

A GRASP algorithm for the concrete delivery problem.

Ousmane Ali⁽¹⁾, Jean-François Côté⁽²⁾, Leandro C. Coelho⁽³⁾

(1) `nassoma-wattara-ousmane.ali.1@ulaval.ca`

(2) `Jean-Francois.Cote@fsa.ulaval.ca`

(3) `Leandro.Coelho@fsa.ulaval.ca`

August 19, 2023

Abstract

The concrete delivery problem (CDP) is a combinatorial optimization problem that involves scheduling the delivery of ready-mixed concrete to construction sites while balancing the conflicting goals of minimizing transportation costs and maximizing customer satisfaction. This paper proposes an exact formulation and a heuristic approach based on the Greedy Randomized Adaptive Search Procedure (GRASP) to tackle a new variant of the CDP that incorporates realistic side constraints, such as drivers' working shifts, minimum working time for each driver, and overtime penalties. Additionally, the problem also considers the possibility of customers requiring multiple types of concrete to be delivered within the same time window. The performance of the proposed heuristic is evaluated on real-world instances, and it is compared to another variant to demonstrate its effectiveness.

Keywords: Vehicle scheduling; concrete delivery; GRASP; ready-mixed concrete

1 Introduction

Concrete is a widely used building material in construction projects. Its perishable nature is affected by many factors that impact its quality [Sinha et al., 2021], which is crucial for the durability and strength of the final construction. Concrete comes in two types: ready-mixed concrete (RMC) and site-mixed concrete (SMC). RMC is manufactured in a batch plant and delivered to the construction site, while SMC is produced on-site using raw materials stored on the construction site. Using SMC can avoid delays caused by road traffic, but it has a slower and more difficult production process, requires storage for mixing materials and equipment, and is suitable for low amounts of concrete. On the other hand, RMC has better quality and benefits from lower production costs [Muresan, 2019]. However, to take advantage of these benefits, the batch plant manager must ensure efficient and prompt delivery on the construction site, which may require a fleet of high-cost revolving drum trucks (concrete mixers) to dispatch the RMC.

Concrete delivery under the form of RMC is subject to many operational constraints that make the Concrete Delivery Problem (CDP) very challenging. In this paper, we study a variant of the CDP to schedule the daily production and dispatching of RMC for a company located in the province of Quebec, Canada. This company operates multiple batch plants with varying production rates, using a fleet of concrete mixers of different capacities. Each plant has its own fleet of trucks; however, under certain conditions, the trucks can move between plants if necessary. The trucks must return to their home plant at the end of the

day. The company owns two types of trucks with different capacities and can call on an external fleet when needed. They serve construction sites from any of their plants, with the first delivery starting at the time specified by the customer. The loading and unloading of a concrete mixer depend on the truck capacity, the loading rate at the plant, and the unloading rate at the construction site. Drivers are allocated based on their daily work schedules. A customer may request several types of concrete to be delivered within the same time window, with no required sequence for the orders, but an order can only start after the completion of the previous order. This constraint generalizes the linked order constraints of Durbin and Hoffman [2008] where some customers place two orders for the same day and request that they are linked (the second order begins only after the first order is completed). The setting of our study is similar to a variant of the CDP previously studied in Schmid et al. [2009, 2010]. However, we include the plant’s production rate and driver shift schedule. The company uses a centralized dispatcher system to schedule all daily orders, but this system has issues satisfying all daily demands without using an external fleet.

According to Blazewicz et al. [2019], the CDP combines vehicle routing with scheduling issues to plan routes to deliver concrete from batch plants (depots) to customers’ construction sites. RMC is an on-demand product with a short life cycle from production through end use. It cannot be stored and cannot stay too long on a truck, or it will harden. Hence, concrete mixers must deliver RMC at the planned construction site shortly after its production. Tommelein and Li [1999] describes RMC production and delivery as an example of a just-in-time (JIT) production system in construction. A customer quantity requirement is often greater than the truck size and must be fulfilled by multiple deliveries. In that sense, CDP is similar to the vehicle routing problem with split delivery [Archetti and Speranza, 2008], except that the same truck may visit a customer more than once. Concrete hardens quickly, so multiple deliveries must be done back-to-back or at least close in time to avoid the problem of cold joint, which can reduce the strength and durability of the concrete. Customers request to be served within a specific time window, which can complicate truck-loading schedules when a plant can only load one truck at a time. Similarly, only one truck can unload at a time at a customer location, sometimes leading to concrete mixers queuing and waiting their turn to deliver. Furthermore, with trucks of varied sizes, loading, travel, and unloading times that may be uncertain, the CDP is a complex and challenging problem.

In this paper, we propose a mathematical model and a Greedy Randomized Search Procedure (GRASP) heuristic to solve a new variant of the CDP. Our model takes into consideration the working shifts of the drivers and the scheduling of multiple orders within the same time window at a construction site. To the best of our knowledge, this paper is the first that deals with these specific constraints in the context of the CDP.

The rest of the paper is structured as follows: Section 2 provides a literature review of previous work related to the CDP. Section 3 provides a formal description and mathematical model of the problem. In Section 4, we describe the GRASP algorithm and the constructive heuristics developed to solve this variant of the CDP. Section 5 presents computational experiments to evaluate the proposed approach, and finally, the conclusions are presented in Section 6.

2 Literature review

Academic Research on concrete batch and delivery began in the late 1990s. Tommelein and Li [1999] described RMC as a prototypical example of a JIT production system in construction and identified two practices for delivering it. One approach is for the customer to haul the product from the batch plant with their concrete mixer, while the other is for the

84 batch plant to deliver the concrete directly to the customer’s location. This latter approach
 85 is the one that has been studied in all related papers found in the literature. Several
 86 works to schedule and dispatch concrete production and delivery have mainly focused on
 87 simulation-related methods. These methods can be standalone, such as those used by Zayed
 88 and Halpin [2001], Wang et al. [2001], Tian et al. [2010], Panas and Pantouvakis [2013] and
 89 Galić and Kraus [2016], among others. Alternatively, these methods can be hybridized with
 90 optimization techniques, such as those used by Feng et al. [2004], Lu and Lam [2005], and
 91 Feng and Wu [2006]. Wang et al. [2001] developed a simulation model to reveal the effect
 92 and value of the concrete mixers’ inter-arrival time on the productivity of hired unloading
 93 equipment on site. Feng et al. [2004] used a combination of genetic algorithm (GA) and
 94 simulation process to minimize the total waiting time for trucks at a customer site. The
 95 study focused on loading trucks with identical capacities at the same batch plant, with fixed
 96 loading and unloading durations. The GA was used to find the best loading sequence of RMC
 97 trucks to be assigned to different construction sites. The simulation process determined
 98 the loading, arrival, departure, and waiting time of trucks and thus evaluated the cost of
 99 each dispatching sequence. They evaluated their method using data from a batch plant
 100 in Taiwan with up to nine customers served. Mayteekrieangkrai and Wongthatsanekorn
 101 [2015] addressed the same problem with the same data using a bee algorithm (BA) and
 102 found better solutions than the GA. Lu and Lam [2005] used the same combination of GA
 103 and simulation to determine the optimal number of concrete mixers to be deployed and an
 104 optimal schedule for batch and delivering concrete. Their objective was to minimize the idle
 105 time of the site crew due to late concrete deliveries and truck queuing time. In this setting,
 106 it was also necessary to deliver a batch of mortar on-site to lubricate the unloading pump
 107 before the concrete delivery. As such, the simulation model also included the batch and
 108 delivery of mortar. Finding the best RMC fleet size was also the purpose of the discrete-
 109 event simulation model proposed by Panas and Pantouvakis [2013].

110 In addition to simulation-based methods, several other approaches have been used in the
 111 literature to solve the CDP. These include metaheuristics [Faria et al., 2006, Misir et al.,
 112 2011, Maghrebi et al., 2016b, Yang et al., 2022], exact methods [Yan and Lai, 2007, Asbach
 113 et al., 2009, Kinable et al., 2014], matheuristics [Schmid et al., 2009, 2010], Benders Decom-
 114 position [Maghrebi et al., 2014a], column generation (CG) [Maghrebi et al., 2014b, 2016a],
 115 Lagrangian relaxation [Narayanan et al., 2015], and machine learning approaches such as
 116 those used by Graham et al. [2006], Maghrebi and Waller [2014], Maghrebi et al. [2016c].
 117 Matsatsinis [2004] designed a decision support system (DSS) for the dynamic routing of
 118 both concrete and pumps that may be necessary for some construction sites to aid in the
 119 unloading of concrete. The DSS considered the availability of three plants but stipulated
 120 that vehicles fulfilling the same order must all load at the same plant. Orders that could
 121 not be executed immediately could be postponed for the next day. The routing of the
 122 pumps was modeled as a multi-depot vehicle routing problem with time windows. Naso
 123 et al. [2007] proposed a sequential GA method combined with constructive heuristics to
 124 solve another variant of the CDP. In this problem, the plant’s production schedule must
 125 account for orders for concrete to be delivered to a customer site and orders that customers
 126 must pick up themselves. The algorithm first schedules the plant loading operations before
 127 scheduling truck deliveries. The authors also developed a non-linear model that minimizes
 128 transportation costs, waiting times, outsourced costs, and overtime work. They ran experi-
 129 ments using real-world instances of a concrete supply chain in the Netherlands and found a
 130 reduction in the number of outsourced requests. Yan and Lai [2007] also considered overtime
 131 considerations in their paper, which focused on scheduling RMC for one batch plant with
 132 two loading docks. The study took into account that overtime wages are paid for factory

133 and construction site operations after 4 PM. They developed a mixed-integer programming
 134 (MIP) model on a time-space network to minimize travel times and operating costs at both
 135 normal and overtime working hours at the plant and the construction sites. They tested
 136 the model using real data consisting of three days of operation using a two-stage algorithm.
 137 First, they solved the MIP relaxation with CPLEX. Then, they simplified the original model
 138 by fixing some decision variables before solving it. The algorithm was found to improve the
 139 actual plant operation by 10%. A time-space network is the key component of the real-time
 140 DSS developed by Durbin and Hoffman [2008] to solve a dynamic CDP every five minutes.
 141 The DSS is able to receive new orders, schedule them on the fly, and handle unexpected
 142 events such as plant closures, truck breakdowns, and delays in transportation times. The
 143 authors combined the DSS with a tabu search (TS) heuristic to warm start CPLEX, which
 144 made the model performant enough to solve instances with up to 1,500 loads per day with
 145 up to 250 trucks. The DSS also considers the case of a customer who places two orders,
 146 with the first being completed before the second starts. Further insights on the real-time
 147 planning and monitoring of CDP are available in Garza Cavazos [2021]. Another variant
 148 of the CDP is modeled by Schmid et al. [2009] as an integer multicommodity network flow
 149 (MCNF) problem on a time-space network. In this paper, concrete is delivered using a
 150 heterogeneous fleet of vehicles, and each plant can load an unlimited number of trucks si-
 151 multaneously. Some of the trucks have specialized equipment and must arrive first at certain
 152 construction sites to assist in unloading the concrete. The objective is to fulfill all orders,
 153 minimize the travel cost, and avoid delays between two consecutive unloading operations
 154 for an order. The model is typically solved using a matheuristic algorithm that combines
 155 the MCNF with a variable neighborhood search (VNS) heuristic. The method can quickly
 156 solve large problem instances with more than 60 orders per day without encountering any
 157 memory issues. The same problem is addressed by Schmid et al. [2010], who proposed a
 158 MIP model combined with a VNS and a very large neighborhood search (VLNS) to develop
 159 two matheuristics approaches. Comparisons between both matheuristics and a standalone
 160 VNS show that the former methods are much better and suitable for solving larger problem
 161 instances. These methods also provide better solutions for small to medium instances than
 162 the matheuristic used in Schmid et al. [2009]. A pure VNS approach with the same problem
 163 but without the use of instrumentation has been applied by Payr and Schmid [2009].

