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 15, 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

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

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

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

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

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

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 $\mathcal{D}_o = \left(d_o^0, d_o^1, \cdots, d_o^{n_o}\right)$. The deliveries of customer i are the ordered set $\mathcal{D}_i = (\mathcal{D}_{o_1}, \mathcal{D}_{o_2}, \cdots, \mathcal{D}_{o_{|O_i|}})$, 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 outcoming 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Let us consider the solution of a small instance of our problem shown in Figures 1 and 2. Two batch plants B_1 and B_2 with three drivers serve two construction sites. Customer C_1 requests one order of $q_1^1 = 12$ m^3. Customer C_2 requests three orders of $q_2^1 = 3$, $q_2^2 = 11$, and $q_2^3 = 1$ m^3 of three different types of concrete. B_1 has two drivers (D_1, D_2) using concrete mixers of capacity 8 m^3, and B_2 has D_3 with a capacity 12 m^3. Figure 1 shows the flow of concrete mixers flow in the network. We schedule D_1 and D_2 to deliver 8 and 4 m^3 to C_1, 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 C_2 before returning to his home plant. Meanwhile, after their first trip, D_1 and D_2 deliver the remaining orders of C_2.

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

4 Constructive heuristics and GRASP

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

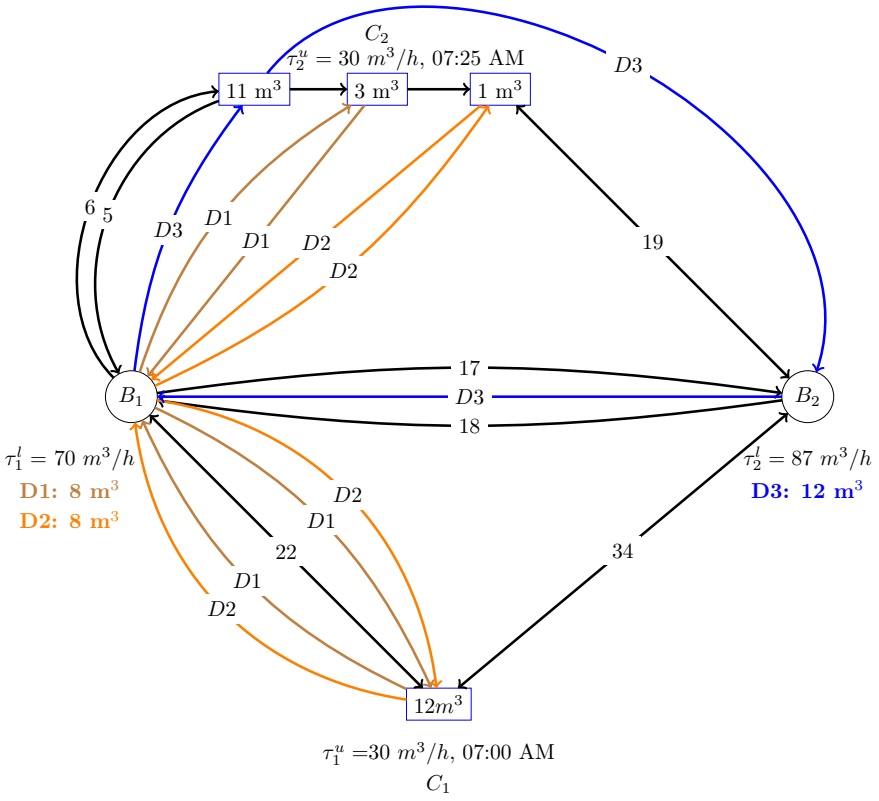

Figure 1: Solution of an instance with two plants, two construction sites, four orders, and three drivers.

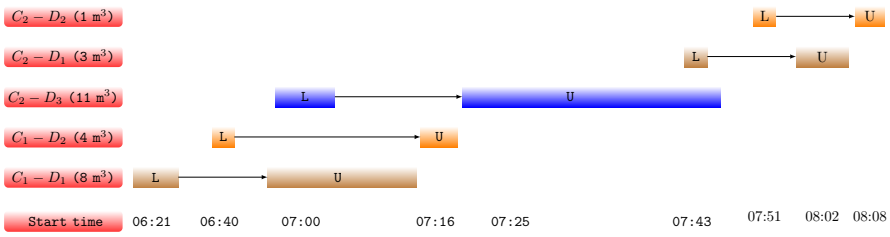

Figure 2: Schedule of the instance of Figure 1.

4.1 GRASP algorithm

The GRASP algorithm, proposed by Feo and Resende [1989], is a two-phase iterative suite consisting of constructive and local search algorithms. In the construction phase, a feasible solution S is generated using a greedy randomized algorithm. Then, a local search algorithm explores the neighborhood of S to identify a local optimum. Algorithm 1 outlines the pseudocode of the GRASP algorithm. To avoid redundant computations, a hash table H is used to ensure that the local search is not applied multiple times to the same solution. The procedure stops and returns the best overall solution when certain stopping conditions, such as reaching a time limit or a maximum number of iterations, are met. In our implementation, we also add a path relinking procedure that uses information about previous solutions to improve the search process.

