also denoted as customers. $\|C\| = n$ |
| $V$ | $V = C \cup \{0\} \cup \{n+1\}$ |
| $0, n+1$ | resp. the start and end depots of the trucks. |
| $K$ | Set of trucks |
| $q_i$ | Requested amount of concrete by customer $i \in C$ |
| $l_k$ | 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 $j$, $i, j \in V \cup P$ |
| $\gamma$ | Maximum time lag between consecutive deliveries. |

## 2 Problem Outline

In the CDP as defined by Kinable et al. (2013), concrete has to be transported from production sites $P$ to a set of construction sites $C$. The transport is conducted by a fleet of vehicles $K$. Each vehicle $k \in K$ has a capacity $l_k$ (measured in tons of concrete) and each customer $i \in C$ has a demand $q_i$ which usually exceeds the capacity of a single truck. Concrete for a given customer $i \in C$ may only be delivered within a time window $[a_i, b_i]$, and deliveries from multiple trucks to a customer may not overlap in time. To ensure proper bonding of the concrete, the time between two consecutive deliveries for the same customer may not exceed $\gamma$. The time $p_k$, $k \in K$, required to perform a single delivery is truck dependent. Deliveries may not be preempted and trucks are always filled to their maximum capacity. Furthermore, the payload of a single truck may not be shared amongst multiple customers; whenever the capacity of a truck exceeds the customer's remaining demand, the excess amount is considered waste. A customer $i \in C$ is satisfied if at least $q_i$ tons of concrete have been delivered. The objective of the problem is to maximize the total demand of the satisfied customers.

The routing part of the CDP problem is similar to many other pickup and delivery problems. Initially, the trucks are all stationed at a starting depot. Trucks first have to drive from the starting depot to some production site to load concrete. After loading concrete, a truck travels from the production site to a customer to unload. Once the delivery is completed, the truck can either return to a production station to reload and service another customer, or the truck can return to the starting depot. Traveling between any location requires $t_{ij}$ time, $i, j \in \{0\} \cup C \cup P \cup \{n+1\}$, where 0 is the starting depot, and $n+1$ is the ending depot (0 and $n+1$ can have the same physical location). Finally, note that a single truck may also perform multiple deliveries for the same customer.

The routing problem can be formalized on a directed, weighted graph. Let the sets $P$, $C$ be defined as in Table 1. In addition, for each customer $i \in C$, an ordered set consisting of deliveries, $C^i = \{1, \ldots, m(i), \ldots, n(i)\}$, is defined,

where $m(i) = \lceil \frac{q_i}{\max_{k \in K}(l_k)} \rceil$ and $n(i) = \lceil \frac{q_i}{\min_{k \in K}(l_k)} \rceil$ are respectively lower and upper bounds on the number of deliveries required for 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 $c^i_u, c^j_v \in D, c^i_u \neq c^j_v$
- $c_{c^i_j, n+1} = t_{i,n+1}$
- $c_{0,n+1} = 0$

A solution to CDP consists of a selection of $|K|$ $s-t$ paths, which collectively satisfy the routing and scheduling constraints as outlined above. A summary of the notation used throughout this paper is provided in Table 1.

## 3 Related Research

This Section provides a brief overview on related works in the area of concrete production and distribution. An extensive overview and classification of these works has recently been published in Kinable et al. (2013).

A generalization of the CDP problem discussed in this paper is treated in Asbach et al. (2009). Next to the constraints discussed in Section 2, Asbach et al. (2009) consider vehicle synchronization at the loading depots, vehicles with specialized equipment, and additional time lags between deliveries for a single customer. In addition, Asbach et al. (2009) add constraints ensuring that some customers only receive concrete from a subset of production stations, and constraints limiting the time that concrete may reside in the vehicle's drum. Here we do not explicitly consider these constraints, as they can simply be incorporated in our model by modifying the arcs or costs in the underlying routing graph (Section 2). Asbach et al. (2009) present a MIP model to their problem, but solving it through Branch-and-Bound turned out intractable; instead, a heuristic procedure is used. The aforementioned MIP model served as the basis for the MIP approach in Kinable et al. (2013).

Schmid et al. (2009) and Schmid et al. (2010) consider a concrete delivery problem with soft time windows. In contrast to our work, their objective is to satisfy all customers, while minimizing total tardiness. Similar to Asbach et al. (2009), Schmid et al. (2009) and Schmid et al. (2010) also consider orders requiring vehicles with special equipment to be present during the entire delivery process, e.g. a pump or conveyor. Both works employ hybridized solution approaches combining MIP models and Variable Neighborhood Search heuristics.

Naso et al. (2007) propose a heuristic solution to a problem which involves the assignment of orders to concrete production centers, and the distribution of the concrete produced. Contrary to this work, deliveries are performed by a fleet of homogeneous vehicles. Furthermore, customers require an uninterrupted flow of concrete, meaning that the next truck must be available as soon as the previous truck finishes unloading. All orders must be fulfilled; unfulfilled orders are covered by external companies at a hire cost.

While in this work we consider both the routing and scheduling aspects of the concrete production and distribution, Yan and Lai (2007), Silva et al. (2005) focus primarily on the scheduling problems at the concrete production sites. In Yan and Lai (2007), only a single production center is considered, and in Silva et al. (2005) a single delivery takes a fixed amount of time which does not depend on the vehicle's location. Moreover, both works consider a homogeneous fleet of vehicles, as opposed to a heterogeneous fleet. This simplifies their scheduling problem considerably because the exact number of deliveries required to meet the customer's demand is known beforehand. This is not the case in our problem.

Similarly to this work, Hertz et al. (2012) present a decomposition approach for a concrete delivery problem. Their approach first assigns deliveries to vehicles, and then solves an independent routing problem for each vehicle. The latter is possible because the aforementioned work does not consider synchronization of the deliveries at the construction sites. Furthermore, no time windows on the deliveries are imposed.