164 Regarding objectives, most authors have focused on minimizing travel time and delays
 165 between consecutive deliveries. However, some authors have been more interested in max-
 166 imizing customer satisfaction alone. We find these situations in the works of Durbin and
 167 Hoffman [2008], Kinable et al. [2014], Kinable and Trick [2014], Sulaman et al. [2017]. Kin-
 168 able et al. [2014] introduce a general MIP and constraint programming (CP) models of the
 169 CDP reflecting the main constraints commonly found in all CDP works: time lag and no
 170 overlapping between consecutive deliveries, covering of all customers' demands, delivery time
 171 window, and heterogeneous fleet. However, the model did not include constraints limiting
 172 the time that concrete may reside in a truck. The authors propose a constructive heuristic
 173 that schedules the visits to the customers one by one according to the start time of the
 174 visit and the truck capacity. The procedure is invoked multiple times for different permu-
 175 tations of the customer's order which is determined using the steepest descent (SD) local
 176 search procedure. One of the paper's main contributions is the creation of the first public
 177 test instances for the CDP with up to 50 customers, four batch plants, and 20 concrete
 178 mixers. They found the CP model to be highly effective in finding high-quality solutions in
 179 a relatively short time or improving existing schedules, while the MIP model can be used
 180 to compute bounds, as it seems ineffective in solving large problem instances. Finally, the
 181 heuristic often yields good solutions in less than a second. A detailed analysis of the MIP

model presented in Kinable et al. [2014] and of two more compact models can be found in the thesis of Hernández López [2020]. In Kinable and Trick [2014], we found an attempt to solve the previous problem with a logic-based Benders' approach. A generalization of the MIP model of Kinable et al. [2014] is addressed in Asbach et al. [2009]. This model simultaneously minimizes the total sum of travel costs and the penalty costs for customers with unfulfilled demand. A customer can request that all concrete deliveries come from the same plant or a subset of plants and that a delivery truck belongs to a subset of the vehicle fleet. The MIP model is used in a local search scheme as a black-box solver to reoptimize an incumbent solution in which a neighborhood operator has unfixed some variables. Tzanetos and Blondin [2023] provide an overview of the various methods used in the literature to address the CDP and categorizes the problem formulations based on the different concepts used in the literature. They also discussed the consistency between industry needs and existing constraints and provided insights into the datasets corresponding to real-world cases, identifying the necessary data for practitioners.

3 Problem description

The focus of this paper is on the distribution of RMC from a Canadian company that operates in the greater Montreal area. When a customer places an order, it is received at a control center and immediately assigned to one of the company's batch plants. These plants produce the concrete and then deliver it to the customer. The problem we are examining involves a set of customer orders, a set of concrete-mixer drivers, and a set of batch plants.

A customer i requests one or more types of concrete to be delivered to their construction site on a specific day, with the delivery service starting at the due time a_i . We call an order o a request for a specific type of concrete. q_i is the sum of the demands q_o of each order o placed, a_i is the desired arrival time of the first concrete mixer, and τ_i^u is the unloading rate for customer i . If the order requires more concrete than a single truck can carry, multiple deliveries are scheduled.

Let O_i be the set of all orders requested by customer i . Each element of O_i must be fully delivered before moving on to another order. Exactly one order $o \in O_i$ must have its first delivery start at a_i , while the others can start at most γ^2 time after o is completed. A plant is assigned to an order, and it must be the supplier of all subsequent deliveries of that order. To avoid cold joint problems with the concrete, subsequent deliveries of the same order must be made in close succession. We define a maximum time delay γ^1 after which no more deliveries are allowed. The customer's unloading rate and the quantity to be unloaded give the time required to unload a truckload. Let \mathcal{C} be the set of construction sites (customers) with a planned delivery for the day, and $\mathcal{O} = \{O_i, i \in \mathcal{C}\}$ be the set of all requested orders for all customers.

The company has two types of concrete mixer trucks with capacities of 8 and 12 cubic meters. Each driver k is assigned to a particular batch plant and is responsible for driving a truck with capacity Q_k . The set of drivers is represented by $K = \cup_{b \in \mathcal{B}} K_b$, where K_b is the set of drivers scheduled to start their shift at batch plant b . t_{ij} is the known time to travel between any two locations i and j . A scheduled driver k is required to start his shift at H_k , work a minimum of M_T hours and a maximum of N_T hours during regular working hours, with the possibility of overtime of up to O_T hours. β_3 and β_4 are the penalties for a driver working less than M_T and more than N_T .

A driver typically loads RMC at his assigned batch plant but may be required to drive to and load at other plants if needed. The batch plant produces concrete on demand using recipes specific to each order. This means that a truck can only haul RMC for one order,

even if there is spare capacity. After unloading the RMC, driver k takes ρ minutes to clean the concrete mixer before proceeding.

Let n_o be the number of deliveries needed to fulfill the order o . n_o is not known in advance because we use a fleet of trucks with different capacities. However, we can compute its lower (n_o^{min}) and upper (n_o^{max}) bounds using the capacities of the largest (Q_{max}) and smallest (Q_{min}) available trucks.

$$n_o^{min} = \left\lceil \frac{q_o}{Q_{max}} \right\rceil \leq n_o \leq n_o^{max} = \left\lceil \frac{q_o}{Q_{min}} \right\rceil \quad \forall o \in \mathcal{O}. \quad (1)$$

Let d_o^j be the j^{th} visit with load q_o^j for order o . We represent the fulfillment of order o by the visits to the ordered set of delivery nodes $\mathcal{D}_o = (d_o^0, d_o^1, \dots, d_o^{n_o})$. The deliveries of customer i are the ordered set $\mathcal{D}_i = (\mathcal{D}_{o_1}, \mathcal{D}_{o_2}, \dots, \mathcal{D}_{o_{|O_i|}})$, where o_r is the r^{th} delivered order. We will refer to $d \in \mathcal{D}_i$ ($d \in \mathcal{D}_o$) as the d^{th} potential delivery of customer i (order o). $\mathcal{D} = \bigcup_{i \in \mathcal{C}} \mathcal{D}_i$ is the union of all delivery nodes.

Each of the company's plants has a single loading dock that can accommodate only one truck at a time. As a result, trucks often form a queue while waiting for their turn at the loading dock. Let \mathcal{B} be the set of batch plants. The plants are heterogeneous, as each plant b has its own hourly loading rate, represented by τ_b^l , which affects the duration of the loading process. After loading the concrete, the driver spends α_b minutes adjusting the concrete in the truck before heading to the customer site. Each plant has its own assigned fleet of trucks, but it can borrow trucks from other plants. Let l_j be the loading dock node associated with delivery node j . After loading RMC at l_j , it must be fully delivered to j at most before Δ time, which is the concrete lifetime. We denote \mathcal{L}_b as the set of loading dock nodes of plant b and \mathcal{L} as the set of all loading docks.

A solution to the problem involves decisions about truck loading schedules, driver assignments to different deliveries, and truck arrival times at construction sites for unloading. For a batch plant, the decision involves choosing which driver to load, when to load them, how much to load, and which construction site to deliver. For a driver, the decision is to determine the sequence of loading depots and delivery sites. And for a construction site, the decision involves determining the arrival times of all scheduled deliveries for the day.

Each driver leaves and returns to their home plant every day. We represent the home plant of a driver k with a starting depot s_k and an ending depot e_k . S and E are the sets of starting and ending depots, respectively.

We define our problem on a complete directed graph where $V = \{S \cup \mathcal{L} \cup \mathcal{D} \cup E\}$ is the set of nodes. The arc sets are $A = \{(i, j, k) \mid i, j \in V, k \in K\}$, $A^D = \{(i, j) \mid i, j \in \mathcal{D}\}$, and $A^L = \{(i, j) \mid i, j \in \mathcal{L}\}$. A corresponds to allowed movements of drivers from node i to node j . For each driver k , the allowed movements are the following:

- From the starting depot s_k to a loading dock $l \in \mathcal{L}$ or to the ending depot e_k .
- From a loading dock $l \in \mathcal{L}$ to a delivery node $d \in \mathcal{D}$.
- From a delivery node $d \in \mathcal{D}$ to a loading dock $l \in \mathcal{L}$ or to the ending depot e_k .

For a customer c , arcs in A^D link consecutive delivery nodes of the same order $\{(i, j) \in \mathcal{D}_o, o \in \mathcal{O}_c, i < j\}$, and pair of delivery nodes of two different orders $\{(i, d_{o_2}^0), i \in \mathcal{D}_{o_1}, i \geq n_{o_1}^{min}, o_1, o_2 \in \mathcal{O}_c, o_1 \neq o_2\}$. Arcs in A^L link all pairs of loading docks of the same batch plant.