Finally, in Durbin and Hoffman (2008), a decision support system is presented which assists in the dispatching of trucks, creating delivery schedules, and determining whether new orders should be accepted. The system is designed in such a way that it can coop with uncertainty and changes in the schedule caused by order cancellations, equipment failures, traffic congestions. The scheduling and routing problems handled by the system are represented by time-space networks, and are solved through MIP.

The Logic Based Benders' decomposition framework utilized in this work has recently been applied in a number of related related assignment, scheduling and routing problems. Applications include Round Robin Tournament Scheduling (Rasmussen and Trick, 2007), Tollbooth Allocation (Bai and Rubin, 2009), Parallel Machine Scheduling (Tran and Beck, 2012), Lock Scheduling (Verstichel et al., 2013) and Strip Packing (Côté et al., 2013).

## 4 A Logic-Based Benders' Decomposition

To solve the problem defined in the previous section, a logic-based Benders' decomposition is developed. The problem is decomposed in a master problem and a subproblem. The master problem, guided by the objective function, decides which customers are being serviced. To aid in this decision, a number of high-level problem characteristics are captured in the master problem, such as the availability of vehicles, their capacities, processing times and travel times between the customers and production centers. For a given subset of customers $\overline{C} \subseteq C$ selected by the master problem, the subproblem attempts to find a feasible delivery schedule in which the demand of all customers in $\overline{C}$ is satisfied, and all scheduling and routing constraints are met. Whenever such a schedule does not exist, a *feasibility* cut is generated and added to the master problem, effectively forcing the master problem to change the set $\overline{C}$. When, on the other hand, a feasible solution to the subproblem exist, a provable optimal solution to the CDP problem is obtained. An overview of the solution procedure is presented in Algorithm 1.

When compared to the MIP or CP approaches presented in Kinable et al. (2013), this decomposition approach decouples the allocation of concrete to customers from the actual routing and scheduling problem. As a consequence, many of the conditional constraints (big-M constraints) can be omitted or strengthened.

**Algorithm 1.** Combinatorial Benders Decomposition of CDP

**Output**: An optimal Concrete Delivery Schedule

```
1  repeat ← true ;
2  while repeat do
3      Solve [MP];
4      get solution (y_i), i ∈ C;
5      repeat ← false ;
6      Solve [SP] for y_i, i ∈ C;
7      if [SP] is infeasible then
8          repeat ← true ;
9          add feasibility cut(s) to [MP] ;
10     else
11         get solution (y, x, C);
12 return Optimal schedule (y, x, C)
```

### 4.1 Master Problem

The master problem is defined through the following MIP model.

$$MP: \ max \sum_{i \in C} q_i y_i \tag{1}$$

$$\sum_{k \in K} l_k z_{ki} \geq q_i y_i \qquad \forall i \in C \tag{2}$$

$$\sum_{i \in C} z_{ki} \leq \Delta_k \qquad \forall k \in K \tag{3}$$

$$\sum_{i \in S} y_i \leq |S| - 1 \qquad \forall S \in \mathbb{S}, S \subseteq C \tag{4}$$

$$y_i \in \{0, 1\} \qquad \forall i \in C \tag{5}$$

$$0 \leq z_{ki} \leq \Delta_{ki} \qquad \forall i \in C \tag{6}$$

Here, boolean variables $y_i$, $i \in C$, denote whether customer $i$ is serviced and integer variables $z_{ki}$ record the number of deliveries from vehicle $k \in K$ to customer $i \in C$. The auxiliary $z_{ki}$ variables are used to produce stronger limits on the $y_i$ variables; they are not used in the subproblem described in Section 4.2.

The first constraint, (2), links the variables $y_i$ and $z_{ki}$, $i \in C, k \in K$: a customer is satisfied if sufficient concrete is delivered. Constraints (3), (6) restrict the number of deliveries made by a single vehicle through the bounds $\Delta_{ki}$, $\Delta_k$, for all $k \in K, i \in C$. $\Delta_{ki}$, $\Delta_k$, are resp. bounds on the maximum number of deliveries vehicle $k$ can make for customer $i$, and bounds on the total number of deliveries a vehicle $k$ can make. Finally, Constraints (4) are the feasibility cuts obtained through the subproblem, prohibiting certain combinations of customers.

$\Delta_{ki}$ is calculated via Algorithm 2, whereas $\Delta_k$ is calculated via the recursive Algorithm 3. The latter algorithm utilizes a sorted array of customers; a customer $i \in C$ precedes a customer $j \in C$ in the array if $b_i < b_j \vee (b_i = b_j \wedge a_i \leq a_j)$. Computing a bound on the maximum number of deliveries a vehicle can make, is achieved by calculating a route from the starting depot 0 to the ending depot $n + 1$ through a number of customers. At each customer, the vehicle makes as many deliveries as possible. The exact number of deliveries it can make for a given customer is limited by: (1) the demand of the customer, (2) the available time to make the deliveries. In turn, the available time to perform the deliveries is limited by the time window of the customer, the processing time of the vehicle, and the time required to reload the vehicle. Furthermore, whenever the vehicle completes its last delivery for a customer $i$ at time $t^i_{compl}$, deliveries for the

**Algorithm 2.** Calculating an upper bound on the number of deliveries vehicle $k \in K$ can make for customer $i \in C$

**Input**: Vehicle $k \in K$, Customer $i \in C$

```
concreteDelivered ← 0 ;
timeConsumed ← 0 ;
Δ_ki ← 0 ;
while a_i + timeConsumed + p_k ≤ b_i ∧ concreteDelivered < q_i do
    Δ_ki ← Δ_ki + 1 ;
    concreteDelivered ← concreteDelivered + l_k ;
    timeConsumed ← timeConsumed + p_k + c_{i,i} ;
return Δ_ki
```

**Algorithm 3.** Calculating an upper bound on the number of deliveries vehicle $k \in K$ can make

**Input**: Vehicle $k \in K$, Array of customers sortedCustomers[], sorted.
**Output**: $\Delta_k$

```
1  return maxDeliveries(k, 0, 0, 0, 0) ;
2  Function int maxDeliveries(k ∈ K, Δ_k, index, i ∈ V, complTime)
3      if index = |C| then
4          return Δ_k;
5      j ← sortedCustomers[index] ;
       /* Determine how many deliveries vehicle k can make for customer
          j                                                              */
6      concreteDelivered ← 0 ;
7      startTime ← max(a_j,complTime +c_ij) ;  /* Start time 1st delivery for
       j ∈ C */
8      timeConsumed ← 0 ;
9      deliveries ← 0 ;
10     while startTime + timeConsumed + p_k ≤ b_j ∧ concreteDelivered < q_j do
11         deliveries ← deliveries + 1 ;
12         concreteDelivered ← concreteDelivered + l_k ;
13         timeConsumed ← timeConsumed + p_k + c_jj ;
14     complTimeNew ← startTime + timeConsumed − c_jj;
15     if deliveries > 0 then
16         return max(maxDeliveries(k, Δ_k +deliveries,index +1,j,complTimeNew),
17                    maxDeliveries(k, Δ_k, index +1, i, complTime));
18     else
19         return maxDeliveries(k, Δ_k, index +1, i, complTime);
```

next customer $j$ cannot commence before $t^i_{compl} + c_{ij}$. The recursive algorithm outlined in procedure 3 iterates over all possible subsets of customers in an efficient manner. At each iteration, the algorithm tracks the total number of deliveries made, the last location $i \in V$ visited, an index to the customer it will visit next, and the time it departed from location $i \in V$.

### 4.2 Subproblem

Let $\overline{y_i}$, $i \in C$, be the optimal selection of customers obtained from problem $MP$, i.e. $\overline{C} = \{i \in C : \overline{y_i} = 1\}$. To assess the feasibility of this selection, a satisfiability subproblem (SP) is solved. Whenever no feasible solution to the subproblem exists, a cut is added to the Master problem:

$$\sum_{i \in \widehat{C}} y_i \leq |\widehat{C}| - 1$$

,where $\widehat{C} \subseteq \overline{C}$. The weakest cuts are obtained for $\widehat{C} = \overline{C}$. By reducing the size of $\widehat{C}$, stronger cuts may be obtained. The strongest cuts are based on Minimum Infeasible Subsets. In this context, a MIS is a subset of customers that cannot be accommodated in the same schedule; removing any of these customers from the set would however result in a subset of compatible customers. Note however that calculating a complete set of MIS is a difficult problem on its own.

The next paragraph outlines an exact procedure to establish the feasibility of a set $\overline{C}$. As this procedure is computationally expensive, we first try to solve the subproblem through the SD-heuristic proposed in Kinable et al. (2013). Furthermore, instead of solving the subproblem for the entire set $\overline{C}$ at once, we first solve the problem for a smaller set $\widehat{C} \subset \overline{C}$. If this smaller subproblem turns out to be feasible, we repeatedly add customers from $\overline{C}$ to $\widehat{C}$ and resolve the resulting subproblem.

A modified version of the CP model presented in Kinable et al. (2013) may be used to establish the feasibility of a set $\overline{C}$ of customers:

**Algorithm 4.** CP subproblem

`Variable definitions:`

1 $s = \{0, 0, 0, oblig.\}$
2 $t = \{0, \infty, 0, oblig.\}$
3 $d^i_j = \{r_i, d_i, 0, oblig.\}$ $\quad \forall i \in \overline{C}, j \in \{1, \ldots, m(i)\}$
4 $d^i_j = \{r_i, d_i, 0, opt.\}$ $\quad \forall i \in \overline{C}, j \in \{m(i)+1, \ldots, n(i)\}$
5 $d^i_{j,k} = \{r_i, d_i, p_k, opt.\}$ $\quad \forall i \in \overline{C}, j \in \{1, \ldots, n(i)\}, k \in K$

`Constraints:`

6 **forall** $i \in \overline{C}$
7 $\quad$ **forall** $j \in \{1, \ldots, n(i)\}$
8 $\quad\quad$ alternative($d^i_j$, $\bigcup_{k \in K} d^i_{jk}$)
9 $\quad$ $\sum_{k \in K} \sum_{j \in \{1,\ldots,n(i)\}} l_k \cdot$ presenceOf($d^i_{jk}$)$\geq q_i$
10 $\quad$ **forall** $j \in \{1, \ldots, n(i)-1\}$
11 $\quad\quad$ endBeforeStart($d^i_j$, $d^i_{j+1}$)
12 $\quad\quad$ startBeforeEnd($d^i_{j+1}$, $d^i_j$, $-\gamma$)
13 $\quad$ **forall** $j \in \{m(i), \ldots, n(i)-1\}$
14 $\quad\quad$ $(\sum_{l \in \{1..j\}, k \in K} l_k \cdot$presenceOf($d^i_{lk}$)$< q_i) \rightarrow$ presenceOf($d^i_{j+1,k}$)
15 $\quad\quad$ presenceOf($d^i_{j+1}$)$\rightarrow$presenceOf($d^i_j$)
16 **forall** $k \in K$
17 $\quad$ noOverlapSequence($\bigcup_{i \in \overline{C}, j \in \{1,\ldots,n(i)\}} d^i_{jk} \cup s \cup t$)
18 $\quad$ first($s$, $\bigcup_{i \in \overline{C}, j \in \{1,\ldots,n(i)\}} d^i_{jk} \cup t$)
19 $\quad$ last($t$, $\bigcup_{i \in \overline{C}, j \in \{1,\ldots,n(i)\}} d^i_{jk} \cup s$)

The CP model utilizes a number of interval variables. For each interval variable, four parameters $\{a, b, d, o\}$ are specified, where $a$,$b$ indicate resp. the earliest start

time and latest completion time of the interval, and $d$ is the minimum length of the interval. The last parameter $o$ dictates whether an interval must (obligatory) or may (optional) be scheduled. The definitions of the constraints (Table 2) are taken from Kinable et al. (2013).