We define $\delta^+(i) = \{(i, j, k) \in A\}$ and $\delta^-(i) = \{(j, i, k) \in A\}$ as the outgoing and incoming arc sets of node $i \in V$; $\delta_D^+(i) = \{(i, j) \in A^D\}$ and $\delta_D^-(i) = \{(j, i) \in A^D\}$ are the

272 outcoming and incoming arc sets of delivery node $i \in \mathcal{D}$. Similarly, $\delta_L^+(i) = \{(i, j) \in A^L\}$
 273 and $\delta_L^-(i) = \{(j, i) \in A^L\}$ are the outcoming and incoming arc sets of loading node $i \in \mathcal{L}$.

274 Let the binary variable x_{ij}^k be 1 if driver k travels from node i to j . Binary variable
 275 y_o is 1 when order o is completely served; v_i and w_i are the start and end of the loading
 276 (unloading) operation at node $i \in \mathcal{L} \cup \mathcal{D}$. Binary variable u_{ij} is 1 if node j is served just after
 277 i , the service being either an unloading or a loading operation. Variable q_j^k is the quantity to
 278 be loaded towards j with vehicle k . Let w_k^1 be a continuous variable indicating the difference
 279 between the driver's work time and the minimum number of hours to be worked in a day,
 280 and w_k^2 indicating the difference between the driver's work time and the normal work time.
 281 Let g_i be the time between due date and first service start for customer i .

282 The objective function minimizes total travel cost (TC), penalties associated with unful-
 283 filled orders (PUO), first delivery delays (FDD), driver underutilization costs (DUC), and
 284 driver overtime costs (DOC). Together, these components drive the optimization process to
 285 find a solution that efficiently balances travel costs, customer satisfaction, on-time delivery,
 286 driver utilization, and scheduling constraints.

$$\min \sum_{(i,j,k) \in A} t_{ij} x_{ij}^k + \beta_1 \sum_{o \in \mathcal{O}} (1 - y_o) + \beta_2 \sum_{i \in \mathcal{C}} g_i + \sum_{k \in K} (\beta_3 w_k^1 + \beta_4 w_k^2) \quad (2)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^k = 1 \quad \forall i \in S, k \in K \quad (3)$$

$$\sum_{j \in \delta^-(i)} x_{ji}^k = 1 \quad \forall i \in E, k \in K \quad (4)$$

$$v_j \geq w_i - M(1 - x_{ij}^k) \quad i \in S, j \in \delta^+(i), k \in K \quad (5)$$

$$v_j \geq w_i + \alpha_b + t_{ij} - M(1 - x_{ij}^k) \quad \forall b \in \mathcal{B}, i \in \mathcal{L}_b, j \in \delta^+(i), k \in K \quad (6)$$

$$v_j \geq w_i + \rho + t_{ij} - M(1 - x_{ij}^k) \quad \forall i \in \mathcal{D}, j \in \delta^+(i), k \in K \quad (7)$$

$$w_i \geq v_i + \frac{q_j^k}{\tau_b^l} - M(1 - x_{ij}^k) \quad \forall b \in \mathcal{B}, i \in \mathcal{L}_b, j \in \delta^+(i), k \in K \quad (8)$$

$$w_j \geq v_j + \frac{q_j^k}{\tau_c^u} - M(1 - x_{ij}^k) \quad \forall c \in \mathcal{C}, j \in \mathcal{D}_c, i \in \delta^-(j), k \in K \quad (9)$$

$$w_j \leq v_i + \Delta + M(1 - x_{ij}^k) \quad j \in \mathcal{D}, i \in \delta^-(j), k \in K \quad (10)$$

$$v_{d_o^0} \geq a_i \quad \forall i \in \mathcal{C}, \forall o \in \mathcal{O}_i \quad (11)$$

$$g_i \geq v_{d_{o_1}^0} - a_i - M \left(\sum_{j \in \delta_D^-(d_{o_1}^0)} u_{jd_{o_1}^0} \right) \quad \forall i \in \mathcal{C}, \forall o_1 \in \mathcal{O}_i \quad (12)$$

$$v_{d_{o_1}^0} \geq w_j - M(1 - u_{jd_{o_1}^0}) \quad \forall o_1 \in \mathcal{O}_i, j \in \delta_D^-(d_{o_1}^0) \quad (13)$$

$$v_{d_{o_1}^0} \leq w_j + \gamma^2 + M(1 - u_{jd_{o_1}^0}) \quad \forall o_1 \in \mathcal{O}, j \in \delta_D^-(d_{o_1}^0) \quad (14)$$

$$\sum_{o_1 \in \mathcal{O}_i} \sum_{j \in \delta_D^-(d_{o_1}^0)} u_{jd_{o_1}^0} = |\mathcal{O}_i| - 1 \quad \forall i \in \mathcal{C}, |\mathcal{O}_i| > 1 \quad (15)$$

$$\sum_{o_1 \in \mathcal{O}_i} \sum_{j \in \delta_D^+(d_{o_1}^0)} u_{d_{o_1}^0, j} = |\mathcal{O}_i| - 1 \quad \forall i \in \mathcal{C}, |\mathcal{O}_i| > 1 \quad (16)$$

$$\sum_{j \in \delta_D^+(d_{o_1}^0)} u_{d_{o_1}^0, j} \leq 1 \quad \forall o_1 \in \mathcal{O} \quad (17)$$

$$\sum_{j \in \delta_D^-(d_{o_1}^0)} u_{j, d_{o_1}^0} \leq 1 \quad \forall o_1 \in \mathcal{O} \quad (18)$$

$$\sum_{j \in \delta_D^+(d_{o_1}^0)} u_{d_{o_1}^0, j} + \sum_{j \in \delta_D^-(d_{o_1}^0)} u_{j, d_{o_1}^0} \geq 1 \quad \forall o_1 \in \mathcal{O} \quad (19)$$

$$v_j \geq w_{j-1} - M(1 - u_{j-1, j}) \quad \forall o \in \mathcal{O}, j \in \mathcal{D}_o, j \geq 1 \quad (20)$$

$$v_j \leq w_{j-1} + \gamma^1 + M(1 - u_{j-1, j}) \quad \forall i \in \mathcal{C}, \forall o \in \mathcal{O}_i, j \in \mathcal{D}_o, j \geq 1 \quad (21)$$

$$u_{j-1, j} \geq u_{j, j+1} \quad \forall o_1 \in \mathcal{O}, j \in \mathcal{D}_{o_1}, 1 \leq j \leq n_{o_1} - 1 \quad (22)$$

$$u_{j-1, j} \geq \sum_{l \in \mathcal{L}} x_{lj} \quad \forall o_1 \in \mathcal{O}, j \in \mathcal{D}_{o_1}, j \geq 1 \quad (23)$$

$$v_j \geq w_i - M(1 - u_{i, j}) \quad \forall i \in \mathcal{L}, j \in \delta_L^+(i) \quad (24)$$

$$\sum_{j \in \delta_L^+(i)} u_{i, j} \leq 1 \quad \forall i \in \mathcal{L} \quad (25)$$

$$\sum_{j \in \delta_L^-(i)} u_{j, i} \leq 1 \quad \forall i \in \mathcal{L} \quad (26)$$

$$\sum_{j \in \delta_L^-(i)} u_{j, i} \geq \sum_{k \in K} \sum_{j \in \delta^-(i)} x_{ji}^k \quad \forall i \in \mathcal{L} \quad (27)$$

$$\sum_{j \in \delta_L^-(i)} u_{j, i} \geq \sum_{j \in \delta_L^-(i+1)} u_{j, i+1} \quad \forall i \in \mathcal{L} \quad (28)$$

$$\sum_{j \in \delta_L^-(i)} u_{j, i} + \sum_{j \in \delta_L^+(i)} u_{i, j} \geq \sum_{k \in K} \sum_{j \in \delta^-(i)} x_{ji}^k \quad \forall i \in \mathcal{L} \quad (29)$$

$$\sum_{k \in K} \sum_{j \in \mathcal{D}_o} q_j^k = q_o \quad \forall o \in \mathcal{O} \quad (30)$$

$$q_j^k \leq \sum_{i \in \delta^-(j)} Q^k x_{ij}^k \quad \forall j \in \mathcal{D}, k \in K \quad (31)$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k \leq 1 \quad i \in \mathcal{L} \cup \mathcal{D} \quad (32)$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k = \sum_{k \in K} \sum_{j \in \delta^-(i)} x_{ji}^k \quad i \in \mathcal{L} \cup \mathcal{D} \quad (33)$$

$$w_k^1 \geq M_T + H_k - v_{e_k} \quad \forall k \in K \quad (34)$$

$$w_k^2 \geq (v_{e_k} - H_k) - N_T \quad \forall k \in K \quad (35)$$

$$w_{e_k} \leq H_k + O_T \quad k \in K \quad (36)$$

$$0 \leq q_j^k \leq Q^k \quad j \in \mathcal{D}, k \in K \quad (37)$$

$$x_{ij}^k \in \{0, 1\} \quad (i, j) \in A, k \in K \quad (38)$$

$$u_{ij} \in \{0, 1\} \quad (i, j) \in A^D, k \in K \quad (39)$$

$$y_o \in \{0, 1\} \quad o \in \mathcal{O} \quad (40)$$

$$v_i \geq 0, w_i \geq 0 \quad i \in V \quad (41)$$

$$w_k^1 \geq 0, w_k^2 \geq 0 \quad i \in V \quad (42)$$

$$g_i \geq 0 \quad i \in \mathcal{C}. \quad (43)$$

287 β_1 to β_4 are the penalty coefficients of each component of the objective function (2).
 288 Constraints (3) and (4) state that driver k leaves his start node exactly once a day and
 289 returns to his end node. A driver cannot serve a node before the start of his shift with
 290 constraints (5). Constraints (6) and (7) set a driver to take some time after loading or
 291 unloading to adjust the concrete or clean the truck before moving on to the next node. The
 292 duration of the loading operation depends on the plant's loading rate and the amount q_j^k
 293 of RMC loaded (8). Similarly, the unloading service depends on the site's rate and q_j^k (9).
 294 Unloading operations must end at most Δ minutes after loading begins (10).

295 Constraints (11) - - (14) ensure that the first service of any customer i must start at the
 296 due time a_i . This first service may be performed at the first delivery node of any order $\in \mathcal{O}_i$,
 297 and one order must be completed before another is started. It also enforces the precedence
 298 constraints between the last delivery of an order and the first delivery of the following order.
 299 Constraints (15) - - (19) find the delivery sequence of all orders $\in \mathcal{O}_i$. With constraints (20)
 300 - - (23), two trucks cannot unload at the same time for consecutive deliveries of the same
 301 order. Additionally, these constraints impose a maximum time delay between the two trucks.
 302 Similarly, constraints (24) - - (29) ensure that two trucks cannot be loaded at the same time
 303 at a plant. Constraints (30) require that the cumulative load of all concrete mixers serving
 304 an order must equal the required quantities, and (31) bound a driver load. Constraints
 305 (32) require that a driver can only visit a loading/delivery node once. Constraints (33) are
 306 degree constraints. Constraints (34) - - (35) calculate the difference between a driver's hours
 307 of service and the minimum and normal hours of service. Finally, constraints (36) - - (43)
 308 define the nature and bounds of the variables.