Variables $d^i_j$, $i \in \overline{C}, j \in \{1, \ldots, n(i)\}$, represent deliveries made for customer $i$. A delivery $j$ for customer $i$ is made if the corresponding interval variable $d^i_j$ is present; otherwise it is absent. Variables $d^i_{j,k}$, $i \in \overline{C}, j \in \{1, \ldots, n(i)\}, k \in K$, link deliveries and the vehicles performing these deliveries. Clearly, each delivery $j \in \{1, \ldots, n(i)\}$ for a customer $i \in \overline{C}$ can only be made by a single vehicle (Line (8)), and the amount of concrete delivered for a customer should cover its demand (Line (9)). Deliveries for the same customer may not overlap (Line (11)) and must respect a maximum time lag $\gamma$ (Line (12)). Similarly, deliveries made by a single vehicle cannot overlap in time and must comply with travel times (Line (17)). Trucks must start their trip at the starting depot, represented by variable $s$, and must return to some ending depot identified by variable $t$ after the deliveries are completed (Lines (18), (19)). Finally, Line (15) ensures that deliveries are made in order, and Line (14) tightens the link between the $d^i_j$ and $d^i_{jk}$ variables.

**Table 2.** Description of CP constraints

| Constraint | Description |
|---|---|
| **presenceOf**($\alpha$) | States that interval $\alpha$ must be present. |
| **alternative**($\alpha, B$) | If interval $\alpha$ is present, then exactly one of the intervals in set $B$ is present. The start and end of interval $\alpha$ coincides with the start and end of the selected interval from set $B$. |
| **endBeforeStart**($\alpha, \beta$) | $endOf(\alpha) \leq startOf(\beta)$. Automatically satisfied if either of the intervals is absent. |
| **startBeforeEnd**($\alpha, \beta, z$) | $startOf(\alpha) + z \leq endOf(\beta)$. 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**($\alpha, B$) | If interval $\alpha$ is present, it must be scheduled before any of the intervals in $B$. |
| **last**($\alpha, B$) | If interval $\alpha$ is present, it must scheduled after any of the intervals in $B$. |

### 4.3 Generating an Initial Set of Cuts

Before invoking the Benders' procedure, first a number of initial cuts are computed and added to thet set $\mathbb{S}$ in the master problem. These custs are generated by enumerating all Minimum Infeasible Subsets consisting of up to three customers. In addition, the constructive heuristic presented in Kinable et al. (2013)

(Algorithm 2) may be used to compute an additional set of cuts. The constructive heuristic is initialized with an ordered set of customers. The heuristic schedules deliveries for these customers in an iterative fashion, where the time of a delivery and the vehicle performing the delivery are determined via a number of heuristic criteria. If at some point, the heuristic fails to schedule the next delivery for a customer due to the lack of an available vehicle, the heuristic would normally remove this customer from the schedule and continue with the next customer. Instead of simply removing the customer from the schedule, an additional check is performed. Given the subset of customers $\widehat{C}$ which have received concrete in the partially completed heuristic schedule, an exact algorithm, e.g. the CP model from Section 4.2, is used to determine whether there exists a feasible solution where each of the customers in $\widehat{C}$ is satisfied. If no such schedule exist, a cut is generated for the customers in $\widehat{C}$ and the heuristic removes the customer it could not satisfy from the schedule. If, on the other hand, the exact approach is capable of finding a feasible schedule satisfying all customers in $\widehat{C}$, the heuristic continues from the schedule generated by the exact method.

Naturally, the approach outlined in the previous paragraph may be repeated for several orderings of the customers.

## 5 Experimental Results

### 5.1 Data Sets

Experiments are conducted on the data set published by Kinable et al. (2013) (available online at (Kinable and Wauters, 2013)). A summary of the instances is provided in Table 3 of Kinable et al. (2013). The instances are subdivided into two sets, resp. A and B. The instances in set A range from $10-20$ *customers*, and $2-5$ *vehicles*. Larger instances, having up to 50 *customers* and 20 *vehicles*, can be found in data set B. Customer demands are within the range of $10-75$ for both sets. Each instance defines a number of different vehicle classes, where vehicles belonging to the same class have the same capacity and processing time. The location of the starting and ending depot, as well as the locations of the customers and up to 4 production depots are defined per instance. The travel time between two locations equals the euclidean distance, rounded upwards. The instances are constructed in such a way that for each customer, there is a production station within $1-25$ time units. The width of the delivery interval for a customer is set proportional to the demand of the customer.

### 5.2 Experiments

A number of experiments are conducted to assess the performance of the Benders' procedure outlined in this paper. The results of these experiments are reported in Tables 4, 5. In these tables, the first column provides the instance name, following a "$W\_x\_y\_z$" naming convention, where $W$ identifies the data set, $x$ is the number of vehicles, $y$ is the number of customers and $z$ is the number of concrete production stations. For each instance, we computed an initial

**Table 3.** Data sets Kinable et al. (2013)

| | **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 | $p_k = l_k$ | $p_k = l_k$ |
| | | |
| **Stations** | 1-4 | 1-4 |
| Cust.-Station | 1-30 | 1-30 |
| Depot-Station | 1-25 | 1-25 |

feasible solution using the CP procedure outlined in Kinable et al. (2013). These solutions, reported in the second column, are used to warm-start the Benders' procedure. The next 5 columns provide data on our Benders' procedure:

- obj: The objective of the best(feasible) solution obtained through the Benders' procedure.
- iCuts: The number of cuts added initially (Section 4.3)
- cuts: The number of cuts added during the Bender's procedure (Section 4.2)
- c-time: The time required to obtain the initial master problem, in seconds. This time is limited to 5 minutes, excluding the generation of the Minimum Infeasible subsets.
- s-time: The time required to solve the Benders' problem. For data set A, this time is limited to 10 minutes, and for data set B 15 minutes.

In Kinable et al. (2013) bounds on the optimal solutions are published. These bounds are computed through four different procedures, but for each instance only the strongest bound is reported in Kinable et al. (2013). The different procedures from Kinable et al. (2013) are:

– Optimal MIP solution (when available)
– Optimal CP solution (when available)
– LP relaxation, strengthened with cuts from all Minimum Infeasible Subsets (MIS) of size 2.
– Solution to the MIP problem consisting of the objective function (1) and all cuts generated from MIS of size $k$ (Constraint (4)), where $k = 3$ for data set A. This bound has only been calculated for data set A in Kinable et al. (2013).