309 Let us consider the solution of a small instance of our problem shown in Figures 1 and 2.
 310 Two batch plants B_1 and B_2 with three drivers serve two construction sites. Customer C_1
 311 requests one order of $q_1^1 = 12 \text{ m}^3$. Customer C_2 requests three orders of $q_2^1 = 3$, $q_2^2 = 11$, and
 312 $q_2^3 = 1 \text{ m}^3$ of three different types of concrete. B_1 has two drivers (D_1 , D_2) using concrete
 313 mixers of capacity 8 m^3 , and B_2 has D_3 with a capacity 12 m^3 . Figure 1 shows the flow of
 314 concrete mixers flow in the network. We schedule D_1 and D_2 to deliver 8 and 4 m^3 to C_1 ,
 315 respectively. We select o_2^2 as the first order of C_2 . D_3 travels to B_1 to load q_2^2 , then visits
 316 C_2 before returning to his home plant. Meanwhile, after their first trip, D_1 and D_2 deliver
 317 the remaining orders of C_2 .

318 Figure 2 shows the sequence of loading and unloading operations. The first deliveries of
 319 C_1 and C_2 are timely and there is no gap between the two deliveries received by C_1 . There
 320 is a delay between the end of the first delivery and the start of the second delivery of C_2 ,
 321 however, it is less than the maximal time delay of 20 minutes defined for this example.

322 4 Constructive heuristics and GRASP

323 In this section, we present the heuristic approach we implement to solve this variant of
 324 the Concrete Delivery Problem. Our method consists of constructing feasible solutions to
 325 the CDP with randomized greedy heuristics and iteratively invoking these heuristics in the
 326 GRASP metaheuristic. From now on, we represent the scheduling of a delivery node d as
 327 the pair (L_d^k, U_d^k) of loading and unloading tasks. $L_d^k = (b, q_d, v_b, w_b)$ indicates that driver
 328 k starts loading $q_d \text{ m}^3$ of RMC at plant b between $[v_b, w_b]$. $U_d^k = (k, v_d, w_d)$ indicates that
 329 driver k serves d between v_d and w_d . We then call a solution $S = \{\cup_{1 \leq j \leq |\mathcal{D}|} (L_j^k, U_j^k), k \in K\}$
 330 the set of all loading and unloading tasks performed on a given day.

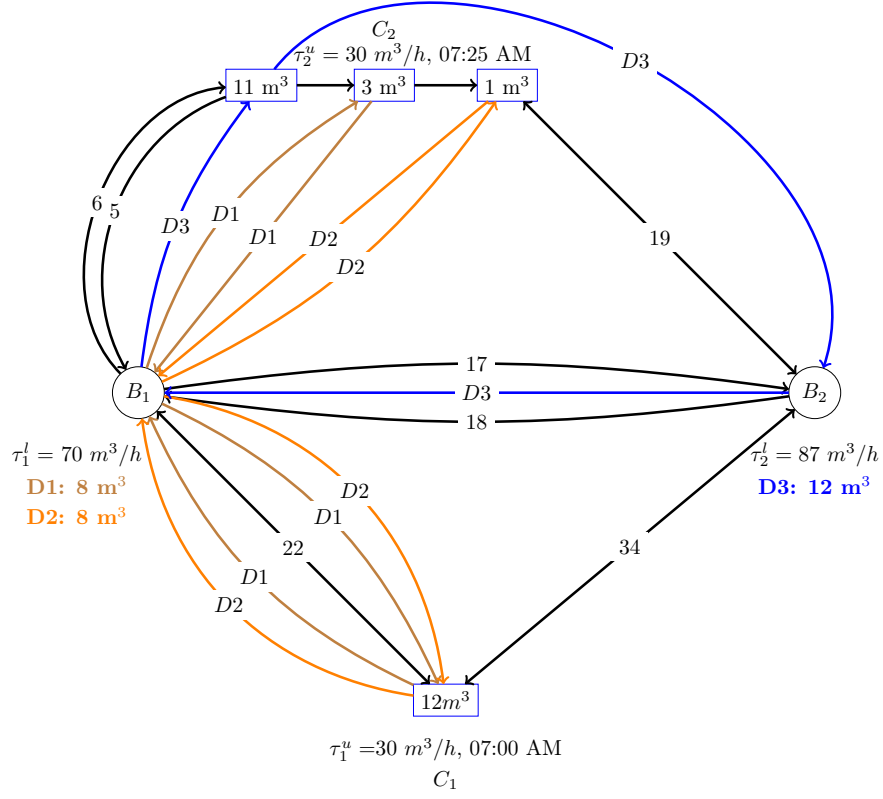


Figure 1: Solution of an instance with two plants, two construction sites, four orders, and three drivers.

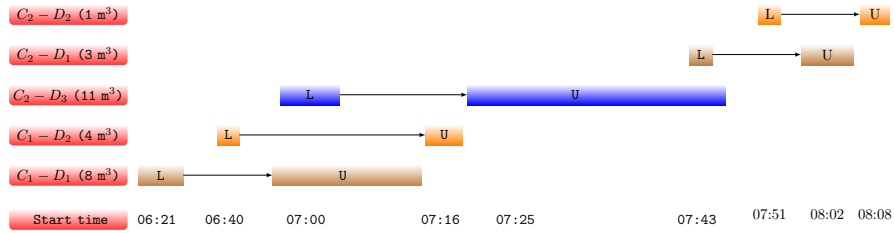


Figure 2: Schedule of the instance of Figure 1.

331 4.1 GRASP algorithm

332 The GRASP algorithm, proposed by Feo and Resende [1989], is a two-phase iterative suite
 333 consisting of constructive and local search algorithms. In the construction phase, a feasible
 334 solution S is generated using a greedy randomized algorithm. Then, a local search algorithm
 335 explores the neighborhood of S to identify a local optimum. Algorithm 1 outlines the
 336 pseudocode of the GRASP algorithm. To avoid redundant computations, a hash table H
 337 is used to ensure that the local search is not applied multiple times to the same solution.
 338 The procedure stops and returns the best overall solution when certain stopping conditions,
 339 such as reaching a time limit or a maximum number of iterations, are met.

Algorithm 1 Pseudo-code of the GRASP algorithm

Input: ϕ : number of iterations before path relinking

```

1  $S^* \leftarrow \emptyset$   $T \leftarrow \emptyset$   $Cost(S^*) \leftarrow \infty$ 
2 Initialize an empty hash table  $H$ 
3 while Conditions not met do
4    $S \leftarrow GreedyRandomizedHeuristic()$ 
5   if  $S \notin H$  then
6      $S \leftarrow LocalSearch(S)$ 
7     Add  $S$  to  $H$ 
8   if  $Cost(S) < Cost(S^*)$  then
9      $S^* \leftarrow S$ 
10 return  $S^*$ 

```

340 For the construction phase, we have developed two greedy randomized heuristics. The
 341 first heuristic, called customer-based insertion, schedules customers one at a time. On
 342 the other hand, the second heuristic, called delivery-based insertion, does not necessarily
 343 complete the scheduling of one customer before starting to plan another customer's schedule.

344 4.2 Customer-based insertion heuristic

345 The customer-based insertion heuristic builds a solution from scratch by scheduling all of a
 346 customer's orders one at a time. As shown in Algorithm 2, for each customer, we randomly
 347 select the first order to schedule. To schedule an order, we iterate over each of its delivery
 348 nodes, find a feasible pair of task, and add it to the current solution. For a delivery node
 349 d , we check if the list of operations $M[d]$ is empty. If it is, for each available driver k we
 350 simulate the loading and unloading operations (L_d^k, U_d^k) with the procedure *SimulateVisit*,
 351 and if k can serve d , we add k to a candidate list CL and (L_d^k, U_d^k) to $M[d]$.

352 The *SimulateVisit* procedure takes a partial solution, a delivery node d , a driver k and
 353 a numeric value λ as parameters. Its goal is to find the earliest time when an unloading
 354 service can start at d and the best position within the route of k . When planning to visit
 355 node d of order o and customer i with driver k , we first compute his expected delivery time
 356 v_d as follows:

$$357 \quad v_d = \begin{cases} a_i + \lambda & \text{for the first visited node} \\ w_j + rand(\lambda, \gamma^1), j \in \mathcal{D}_o, u_{jd} = 1 \\ w_j + rand(\lambda, \gamma^2), j \notin \mathcal{D}_o, u_{jd} = 1 \end{cases}$$

358 Then we determine the start loading time v_l of the corresponding dock node l by sub-
 359 tracting the travel time, adjustment time, and loading duration from v_d . Starting from v_l we
 360 check when k will be available at an idle loading bay and adjust w_l accordingly. Changing
 361 w_l can cause the cold-joint constraints to be violated, especially for a customer's subsequent

deliveries, and thus make the schedule infeasible. Once we have v_l , we can check for the best position to insert the visit of d into the route of k .

We extract from CL the restricted candidate list (RCL) of drivers for whom the cost of visiting d falls within a specified range based on the parameter θ . If RCL is empty and a visit to d cannot be scheduled due to cold-joint constraints, we use the *DelayPreviousDelivery* procedure to determine if there is a previous delivery of the same order that we can delay the start of service to make a visit to d feasible. This procedure returns the index of the node and λ , which is the value of the delay. λ is also the same parameter given to *SimulateVisit*. If there is such a previous node, we backtrack to it and restart the scheduling.

If RCL is not empty, we select a random driver from it, insert its corresponding load and unload tasks into S , and update the remaining quantities for the order and customer demands. The algorithm returns at the end the constructed solution S .

Algorithm 2 Customer-Based insertion algorithm

Input: S : empty solution S , K : vehicles set

Π : custom sorted customers list

```

1  $M \leftarrow \emptyset$  // List of pair of loading and unloading tasks for all delivery nodes
2  $CL \leftarrow \emptyset$  // Candidate driver list all delivery nodes
3 for each customer  $i \in \Pi$ 
4   Shuffle  $\mathcal{O}_i$ 
5   for each order  $o_1 \in \mathcal{O}_i$ 
6      $\lambda \leftarrow 0$ 
7     for  $j = 0$  to  $n_{o_1}^{max} - 1$  do
8       Delivery  $d = \mathcal{D}_{o_1}^j$ 
9       if  $M[d] == \emptyset$  then
10        for each  $k \in K$ 
11           $(L_d^k, U_d^k) \leftarrow \text{SimulateVisit}(S, d, k, \lambda)$ 
12           $Cost(d, k) \leftarrow \text{Cost of } (L_d^k, U_d^k)$ 
13          Add  $(L_d^k, U_d^k)$  to  $M[d]$ 
14           $CL[d] \leftarrow \{k\}$ 
15         $C_{min} \leftarrow \min \{C(x), x \in M[d]\}$ ,  $C_{max} \leftarrow \max \{C(x), x \in M[d]\}$ 
16         $RCL \leftarrow \{k \in CL[d], C(d, k) \leq C_{min} + \theta(C_{max} - C_{min})\}$ 
17        if  $RCL == \emptyset$  then
18           $(ind, delay) = \text{DelayPreviousDelivery}(S, d)$ 
19          if  $j$  is none then
20            break // continue with next customer
21           $j \leftarrow ind$ ,  $\lambda \leftarrow delay$ 
22          continue // backtrack to  $ind$  position
23        Select from  $RCL$  a random driver  $k$  to visit  $d$  with load  $q_d^k$ 
24         $M[d] \leftarrow M[d] \setminus (L_d^k, U_d^k)$ ,  $CL[d] \leftarrow CL[d] \setminus \{k\}$ 
25         $S \leftarrow S \cup (L_d^k, U_d^k)$ 
26         $q_i \leftarrow q_i - q_d^k$ ;  $q_{o_1} \leftarrow q_{o_1} - q_d^k$ 
27         $\lambda \leftarrow 0$ 
28        if  $q_{o_1} \neq 0$  then
29          continue
30        else if  $q_i \neq 0$  then
31          break
32 return  $S$ 

```

374 Consider the scenario where two customers require their services to start at the same time
 375 or within the same time window. In this case, we can either finish scheduling one customer
 376 before the other, or schedule both simultaneously. If there are sufficient resources available,
 377 either option would be feasible. However, when resources are limited, the customer-based
 378 insertion heuristic may leave some customers unscheduled. To address this limitation, we
 379 introduce the delivery-based insertion heuristic.

380 4.3 Delivery-based insertion heuristic

381 The delivery-based insertion heuristic constructs a feasible solution from scratch by schedul-
 382 ing a delivery node one at a time. The pseudocode for this heuristic is outlined in Algo-
 383 rithm 3. It starts with an empty solution S and initially fills CL with the first delivery of
 384 a randomly selected order for each customer. Within each iteration of the *while* loop, we
 385 filter CL to obtain a reduced candidate list CL' of deliveries from a given neighborhood.
 386 With this filtering, we can force the algorithm to schedule nodes with certain criteria first,
 387 which could be nodes from customers with the same demand, due date, or geographic area.
 388 We can also decide whether to schedule customers with the highest (or lowest) demand or
 389 the earliest (latest) due date first.

390 For each delivery d in CL' , the algorithm determines the best available driver k and adds
 391 the corresponding loading and unloading task (L_d^k, U_d^k) to a list of tasks. This is done using
 392 the *SimulateVisit* procedure described earlier. Next, the algorithm constructs a RCL from
 393 the list of tasks. If it is empty, the algorithm proceeds to the next iteration. Otherwise,
 394 it randomly selects a pair of tasks from the RCL , inserts them into S , and updates the
 395 remaining order and customer quantities. If an order o is not fulfilled after visiting d_o^j , we
 396 add its next delivery node d_o^{j+1} to CL . Otherwise, if the customer has remaining orders, we
 397 select the first delivery node $d_{o'}^0$ of another randomly selected order o' and add it to CL .

398 4.4 Local Search

399 After the construction phase, the local search phase iteratively invokes five operators using
 400 a first-improvement strategy. The first operator (*Swap Load*) exchanges loads between
 401 two deliveries of the same order. The *Swap Driver* operator explores alternative driver
 402 assignments by swapping drivers assigned to two delivery nodes. The *Relocate* operator
 403 removes a delivery node and places it in a different location. The *Consolidate Load* operator
 404 assigns an order request initially split between two drivers to a single driver, if the driver's
 405 capacity allows. The *Remove and Reschedule* operator selects an unscheduled customer,
 406 denoted i , within the current solution. It then identifies a set C containing scheduled
 407 customers with the same time slot as i . All elements of C are removed and rescheduled
 408 along with i . This local search process continues until no further improvements can be
 409 made or a timeout is reached.

410 5 Computational experiments

411 In this section, we present and discuss the results of computational experiments conducted
 412 using the GRASP heuristic described in section 4. The implementation of the GRASP
 413 algorithm is coded in C++. We run our experiments on the benchmark data used in
 414 Kinable et al. [2014], and on a dataset extracted from delivery operations records provided
 415 by our industry partner.

Algorithm 3 Delivery-based insertion algorithm

Input: S : empty solution S

```
1  $CL \leftarrow \emptyset$ 
2 for each customer  $i$ 
3   | Select a random order  $o_1 \in \mathcal{O}_i$ 
4   |  $CL \leftarrow CL \cup \{d_{o_1}^0\}$ 
5 while  $CL \neq \emptyset$  do
6   |  $CL' \leftarrow \text{Filter}(CL)$ 
7   |  $Tasks \leftarrow \emptyset$  // List of loading and unloading tasks
8   | for each  $d \in CL'$ 
9     |  $Tasks \leftarrow (L_d^k, U_d^k)$  //  $k$  is the best available driver to visit  $d$ 
10  |  $C_{min} \leftarrow \min \{Cost(x), x \in Tasks\}$ ,  $C_{max} \leftarrow \max \{Cost(x), x \in Tasks\}$ 
11  |  $RCL \leftarrow \{x \in Tasks, Cost(x) \leq C_{min} + \theta(C_{max} - C_{min})\}$ 
12  | if  $RCL \leftarrow \emptyset$  then
13    | continue
14  | Select random operations  $(L_{d_o^j}^k, U_{d_o^j}^k)$  of customer  $i$  (order  $o$ ) from  $RCL$ 
15  |  $S \leftarrow S \cup (L_{d_o^j}^k, U_{d_o^j}^k)$ 
16  |  $q_i \leftarrow q_i - q_{d_o^j}^k$ ;  $q_o \leftarrow q_o - q_{d_o^j}^k$ 
17  | if  $q_o \neq 0$  then
18    |  $CL \leftarrow CL \cup \{d_o^{j+1}\}$ 
19  | else if  $q_i \neq 0$  then
20    | Select another order  $o' \in \mathcal{O}_i$ 
21    |  $CL \leftarrow CL \cup \{d_{o'}^0\}$ 
22  | Update  $CL$ 
23 return  $S$ 
```

5.1 Generation of instances

The dataset used in this section was obtained from our partner and contains records representing delivery operations from up to 8 production centers over a period of 36 days. We have created 36 instances, where each instance corresponds to a daily operation. An instance contains information such as each driver's capacity, associated batch plant, and shift start time. For each customer, we have the due time, the total demand, and the number of orders (type of concrete) received. For each order, we have its demand and associated production plant. Finally, we have the loading capacity for each plant. The name of an instance has the format $C_k_n_o_z$, where k is the number of available drivers, n is the number of customers, o is the total number of daily orders, and z is the number of plants. The dataset is divided into small, medium, and large sets according to the total number of daily orders. We have summarized the dataset in Table 1.

Table 1: Instances summary

	#	Demand (m ³)	#Order	#Client	#Driver	#Depot
Small	8	226.5 - 937.5	7 - 14	4 - 11	13-35	1-3
Medium	14	1,160.5 - 2,971.0	43 - 98	40 - 89	76-137	6-8
Large	14	3,078. - 3,953.5	107 - 136	92 - 127	129-150	8

5.1.1 Results for the instances of Kinable et al. [2014]

Kinable et al. [2014] provided a benchmark dataset for their variant of the *CDP*. Although their primary objective is to maximize the total load delivered each day, without considering loading operations at a plant and ensuring deliveries within specified time windows, we still use this dataset to analyze and compare the performance of our algorithm under different conditions. Table 2 shows the performance of the GRASP heuristic along with the other methods (CP, MIP, SD-heuristic, and Hybrid) used in Kinable et al. [2014]. For each method, the table shows the average execution time in milliseconds and the corresponding gap between the obtained solution and the upper bound provided by Kinable et al. [2014].

Table 2: Performance of the GRASP heuristic on the CDP benchmark instances

		CP		MIP		SD-heuristic		Hybrid		GRASP	
		Gap (%)	Time (ms)	Gap (%)	Time (ms)	Gap (%)	Time (ms)	Gap (%)	Time (ms)	Gap (%)	Time (ms)
Set A	Avg	4.2	197,012	7.3	13,405,798	9.1	23	7.0	100,765	5.7	13,186
	Opt	40/64		37/64		30/64		35/64		35/64	
Set B	Avg	12.1	357,313	-	-	16.3	1,331	-	-	13.8	574,983
	Opt	55/128		-		40/128		-		44/128	

Overall, the GRASP algorithm shows promise in providing near-optimal solutions in a relatively short time for this benchmark of the CDP, making it a suitable candidate for further analysis and real-world application. It outperformed all methods except CP, achieving an average distance of 5.7% (13.8%) with a runtime of 13,186 (574,983) milliseconds for Set A (Set B). The SD-heuristic appears to be much faster, but GRASP provides a better balance between solution quality and computation time. It should also be noted that the exact MIP, CP, and Hybrid are initialized with the results of the SD-heuristic.

5.1.2 Results for the generated instances

The computational results of our instances solved with the GRASP heuristic are summarized in the tables below. The parameters used in our experiments are listed in Table 3.