In columns 'bound*', and 'LP', resp. the bounds reported in Kinable et al. (2013) as well as the bounds obtained through the LP relaxation of the MIP model in Section 3.1.2 of Kinable et al. (2013) are shown. The gaps are computed with respect to the objective in column *obj*. Finally, the last two columns in the table provide the bounds obtained through the Benders' procedure.

When comparing the bounds attained through our Benders' procedure with the LP-bounds, we can observe that the LP based bounds are significantly weaker. The average gap between the LP bounds and the best primal solutions amounts to 9.26%, whereas the Benders' procedure produces an average gap of 1.81%. This gap is also smaller than the average gap (3.62%) obtained from the bounds reported in Kinable et al. (2013). In fact, none of the bounds computed through the Benders' procedure are weaker than the bounds reported in Kinable et al. (2013).

**Table 4.** Computational results Data Set A

| | | Benders' decomposition | | | | | LP | | Bound* | | Benders' | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | iObj | obj | iCuts | cuts | c-time | s-time | bound | gap | bound | gap | bound | gap |
| A_2_5_1 | 85 | 85 | 28 | 0 | 1 | 0 | 85 | 0% | 85 | 0% | 85 | 0% |
| A_2_5_2 | 160 | 160 | 13 | 0 | 0 | 0 | 160 | 0% | 160 | 0% | 160 | 0% |
| A_2_5_3 | 105 | 105 | 26 | 0 | 0 | 0 | 105 | 0% | 105 | 0% | 105 | 0% |
| A_2_5_4 | 105 | 105 | 3 | 0 | 300 | 0 | 105 | 0% | 105 | 0% | 105 | 0% |
| A_2_10_1 | 50 | 50 | 142 | 0 | 3 | 0 | 50 | 0% | 50 | 0% | 50 | 0% |
| A_2_10_2 | 150 | 150 | 215 | 0 | 9 | 0 | 150 | 0% | 150 | 0% | 150 | 0% |
| A_2_10_3 | 220 | 220 | 154 | 0 | 3 | 0 | 230 | 4% | 220 | 0% | 220 | 0% |
| A_2_10_4 | 150 | 150 | 179 | 0 | 9 | 0 | 165 | 9% | 150 | 0% | 150 | 0% |
| A_2_15_1 | 215 | 215 | 567 | 0 | 10 | 0 | 225 | 4% | 215 | 0% | 215 | 0% |
| A_2_15_2 | 290 | 290 | 373 | 2 | 25 | 2 | 320 | 9% | 320 | 9% | **290** | **0%** |
| A_2_15_3 | 205 | 205 | 502 | 0 | 77 | 0 | 215 | 5% | 205 | 0% | 205 | 0% |
| A_2_15_4 | 255 | 255 | 348 | 0 | 11 | 0 | 300 | 15% | 255 | 0% | 255 | 0% |
| A_2_20_1 | 255 | 255 | 549 | 0 | 38 | 0 | 260 | 2% | 255 | 0% | 255 | 0% |
| A_2_20_2 | 270 | 270 | 575 | 2 | 33 | 0 | 285 | 5% | 270 | 0% | 270 | 0% |
| A_2_20_3 | 260 | 260 | 651 | 0 | 15 | 0 | 280 | 7% | 260 | 0% | 260 | 0% |
| A_2_20_4 | 355 | 355 | 36 | 5 | 360 | 1 | 490 | 28% | 380 | 7% | **355** | **0%** |
| A_3_5_1 | 205 | 205 | 0 | 0 | 3 | 0 | 205 | 0% | 205 | 0% | 205 | 0% |
| A_3_5_2 | 115 | 115 | 9 | 0 | 1 | 0 | 115 | 0% | 115 | 0% | 115 | 0% |
| A_3_5_3 | 125 | 125 | 0 | 0 | 0 | 0 | 125 | 0% | 125 | 0% | 125 | 0% |
| A_3_5_4 | 190 | 190 | 0 | 0 | 0 | 0 | 190 | 0% | 190 | 0% | 190 | 0% |
| A_3_10_1 | 205 | 205 | 100 | 0 | 5 | 0 | 205 | 0% | 205 | 0% | 205 | 0% |
| A_3_10_2 | 230 | 230 | 80 | 0 | 14 | 0 | 230 | 0% | 230 | 0% | 230 | 0% |
| A_3_10_3 | 305 | 305 | 114 | 0 | 23 | 0 | 305 | 0% | 305 | 0% | 305 | 0% |
| A_3_10_4 | 300 | 300 | 115 | 0 | 5 | 0 | 300 | 0% | 300 | 0% | 300 | 0% |
| A_3_15_1 | 330 | 330 | 485 | 0 | 35 | 0 | 330 | 0% | 330 | 0% | 330 | 0% |
| A_3_15_2 | 395 | 395 | 268 | 3 | 18 | 1 | 530 | 25% | 425 | 7% | **395** | **0%** |
| A_3_15_3 | 290 | 290 | 378 | 23 | 49 | 7 | 430 | 33% | 330 | 12% | **290** | **0%** |
| A_3_15_4 | 440 | 440 | 10 | 15 | 306 | 49 | 550 | 20% | 475 | 7% | **440** | **0%** |
| A_3_20_1 | 340 | 340 | 667 | 1 | 350 | 157 | 410 | 17% | 345 | 1% | **340** | **0%** |
| A_3_20_2 | 415 | 415 | 28 | 0 | 516 | 0 | 510 | 19% | 415 | 0% | 415 | 0% |