Table 3: Parameters of the real-world instances

Parameter	Value
Unloading duration (τ_c^u)	2 min/ m^3
Cleaning duration (ρ)	10 min
Adjustment duration (α)	10 min
Max travel time (Δ)	120 min
Min working time (M_T)	180 min
Max working time (N_T)	480 min
Max overtime duration (O_T)	120 min
Max delay between two consecutive deliveries of the same order (γ^1)	20 min
Max delay between two consecutive deliveries of different orders (γ^2)	25 min
Penalty of unfulfilled orders (β_1)	10,000
Penalty of first delivery delay (β_2)	1,000
Penalty for driver underutilization (β_3)	30
Driver overtime penalty (β_4)	20

The table 4 contains the detailed results for all instances with a stop criterion of 3600 seconds. The table shows the demand of each instance, the average and the best values of

the objective function when solving each instance five times. All daily orders are served by our algorithm within these five replications. On average, five instances require multiple runs of the algorithm to successfully serve all customers, as shown in column UQ , which is the undelivered quantity. The sum of the driver underutilization costs (DUC) is high for medium and large instances. This indicates the underutilization of the scheduled fleet. Some drivers are not scheduled at all or work very little. The sum of driver overtime costs (DOC) is also high for these instances. This can be explained by the fact that the algorithm prefers to put certain drivers on overtime rather than use a driver who has to travel to another plant for a delivery, which would increase the travel costs (TC).

FDD is the sum of the delays of the first deliveries. For the small instances, all but one of the first deliveries are on time. For the other instances, there is at least one customer with a late first delivery. To get an idea of the actual delay per customer, we report $mFDD$, which is the maximum delay of all first deliveries. This tells us that we have a maximum delay that varies between 0 and 19 minutes for small instances, 3.5 and 60 minutes for medium instances, and 24 to 60 minutes for large instances.

Table 4: Results with all instances

		Demand (m^3)	Avg						Best					
			UQ (m^3)	TC (min)	FDD (min)	DUC (min)	DOC (min)	Z	TC (min)	FDD (min)	$mFDD$ (min)	DUC (min)	DOC (min)	Z
Small	C_13_11_12_1	226.5	0.0	1,243.5	0.0	37.7	0.0	2,373.9	1,243.5	0.0	0.0	0.0	0.0	1,243.5
	C_13_5_8_1	267.0	0.0	745.1	0.0	0.0	0.0	745.1	745.1	0.0	0.0	0.0	0.0	745.1
	C_18_6_11_2	333.5	0.0	753.3	0.0	194.7	0.0	6,593.7	753.3	0.0	0.0	125.3	0.0	4,513.2
	C_15_4_7_2	375.0	0.0	1,014.9	0.0	132.8	291.6	10,830.8	1,014.2	0.0	0.0	172.1	76.1	7,700.1
	C_19_7_8_2	388.0	0.0	1,765.5	0.0	314.8	150.9	14,225.6	1,745.8	0.0	0.0	224.5	87.9	10,237.1
	C_29_10_14_3	613.5	0.0	3,742.0	48.6	211.0	735.6	73,392.7	3,628.8	49.2	18.8	112.9	547.2	67,145.2
	C_31_8_11_3	776.0	0.0	2,330.6	0.0	781.6	656.5	38,909.6	2,297.2	0.0	0.0	606.7	567.0	31,836.1
	C_35_9_10_3	937.5	0.0	4,175.3	0.0	341.8	1,329.5	41,021.0	4,174.4	0.0	0.0	221.7	1,161.6	34,057.0
Medium	C_76_40_43_6	1,160.5	0.0	6,311.6	17.2	3,014.6	706.0	128,106.4	5,967.0	6.2	3.5	2,961.6	644.4	113,884.7
	C_104_42_47_8	1,565.0	0.0	9,358.2	34.1	3,890.1	679.0	173,754.5	9,777.5	32.0	13.0	2,830.0	240.6	131,507.6
	C_94_63_70_7	1,746.5	0.0	14,369.1	90.2	1,353.9	1,391.4	173,007.0	14,428.7	78.9	19.3	1,147.9	1,464.3	157,038.1
	C_116_67_78_8	1,839.5	0.0	13,602.9	67.9	2,419.4	857.3	171,229.1	13,871.9	67.2	33.2	2,347.7	523.3	162,021.5
	C_116_71_82_8	2,060.0	0.0	12,678.2	158.6	1,991.3	1,125.8	253,575.1	12,089.7	147.7	27.2	1,966.0	922.7	237,183.6
	C_117_57_70_6	2,327.5	0.0	15,495.0	142.4	1,388.6	1,680.5	233,181.4	15,153.3	114.2	22.2	1,823.7	1,671.5	217,448.3
	C_137_79_83_7	2,425.0	0.0	18,960.4	375.8	2,124.7	1,857.9	495,614.5	18,360.3	335.1	34.2	2,552.7	1,794.2	465,977.3
	C_127_66_80_8	2,512.5	0.0	16,033.3	10.4	1,579.6	1,552.1	104,879.1	17,599.1	14.4	5.0	1,046.3	1,242.3	88,203.5
	C_128_68_74_7	2,595.0	0.0	19,774.4	361.6	944.6	2,616.7	461,998.2	18,728.8	191.0	42.5	1,418.8	2,841.2	309,116.8
	C_136_85_97_8	2,673.0	0.0	18,223.6	188.5	2,134.5	2,352.2	317,811.4	18,894.6	137.4	48.6	1,155.6	2,137.0	233,669.0
	C_128_78_85_8	2,685.5	0.0	18,463.9	305.8	1,277.3	3,157.0	425,759.3	18,041.6	192.4	35.7	1,374.3	3,234.2	316,314.1
	C_131_77_85_8	2,893.5	0.0	18,003.0	212.6	1,248.7	1,716.1	302,370.4	18,063.0	200.2	43.2	780.7	1,539.1	272,509.6
	C_137_89_97_7	2,939.5	0.0	23,293.0	636.0	849.9	3,523.9	755,280.5	22,038.1	438.3	58.6	849.8	3,461.1	555,029.1
	C_133_84_98_7	2,971.0	18.7	22,000.0	1,157.6	805.4	2,291.4	1,249,585.1	20,992.6	1,061.2	59.4	546.1	2,742.5	1,153,396.1
	C_132_98_109_8	3,078.5	0.0	23,469.1	247.4	548.7	4,162.3	370,622.5	23,425.8	160.9	29.4	493.2	4,426.1	287,683.7
	C_129_101_119_8	3,229.0	0.0	19,051.7	269.0	1,232.1	3,820.4	401,387.9	19,453.3	166.7	35.8	1,138.1	4,719.6	314,728.3
	C_141_114_136_8	3,350.5	8.7	21,505.6	1,127.2	1,422.7	2,339.2	1,238,202.0	21,863.2	943.9	55.4	1,177.0	2,513.4	1,051,386.9
Large	C_140_114_132_8	3,401.5	0.0	23,938.4	1,055.1	910.2	3,269.1	1,171,715.1	23,969.7	988.5	54.4	981.3	3,253.1	1,107,019.4
	C_143_101_123_8	3,437.5	0.0	25,711.0	352.2	903.0	4,085.1	486,755.0	26,284.9	335.5	43.5	566.1	3,913.0	457,007.0
	C_137_112_129_8	3,471.0	0.0	22,562.4	340.8	402.4	3,371.1	442,869.1	22,338.3	275.1	45.6	375.2	3,819.8	385,076.1
	C_142_114_129_8	3,499.5	0.0	22,478.3	801.0	1,294.8	4,168.2	945,633.4	22,387.6	654.6	55.3	1,090.4	4,472.6	799,184.0
	C_149_98_122_8	3,513.0	3.0	20,767.5	778.4	2,236.9	2,852.2	923,357.9	21,224.7	684.5	51.1	2,267.6	3,046.6	834,708.6
	C_139_98_108_8	3,541.0	0.0	21,613.4	661.0	852.3	3,716.2	782,546.8	22,043.2	522.3	51.6	482.8	3,648.5	631,819.8
	C_144_108_122_8	3,670.5	0.0	24,738.3	746.0	664.9	3,465.8	860,025.7	25,806.9	675.4	50.4	548.5	3,209.2	781,866.1
	C_142_92_107_8	3,684.5	0.0	24,553.3	298.6	644.2	4,423.6	431,001.0	23,586.4	182.6	24.0	803.0	4,396.5	318,243.2
	C_138_114_136_8	3,739.5	0.0	25,366.9	791.5	668.6	5,501.8	946,993.9	23,639.0	600.4	55.2	1,230.8	5,314.0	767,223.0
	C_150_127_136_8	3,946.5	0.0	25,785.2	613.8	629.1	4,686.4	752,189.6	27,332.9	507.3	57.1	250.3	5,295.7	648,057.4
	C_148_112_131_8	3,953.5	6.9	26,231.2	1,002.0	517.8	3,467.4	1,113,149.3	25,559.2	886.9	56.0	559.3	4,213.2	1,013,500.2

Table 5 shows the percentage of concrete delivered by varying the execution times of the algorithm. For small instances, all requests are delivered in less than a minute, while for medium and large instances, over 99% of requests are delivered. For these instances, it takes more than ten minutes to ensure complete delivery of all orders.

Figure 3 illustrates the Gantt diagrams showing the loading and unloading operations for the orders assigned to each plant in instance $C_{29_10_14_3}$. This instance has three plants, 29 drivers, and two customers with multiple orders. Customer 8 has two orders

Table 5: RMC delivered within different runtimes

		Runtime (min)				
		1	5	10	30	60
Small	%Load	100.00	100.00	100.00	100.0	100.0
Medium	%Load	99.71	99.93	99.86	100.0	100.0
Large	%Load	99.64	99.64	99.79	100.0	100.0

471 (o_8, o_9) assigned to plant 0, and customer 9 has four orders (o_{10} - - o_{13}) assigned to plant 2.
 472 All orders requirements are in Table 6.

Table 6: Informations of instance *C29_10_14_3*

Order	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Customer	0	1	2	3	4	5	6	7	8	8	9	9	9	9
Demand	3.0	6.0	6.5	9.5	10.0	15.0	17.0	118.0	1.0	210.5	1.0	3.0	19.0	194.0
Plant	1	1	1	1	1	1	1	2	0	0	2	2	2	2
TW	09:00	10:00	12:00	09:15	07:00	07:00	10:30	10:00	07:00	07:00	07:25	07:25	07:25	07:25

473 We can observe that the loading operations for each plant are non-overlapping and the
 474 unloading operations occur sequentially, one after the other, at the worksite. Plant 0 is in
 475 charge of both orders of customer 8. In figure 3a, order o_9 is served before o_8 . Each row on
 476 the figure represents a pair of loading and unloading task for a delivery. Figure 3c shows
 477 how the loading bay is alternately assigned to orders o_{13} and o_7 between 10 : 00 and 15 : 30
 478 a.m.