**Table 4.** Computational results Data Set A

| | | Benders' decomposition | | | | | LP | | Bound* | | Benders' | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | iObj | obj | iCuts | cuts | c-time | s-time | bound | gap | bound | gap | bound | gap |
| A_3_20_3 | 355 | **360** | 48 | 0 | 324 | 0 | 425 | 15% | 360 | 0% | 360 | 0% |
| A_3_20_4 | 480 | 480 | 31 | 0 | 319 | 0 | 590 | 19% | 480 | 0% | 480 | 0% |
| A_4_5_1 | 140 | 140 | 0 | 0 | 0 | 0 | 140 | 0% | 140 | 0% | 140 | 0% |
| A_4_5_2 | 150 | 150 | 0 | 0 | 0 | 0 | 150 | 0% | 150 | 0% | 150 | 0% |
| A_4_5_3 | 165 | 165 | 0 | 0 | 0 | 0 | 165 | 0% | 165 | 0% | 165 | 0% |
| A_4_5_4 | 230 | 230 | 0 | 0 | 0 | 0 | 230 | 0% | 230 | 0% | 230 | 0% |
| A_4_10_1 | 310 | 310 | 114 | 0 | 46 | 0 | 310 | 0% | 310 | 0% | 310 | 0% |
| A_4_10_2 | 370 | 370 | 48 | 0 | 4 | 0 | 390 | 5% | 370 | 0% | 370 | 0% |
| A_4_10_3 | 375 | 375 | 85 | 1 | 42 | 2 | 470 | 20% | 445 | 16% | **375** | **0%** |
| A_4_10_4 | 285 | 285 | 23 | 0 | 1 | 0 | 285 | 0% | 285 | 0% | 285 | 0% |
| A_4_15_1 | 415 | 415 | 7 | 2 | 307 | 600 | 570 | 27% | 545 | 24% | 545 | 24% |
| A_4_15_2 | 475 | 475 | 4 | 18 | 312 | 600 | 650 | 27% | 610 | 22% | **520** | **9%** |
| A_4_15_3 | 430 | 430 | 368 | 0 | 99 | 0 | 495 | 13% | 450 | 4% | **430** | **0%** |
| A_4_15_4 | 490 | 490 | 5 | 0 | 304 | 600 | 515 | 5% | 515 | 5% | 515 | 5% |
| A_4_20_1 | 525 | 525 | 3 | 8 | 302 | 600 | 660 | 20% | 585 | 10% | **575** | **9%** |
| A_4_20_2 | 425 | 425 | 497 | 4 | 38 | 1 | 490 | 13% | 440 | 3% | **425** | **0%** |
| A_4_20_3 | 375 | 375 | 40 | 12 | 828 | 600 | 530 | 29% | 425 | 12% | **405** | **7%** |
| A_4_20_4 | 465 | 465 | 18 | 17 | 1801 | 3 | 590 | 21% | 500 | 7% | **465** | **0%** |
| A_5_5_1 | 200 | 200 | 0 | 0 | 0 | 0 | 200 | 0% | 200 | 0% | 200 | 0% |
| A_5_5_2 | 200 | 200 | 0 | 0 | 0 | 0 | 200 | 0% | 200 | 0% | 200 | 0% |
| A_5_5_3 | 220 | 220 | 0 | 0 | 0 | 0 | 220 | 0% | 220 | 0% | 220 | 0% |
| A_5_5_4 | 175 | 175 | 9 | 0 | 2 | 0 | 175 | 0% | 175 | 0% | 175 | 0% |
| A_5_10_1 | 350 | 350 | 0 | 0 | 0 | 0 | 350 | 0% | 350 | 0% | 350 | 0% |
| A_5_10_2 | 345 | 345 | 0 | 0 | 0 | 0 | 345 | 0% | 345 | 0% | 345 | 0% |
| A_5_10_3 | 285 | 285 | 25 | 0 | 5 | 0 | 300 | 5% | 285 | 0% | 285 | 0% |
| A_5_10_4 | 380 | 380 | 0 | 0 | 0 | 0 | 380 | 0% | 380 | 0% | 380 | 0% |
| A_5_15_1 | 455 | 455 | 2 | 4 | 308 | 600 | 590 | 23% | 590 | 23% | **510** | **11%** |
| A_5_15_2 | 580 | 580 | 0 | 0 | 301 | 600 | 695 | 17% | 695 | 17% | 695 | 17% |
| A_5_15_3 | 350 | 350 | 2 | 3 | 576 | 600 | 435 | 20% | 395 | 11% | **385** | **9%** |
| A_5_15_4 | 500 | 500 | 3 | 0 | 1218 | 600 | 600 | 17% | 520 | 4% | 520 | 4% |
| A_5_20_1 | 700 | 705 | 252 | 61 | 245 | 536 | 900 | 22% | 760 | 7% | **705** | **0%** |
| A_5_20_2 | 555 | 555 | 15 | 4 | 388 | 600 | 810 | 31% | 645 | 14% | **630** | **12%** |
| A_5_20_3 | 595 | 595 | 8 | 16 | 302 | 600 | 695 | 14% | 645 | 8% | **615** | **3%** |
| A_5_20_4 | 520 | 520 | 13 | 2 | 355 | 600 | 705 | 26% | 560 | 7% | **557.5** | **7%** |
| AVG | | | | | | | | 9.26% | | 3.72% | | 1.81% |
| Optimal | | | | | | | | 28 | | 41 | | 52 |

In summary, we reduced the average gap for the instances in data set A from 3.72% to 1.81%, 19 instances had their bounds improved, and 11 new optimal solutions were found. For almost 65% of the instances, the optimal solutions were already obtained at the first iteration of the master problem, i.e. no additional cuts had to be generated.

**Table 5.** Computational results Data Set B

| | | Benders' decomposition | | | | | LP | | Bound* | | Benders' | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | iObj | obj | iCuts | cuts | c-time | s-time | bound | gap | bound | gap | bound | gap |
| B_6_20_1 | 725 | 725 | 0 | 102 | 301 | 356 | 805 | 10% | 805 | 10% | **725** | **0%** |
| B_6_20_2 | 700 | 700 | 0 | 31 | 300 | 900 | 855 | 18% | 855 | 18% | **830** | **16%** |
| B_6_20_3 | 675 | 675 | 0 | 0 | 300 | 900 | 760 | 11% | 760 | 11% | 760 | 11% |
| B_6_20_4 | 615 | 615 | 0 | 29 | 300 | 900 | 705 | 13% | 705 | 13% | **680** | **10%** |
| B_6_30_1 | 860 | 860 | 0 | 1 | 300 | 900 | 1300 | 34% | 1300 | 34% | **1290** | **33%** |
| B_6_30_2 | 850 | 850 | 0 | 34 | 300 | 900 | 1140 | 25% | 1140 | 25% | **1105** | **23%** |
| B_6_30_3 | 725 | 725 | 0 | 120 | 301 | 900 | 1060 | 32% | 1060 | 32% | **1035** | **30%** |
| B_6_30_4 | 625 | 625 | 2 | 1 | 302 | 900 | 1000 | 38% | 1000 | 38% | **913.33** | **32%** |
| B_6_40_1 | 835 | 835 | 0 | 23 | 302 | 900 | 1545 | 46% | 1545 | 46% | **1431.11** | **42%** |
| B_6_40_2 | 1045 | 1045 | 0 | 33 | 301 | 900 | 1635 | 36% | 1635 | 36% | **1473.75** | **29%** |
| B_6_40_3 | 735 | 735 | 11 | 79 | 315 | 900 | 1570 | 53% | 1570 | 53% | **1191.67** | **38%** |
| B_6_40_4 | 750 | 750 | 2 | 9 | 305 | 900 | 1450 | 48% | 1450 | 48% | **1095** | **32%** |
| B_6_50_1 | 1010 | 1010 | 0 | 22 | 305 | 900 | 1890 | 47% | 1890 | 47% | **1490** | **32%** |
| B_6_50_2 | 955 | 955 | 2 | 25 | 307 | 900 | 2250 | 58% | 2250 | 58% | **1495** | **36%** |
| B_6_50_3 | 920 | 920 | 1 | 16 | 311 | 900 | 1740 | 47% | 1740 | 47% | **1295** | **29%** |
| B_6_50_4 | 1000 | 1000 | 1 | 0 | 385 | 900 | 2080 | 52% | 2080 | 52% | **1395** | **28%** |
| B_8_20_1 | 920 | 920 | 0 | 1 | 300 | 897 | 935 | 2% | 935 | 2% | **920** | **0%** |
| B_8_20_2 | 850 | 850 | 0 | 0 | 300 | 900 | 865 | 2% | 865 | 2% | 865 | 2% |
| B_8_20_3 | 655 | 655 | 0 | 0 | 293 | 0 | 655 | 0% | 655 | 0% | 655 | 0% |
| B_8_20_4 | 820 | 820 | 0 | 0 | 69 | 0 | 820 | 0% | 820 | 0% | 820 | 0% |
| B_8_30_1 | 920 | 920 | 0 | 0 | 300 | 900 | 1085 | 15% | 1085 | 15% | 1085 | 15% |
| B_8_30_2 | 1005 | 1005 | 0 | 0 | 300 | 900 | 1115 | 10% | 1115 | 10% | 1115 | 10% |
| B_8_30_3 | 975 | 975 | 0 | 4 | 300 | 900 | 1155 | 16% | 1155 | 16% | **1140** | **14%** |
| B_8_30_4 | 1110 | 1110 | 0 | 0 | 300 | 900 | 1320 | 16% | 1320 | 16% | 1320 | 16% |
| B_8_40_1 | 1180 | 1180 | 0 | 1 | 301 | 900 | 1665 | 29% | 1665 | 29% | **1655** | **29%** |
| B_8_40_2 | 1190 | 1190 | 0 | 6 | 300 | 900 | 1415 | 16% | 1415 | 16% | 1415 | 16% |
| B_8_40_3 | 995 | 995 | 0 | 0 | 300 | 900 | 1495 | 33% | 1495 | 33% | 1495 | 33% |
| B_8_40_4 | 1105 | 1105 | 0 | 0 | 301 | 900 | 1730 | 36% | 1730 | 36% | 1730 | 36% |
| B_8_50_1 | 1130 | 1130 | 0 | 11 | 323 | 901 | 1980 | 43% | 1980 | 43% | **1925** | **41%** |
| B_8_50_2 | 1150 | 1150 | 0 | 0 | 304 | 900 | 1935 | 41% | 1935 | 41% | 1935 | 41% |
| B_8_50_3 | 1210 | 1210 | 0 | 0 | 302 | 900 | 1960 | 38% | 1960 | 38% | 1960 | 38% |
| B_8_50_4 | 1105 | 1105 | 0 | 0 | 303 | 900 | 1835 | 40% | 1835 | 40% | **1740** | **36%** |
| B_10_20_1 | 805 | 805 | 0 | 0 | 116 | 0 | 805 | 0% | 805 | 0% | 805 | 0% |
| B_10_20_2 | 825 | 825 | 0 | 0 | 237 | 0 | 825 | 0% | 825 | 0% | 825 | 0% |
| B_10_20_3 | 730 | 730 | 0 | 0 | 4 | 0 | 730 | 0% | 730 | 0% | 730 | 0% |
| B_10_20_4 | 765 | 765 | 0 | 0 | 1 | 0 | 765 | 0% | 765 | 0% | 765 | 0% |
| B_10_30_1 | 910 | 910 | 0 | 0 | 300 | 900 | 1215 | 25% | 1215 | 25% | 1215 | 25% |
| B_10_30_2 | 1170 | 1170 | 0 | 0 | 300 | 900 | 1355 | 14% | 1355 | 14% | 1355 | 14% |
| B_10_30_3 | 1135 | 1135 | 0 | 0 | 300 | 900 | 1210 | 6% | 1210 | 6% | 1210 | 6% |
| B_10_30_4 | 1165 | 1165 | 0 | 0 | 300 | 900 | 1235 | 6% | 1235 | 6% | 1235 | 6% |
| B_10_40_1 | 1210 | 1210 | 0 | 0 | 301 | 900 | 1475 | 18% | 1475 | 18% | 1475 | 18% |
| B_10_40_2 | 1485 | 1485 | 0 | 0 | 301 | 900 | 1580 | 6% | 1580 | 6% | 1580 | 6% |
| B_10_40_3 | 1375 | 1375 | 0 | 0 | 302 | 900 | 1605 | 14% | 1605 | 14% | 1605 | 14% |
| B_10_40_4 | 1365 | 1365 | 0 | 0 | 301 | 900 | 1455 | 6% | 1455 | 6% | 1455 | 6% |
| B_10_50_1 | 1425 | 1425 | 0 | 0 | 308 | 900 | 2265 | 37% | 2265 | 37% | 2265 | 37% |
| B_10_50_2 | 1010 | 1010 | 0 | 0 | 302 | 900 | 1900 | 47% | 1900 | 47% | **1745** | **42%** |
| B_10_50_3 | 1260 | 1260 | 0 | 0 | 302 | 900 | 2005 | 37% | 2005 | 37% | 2005 | 37% |
| B_10_50_4 | 1455 | 1455 | 0 | 0 | 303 | 901 | 1925 | 24% | 1925 | 24% | 1925 | 24% |
| AVG | | | | | | | | 23.83% | | 23.83% | | 20.51% |
| Optimal | | | | | | | | 6 | | 6 | | 8 |