479 We report for the instance *C29_10_14_3* a *FDD* of 49.6 minutes. The gantt chart
 480 indicates that customers 4, 5 and 8 whom need their delivery at 07 : 00 a.m. have all a first
 481 late delivery. The reason for these delays is that no

482 We tried to reduce the maximum delay of the first delivery but found that some orders
 483 couldn't be fulfilled because of driver shifts and overtime constraints. We decided to re-
 484 move these constraints and rerun our computation. We found that for small instances, the
 485 maximum delay dropped to zero, and for medium and large instances, it was reduced by
 486 almost 50%. This suggests that when we input a good work schedule, our algorithm works
 487 effectively to ensure that all orders are filled and that delays for first deliveries and travel
 488 costs are minimized. In Table 7, we can see that the average travel time decreases for all
 489 instances except the large ones. That's because the goals of minimizing travel cost and not
 490 delaying the first delivery are sometimes in conflict: to ensure that the first delivery is not
 491 late, drivers assigned to one plant often have to travel to another. And since all plants are
 492 used in large cases, this can increase travel times.

493 5.2 Sensitivity analyses

494 5.2.1 Influence of γ^1

495 The influence of the parameter γ^1 on the solution quality is demonstrated in Figure 5. Tests
 496 were conducted by varying γ^1 values between 10 and 25 minutes, without any driver shifts
 497 or overtime constraints. In small instances, all customers are served regardless of the value

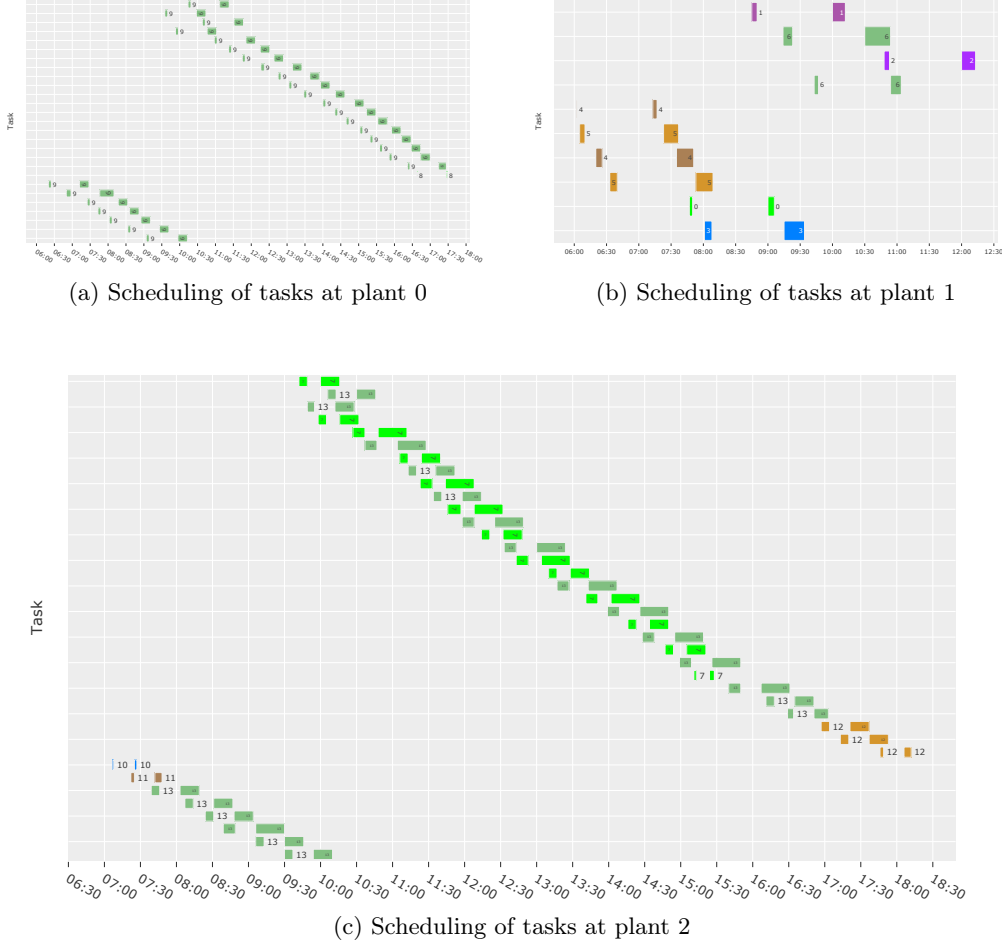


Figure 3: Loading and unloading schedules for the plants of instance $C29_10_14_3$

of γ^1 . For medium (large) instances, all customers are served with λ^1 starting at 15 (20) minutes. The delay of the initial delivery decreases as the value of γ^1 increases. The results indicate that if γ^1 is high, the plant has the flexibility to delay ongoing deliveries to make room for new ones or make customers wait until the drivers are available. By increasing the maximum delay between two successive deliveries from 10 to 20 minutes, the first delivery delay is reduced by over 50% for the medium and large size instances. The average travel time remains constant for all instances, regardless of the value of γ^1 .

5.2.2 Influence of the local search step

We checked how well the local search part of the GRASP algorithm works by running tests using only constructive heuristics. The results are presented in Table 9. For medium and large instances, the use of local search was necessary to service all customers within 60 minutes. And without local search, most objective metrics experienced a significant increase, except for travel costs.

Table 7: Comparaison with and without shift and overtime constraints

size	D							
	TC (min)	PUO (min)	FDD (min)	mFDD (min)	TC (min)	PUO (min)	FDD (min)	mFDD (min)
Small	1,971.27	0.00	6.08	2.68	1,842.47	0.00	0.00	0.00
Medium	15,922.87	1.34	228.67	35.27	14,771.08	0.00	83.84	19.04
Large	23,424.94	1.33	621.30	50.33	22,377.07	0.00	195.39	33.29

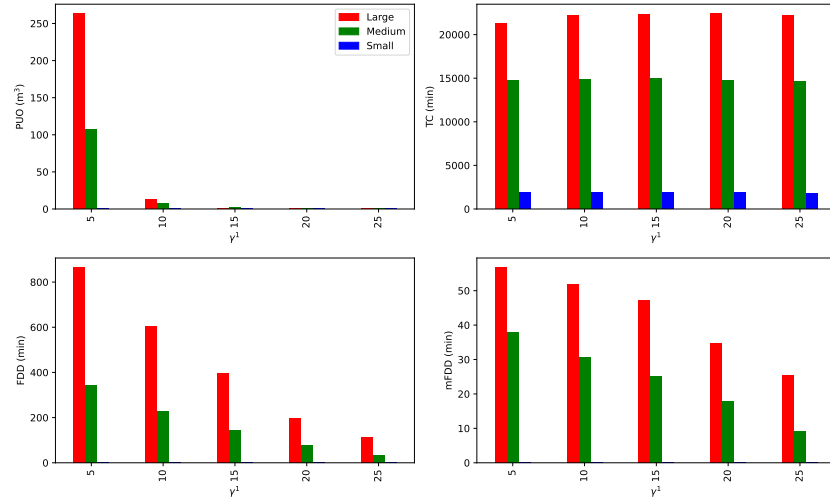
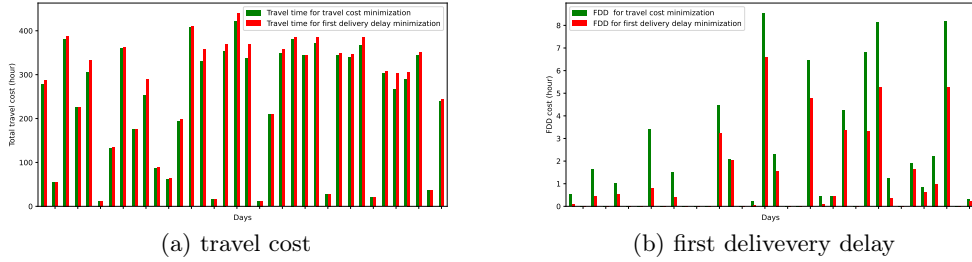


Figure 5: Influence of the maximum time delay between consecutive deliveries of the same order (γ^1)

5.2.3 Comparisons with the real operations

- resultats avec les instances (8 au total) où uniquement la flotte de Unibéton a été utilisée.
- enlever les contraintes de temps. reduire le gamma1
- (utiliser les mêmes pour la comparaison avec les données réelles) distance, demande servie, first, underwork, overtime.
- resultats avec les autres instances - distance, demande

Table 8: Influence of the local search step on the GRASP algorithm

size	Load (min)	Served_G (min)	TC (min)	FDD (min)	DUC (min)	DOC (min)	LS		Served_L (min)	FDD (min)
							Time (min)	TC (min)		
Small	3,917.00	100.00	2,039.77	8.04	225.15	221.71	24.22	1,965.93	100.00	5.94
Medium	32,394.00	99.82	14,118.33	358.96	2,242.93	1,328.91	58.32	15,896.96	100.00	239.98
Large	49,516.00	99.40	23,176.19	1,008.66	839.28	3,814.06	60.20	23,650.53	100.00	610.78
Avg	nan	nan	12,830.60	426.42	1,236.43	1,659.86	48.26	15,494.82	nan	321.84

Table 9: Influence of the local search step on the GRASP algorithm

size	GRASP\LS							GRASP						
	Load (m ³)	%Served	TC (min)	FDD (min)	DUC (min)	DOC (min)	Time (min)	%Served	TC (min)	FDD (min)	DUC (min)	DOC (min)	Time (min)	
Small	3917.0	100	1998.75	64.60	231.63	215.43	21.24	100	1946.06	1.55	229.40	298.74	50.34	
Medium	32394.0	99.6	13144.05	2528.33	2496.47	1086.12	49.41	100	13564.08	75.77	2250.66	1347.59	50.6	
Large	49516.0	99.2	21395.23	4885.11	1250.14	3543.16	50.20	100	21784.95	471.62	1168.73	3640.04	50.74	

516 servie, first delivery. - rouler les instances sans shift des livreurs. comparer la distance
517 parcourue, et demande livrée. - rouler sans les livreurs externes. regarder la demande servie
518 par jour.

519 6 Conclusion

520 Il serait intéressant de voir l'effet des paramètres lambda 1 et lambda 2 différents par clients
521 et /ou ordre. Également le temps de service différent par client.
522 l'algorithme serait intéressant à utiliser dans le cadre d'un

523 Acknowledgments

Table 10: Comparaison with partner operations

Instance	Demand (min)	D (min)	γ^1			
			('distanceCost', 5) (min)	('distanceCost', 10) (min)	('distanceCost', 15) (min)	('distanceC
C_13_5_8_1	267.00	543.05	523.46	523.46	523.46	
C_76_40_43_6	1,160.50	5,244.68	4,917.41	4,726.81	4,662.97	
C_116_67_78_8	1,839.50	11,664.77	12,220.90	11,084.60	11,019.80	1
C_15_4_7_2	375.00	789.28	702.88	702.88	703.46	
C_18_6_11_2	333.50	555.13	471.47	471.47	471.47	
C_94_63_70_7	1,746.50	12,011.22	12,776.30	12,201.20	11,895.40	1

524 Financial support for this work was provided by the Canadian Natural Sciences and
525 Engineering Research Council (NSERC) under grants 2015-04893 and 2019-00094. This
526 support is gratefully acknowledged.

527 References

- 528 C. Archetti and M. G. Speranza. The split delivery vehicle routing problem: A survey. In *The*
529 *vehicle routing problem: Latest advances and new challenges*, pages 103–122. Springer,
530 2008.
- 531 L. Asbach, U. Dorndorf, and E. Pesch. Analysis, modeling and solution of the concrete
532 delivery problem. *European Journal of Operational Research*, 193(3):820–835, 2009.
- 533 J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, M. Sterna, and J. Weglarz. *Handbook on*
534 *scheduling: From theory to practice*. Springer, 2019.
- 535 M. Durbin and K. Hoffman. Or practice—the dance of the thirty-ton trucks: Dispatching
536 and scheduling in a dynamic environment. *Operations Research*, 56(1):3–19, 2008.
- 537 J. M Faria, C. A. Silva, J. MC Sousa, M. Surico, and U. Kaymak. Distributed optimiza-
538 tion using ant colony optimization in a concrete delivery supply chain. In *2006 IEEE*
539 *International Conference on Evolutionary Computation*, pages 73–80. IEEE, 2006.
- 540 C.-W. Feng and H.-T. Wu. Integrating fmGA and CYCLONE to optimize the schedule of
541 dispatching RMC trucks. *Automation in Construction*, 15(2):186–199, 2006.
- 542 C.-W. Feng, T.-M. Cheng, and H.-T. Wu. Optimizing the schedule of dispatching rmc trucks
543 through genetic algorithms. *Automation in Construction*, 13(3):327–340, 2004.
- 544 T. A. Feo and M. GC Resende. A probabilistic heuristic for a computationally difficult set
545 covering problem. *Operations research letters*, 8(2):67–71, 1989.
- 546 M. Galić and I. Kraus. Simulation model for scenario optimization of the ready-mix concrete
547 delivery problem. *Selected Scientific Papers-Journal of Civil Engineering*, 11(2):7–18,
548 2016.
- 549 J. Garza Cavazos. *Dynamic planning and real-time monitoring of ready-mixed concrete*
550 *delivery problem*. PhD thesis, Universidad Autónoma de Nuevo León, 2021.
- 551 L. D. Graham, D. R. Forbes, and S. D. Smith. Modeling the ready mixed concrete delivery
552 system with neural networks. *Automation in construction*, 15(5):656–663, 2006.
- 553 O. A. Hernández López. *Study of mixed integer programming models for the concrete delivery*
554 *problem*. PhD thesis, Universidad Autónoma de Nuevo León, 2020.
- 555 J. Kinable and M. Trick. A logic based benders’ approach to the concrete delivery prob-
556 lem. In *International Conference on Integration of Constraint Programming, Artificial*
557 *Intelligence, and Operations Research*, pages 176–192. Springer, 2014.
- 558 J. Kinable, T. Wauters, and G. V. Berghé. The concrete delivery problem. *Computers &*
559 *Operations Research*, 48:53–68, 2014.
- 560 M. Lu and H.-C. Lam. Optimized concrete delivery scheduling using combined simulation
561 and genetic algorithms. In *Proceedings of the Winter Simulation Conference*. IEEE, 2005.

- 562 M. Maghrebi and S. T. Waller. Exploring experts decisions in concrete delivery dispatching
563 systems using bayesian network learning techniques. In *2014 2nd International Conference*
564 *on Artificial Intelligence, Modelling and Simulation*, pages 103–108. IEEE, 2014.
- 565 M. Maghrebi, V. Periaraj, S. T. Waller, and C. Sammut. Using benders decomposition for
566 solving ready mixed concrete dispatching problems. In *The 31st International Symposium*
567 *on Automation and Robotics in Construction and Mining*, 2014a.
- 568 M. Maghrebi, V. Periaraj, S. T. Waller, and Claude Sammut. Solving ready-mixed concrete
569 delivery problems: Evolutionary comparison between column generation and robust ge-
570 netic algorithm. In *Computing in Civil and Building Engineering (2014)*, pages 1417–1424.
571 2014b.
- 572 M. Maghrebi, V. Periaraj, S. T. Waller, and C. Sammut. Column generation-based approach
573 for solving large-scale ready mixed concrete delivery dispatching problems. *Computer-*
574 *Aided Civil and Infrastructure Engineering*, 31(2):145–159, 2016a.
- 575 M. Maghrebi, S. Travis Waller, and Claude Sammut. Sequential meta-heuristic approach
576 for solving large-scale ready-mixed concrete–dispatching problems. *Journal of Computing*
577 *in Civil Engineering*, 30(1):04014117, 2016b.
- 578 M. Maghrebi, T. Waller, and C. Sammut. Matching experts’ decisions in concrete deliv-
579 ery dispatching centers by ensemble learning algorithms: Tactical level. *Automation in*
580 *Construction*, 68:146–155, 2016c.
- 581 N. F. Matsatsinis. Towards a decision support system for the ready concrete distribution
582 system: A case of a greek company. *European Journal of Operational Research*, 152(2):
583 487–499, 2004.
- 584 N. Mayteekrieangkrai and W. Wongthatsanekorn. Optimized ready mixed concrete truck
585 scheduling for uncertain factors using bee algorithm. *Songklanakarin Journal of Science*
586 *& Technology*, 37(2), 2015.
- 587 M. Misir, W. Vancroonenburg, K. Verbeeck, and G. V. Berghe. A selection hyper-heuristic
588 for scheduling deliveries of ready-mixed concrete. In *Proceedings of the metaheuristics*
589 *international conference (MIC 2011)*, pages 289–298. Udine, Italy, 2011.
- 590 F. Muresan. Comparing ready-mix concrete and site-mixed
591 concrete, 2019. URL [https://www.ny-engineers.com/blog/](https://www.ny-engineers.com/blog/ready-mix-concrete-and-site-mixed-concrete)
592 [ready-mix-concrete-and-site-mixed-concrete](https://www.ny-engineers.com/blog/ready-mix-concrete-and-site-mixed-concrete).
- 593 P. K. Narayanan, D. Rey, M. Maghrebi, and S. T. Waller. Using lagrangian relaxation to
594 solve ready mixed concrete dispatching problems. *Transportation Research Record*, 2498
595 (1):84–90, 2015.
- 596 D. Naso, M. Surico, B. Turchiano, and U. Kaymak. Genetic algorithms for supply-chain
597 scheduling: A case study in the distribution of ready-mixed concrete. *European Journal*
598 *of Operational Research*, 177(3):2069–2099, 2007.
- 599 A. Panas and J.-P. Pantouvakis. Simulation-based Concrete Truck-Mixers Fleet Size Deter-
600 mination for On-Site Batch Plant Operation. *Procedia - Social and Behavioral Sciences*,
601 74:459–467, 2013.
- 602 F. Payr and V. Schmid. Optimizing deliveries of ready-mixed concrete. In *2009 2nd Inter-*
603 *national Symposium on Logistics and Industrial Informatics*, pages 1–6, 2009.

- 604 V. Schmid, K. F. Doerner, M. W.P. Hartl, R. F. and Savelsbergh, and W. Stoecher. A
605 hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43
606 (1):70–85, 2009.
- 607 V. Schmid, K F. Doerner, R. F. Hartl, and J-J. Salazar-González. Hybridization of very
608 large neighborhood search for ready-mixed concrete delivery problems. *Computers &
609 Operations Research*, 37(3):559–574, 2010.
- 610 A. Sinha, N. Singh, G. Kumar, and S. Pal. Quality factors prioritization of ready-mix
611 concrete and site-mix concrete: A case study in indian context. In D. Singh, A. K.
612 Awasthi, I. Zelinka, and K. Deep, editors, *Proceedings of International Conference on
613 Scientific and Natural Computing*, Algorithms for Intelligent Systems, pages 179–187,
614 Singapore, 2021. Springer.
- 615 M. Sulaman, X. Cai, M. Mısıır, and Z. Fan. Simulated annealing with a time-slot heuristic
616 for ready-mix concrete delivery. In *Asia-Pacific Conference on Simulated Evolution and
617 Learning*, pages 39–50. Springer, 2017.
- 618 X. Tian, Y. Mohamed, and S. AbouRizk. Simulation-based aggregate planning of batch plant
619 operations. *Canadian Journal of Civil Engineering*, 37(10):1277–1288, 2010. Publisher:
620 NRC Research Press.
- 621 I.D. Tommelein and A. Li. Just-in-time concrete delivery: mapping alternatives for vertical
622 supply chain integration. In *Proceedings of the 7th Annual Conference of the International
623 Group for Lean Construction*, volume 7, pages 97–108, University of California, 1999.
624 Berkeley.
- 625 A. Tzanetos and M. Blondin. Systematic search and mapping review of the concrete delivery
626 problem (cdp): Formulations, objectives, and data. *Automation in Construction*, 145:
627 104631, 2023.
- 628 S. Q. Wang, C. L. Teo, and G. Ofori. Scheduling the truckmixer arrival for a ready mixed
629 concrete pour via simulation with @risk. *Journal of Construction Research*, 2(2):169–179,
630 2001.
- 631 S. Yan and W. Lai. An optimal scheduling model for ready mixed concrete supply with
632 overtime considerations. *Automation in Construction*, 16(6):734–744, 2007.
- 633 J. Yang, B. Yue, F. Feng, J. Shi, H. Zong, J. Ma, L. Shangguan, and S. Li. Concrete vehicle
634 scheduling based on immune genetic algorithm. *Mathematical Problems in Engineering*,
635 2022.
- 636 T. M. Zayed and D. Halpin. Simulation of concrete batch plant production. *Journal of
637 Construction Engineering and Management*, 127(2):132–141, 2001.