The instances in data set B are significantly harder to solver than the instances in data set A. Table 5 shows the results obtained for the instances up to 10 vehicles; no improvements could be made to the remaining instances. For the instances in Table 5, the average gap induced by the Benders' bounds is 10.08%, opposed to 11.32% from the bounds published in Kinable et al. (2013). Furthermore, optimality was attained for two additional instances; 21 instances had their bounds improved.

Future attempts to improve the Benders' decomposition approach should be targeted at improving the the runtime of the subproblem, as this procedure takes the largest amount of time. Especially for the larger instances, solving the subproblem is challenging.

## 6 Conclusion

In this work, we presented a Logic Based Benders' decomposition which decouples the CDP into a master problem and a subproblem. The master problem allocates concrete to customers, whereas the subproblem handles the routing and scheduling of the concrete delivery trucks. By decomposing the problem, part of the complexity is shifted to the subproblem. Furthermore, dedicated procedures may be used to solve these problems. Here, we solve the master problem through MIP, whereas the subproblem is solved through a dedicated scheduling heuristic and a CP model from (Kinable et al., 2013). Because the subproblem does not have to deal with the allocation of concrete as this is being handled by the master problem, we simplified and strengthened the latter CP model, thereby significantly improving its performance.

Computing bounds for CDP is a non-trivial task. Linear Programming-based bounds are generally very weak (see Section 5), as the problem has a large number of conditional constraints. Furthermore, exact approaches based on CP often provide little insight as to the quality of the solution. Our Benders' decomposition may however provide a viable alternative. Extensive computational tests show that the bounds computed through the Benders' decomposition are consistently stronger than the bounds in Kinable et al. (2013), which where obtained by aggregating the bounds of four different procedures.

By improving the bounds for a large number of instances, and simultaneously improving several primal solutions, we were able to solve a number of previously unsolved benchmark instances to optimality. To further enhance the performance of this Benders' procedure, one would have to find a way to speed up the subproblem. One possible direction would be to decouple the subproblem even further, for example by fixing more variables in the master problem. Alternatively, one could try to replace the current subproblem by a relaxation of the exact subproblem, which is easier to solve. Using this relaxation, it could still be possible to add a number of cuts to the master problem, thereby refining the master problem, with significantly less computational effort.

## References

Asbach, L., Dorndorf, U., Pesch, E.: Analysis, modeling and solution of the concrete delivery problem. European Journal of Operational Research 193(3), 820–835 (2009)

Bai, L., Rubin, P.A.: Combinatorial benders cuts for the minimum tollbooth problem. Operations Research 57(6), 1510–1522 (2009)

Côté, J.-F., Dell'Amico, M., Iori, M.: Combinatorial benders' cuts for the strip packing problem. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation and Department of Computer Science and Operations Research, Université de Mon (CIRELT), Tech. Rep. CIRRELT-2013-27 (April 2013)

Durbin, M., Hoffman, K.: OR Practice - The Dance of the Thirty-Ton Trucks: Dispatching and Scheduling in a Dynamic Environment. Operations Research 56(1), 3–19 (2008)

Hertz, A., Uldry, M., Widmer, M.: Integer linear programming models for a cement delivery problem. European Journal of Operational Research 222(3), 623–631 (2012)

Kinable, J., Wauters, T.: CDPLib (2013), `https://sites.google.com/site/cdplib/`

Kinable, J., Wauters, T., Vanden Berghe, G.: The Concrete Delivery Problem. Computers & Operations Research (2014) (in press)

Naso, D., Surico, M., Turchiano, B., Kaymak, U.: Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete. European Journal of Operational Research 177(3), 2069–2099 (2007)

Rasmussen, R., Trick, M.: A benders approach for the constrained minimum break problem. European Journal of Operational Research 177(1), 198–213 (2007)

Schmid, V., Doerner, K.F., Hartl, R.F., Savelsbergh, M.W.P., Stoecher, W.: A hybrid solution approach for ready-mixed concrete delivery. Transportation Science 43(1), 70–85 (2009)

Schmid, V., Doerner, K.F., Hartl, R.F., Salazar-Gonzlez, J.-J.: Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. Computers and Operations Research 37(3), 559–574 (2010)

Silva, C., Faria, J.M., Abrantes, P., Sousa, J.M.C., 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 2005, pp. 7633–7638 (2005)

Tran, T.T., Beck, J.C.: Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 774–779. IOS Press (2012)

Verstichel, J., Kinable, J., Vanden Berghe, G., De Causmaecker, P.: A combinatorial benders decomposition for the lock scheduling problem. KU Leuven, Tech. Rep., `http://allserv.kahosl.be/ jannes/lockscheduling/ combinatorialBenders_19112013.pdf` (September 2013)

Yan, S., Lai, W.: An optimal scheduling model for ready mixed concrete supply with overtime considerations. Automation in Construction 16(6), 734–744 (2007)