Algorithm 1 Pseudo-code of the GRASP algorithm

Input: ϕ: number of iterations before path relinking

```
S* ← ∅    T ← ∅ Cost(S*) ← ∞
Initialize an empty hash table H
counter ← 0
while Conditions not met do
    S ← GreedyRandomizedHeuristic()
    if S ∉ H then
        S ← LocalSearch(S)
        Add S to H
    T ← T ∪ {S}
    if Cost(S) < Cost(S*) then
        S* ← S
    if counter mod φ == 0 then
        S* ← PathRelinking(T)
        T ← ∅
    counter ← counter + 1
return S*
```

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

4.2 Customer-based insertion heuristic

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

The $SimulateVisit$ procedure takes a partial solution, a delivery node d, a driver k and a numeric value λ as parameters. Its goal is to find the earliest time when an unloading service can start at d and the best position within the route of k. When planning to visit

node d of order o and customer i with driver k, we first compute his expected delivery time v_d as follows:

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

Then we determine the start loading time v_l of the corresponding dock node l by subtracting the travel time, adjustment time, and loading duration from v_d. Starting from v_l we check when k will be available at an idle loading bay and adjust w_l accordingly. Changing 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.

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

4.3 Delivery-based insertion heuristic

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

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

Algorithm 2 Customer-Based insertion algorithm

Input: S: empty solution S, K: vehicles set
Π: custom sorted customers list

$M \leftarrow \emptyset$ `// List of pair of loading and unloading tasks for all delivery nodes`
$CL \leftarrow \emptyset$ `// Candidate driver list all delivery nodes`
for each customer $i \in \Pi$
  Shuffle $\mathcal{O}_i$
  for each *order* $o_1 \in \mathcal{O}_i$
    $\lambda \leftarrow 0$
    for $j = 0$ **to** $n_{o_1}^{max} - 1$ **do**
      Delivery $d = \mathcal{D}_{o_1}^{j}$
      if $M[d] == \emptyset$ **then**
        for each $k \in K$
          $(L_d^k, U_d^k) \leftarrow$ SimulateVisit(S,d,k,λ)
          $Cost(d,k) \leftarrow$ Cost of (L_d^k, U_d^k)
          Add (L_d^k, U_d^k) to $M[d]$
          $CL[d] \leftarrow \{k\}$
      $C_{min} \leftarrow min\{C(x),\ x \in M[d]\}$, $C_{max} \leftarrow max\{C(x),\ x \in M[d]\}$
      $RCL \leftarrow \{k \in CL[d],\ C(d,k) \leq C_{min} + \theta(C_{max} - C_{min})\}$
      if $RCL == \emptyset$ **then**
        $(ind, delay) = DelayPreviousDelivery(S,d)$
        if j *is none* **then**
          break `// continue with next customer`
        $j \leftarrow ind$, $\lambda \leftarrow delay$
        continue `// backtrack to` *ind* `position`
      Select from RCL a random driver k to visit d with load q_d^k
      $M[d] \leftarrow M[d] \backslash (L_d^k, U_d^k)$, $CL[d] \leftarrow CL[d] \backslash \{k\}$
      $S \leftarrow S \cup (L_d^k, U_d^k)$
      $q_i \leftarrow q_i - q_d^k$; $q_{o_1} \leftarrow q_{o_1} - q_d^k$
      $\lambda \leftarrow 0$
      if $q_{o_1} \neq 0$ **then**
        continue
      else if $q_i \neq 0$ **then**
        break
return S

Algorithm 3 Delivery-based insertion algorithm

Input: S: empty solution S

$CL \leftarrow \emptyset$
for each customer i
  Select a random order $o_1 \in \mathcal{O}_i$
  $CL \leftarrow CL \cup \{d^0_{o_1}\}$
while $CL \neq \emptyset$ **do**
  $CL' \leftarrow$ Filter (CL)
  $Tasks \leftarrow \emptyset$ `// List of loading and unloading tasks`
  for each $d \in CL'$
    $Tasks \leftarrow (L^k_d, U^k_d)$ `//` k `is the best available driver to visit` d
  $C_{min} \leftarrow \min\{Cost(x),\ x \in Tasks\}$, $C_{max} \leftarrow \max\{Cost(x),\ x \in Tasks\}$
  $RCL \leftarrow \{x \in Tasks,\ Cost(x) \leq C_{min} + \theta(C_{max} - C_{min})\}$
  if $RCL \leftarrow \emptyset$ **then**
    continue
  Select random operations $(L^k_{d^j_o}, U^k_{d^j_o})$ of customer i (order o) from RCL
  $S \leftarrow S \cup (L^k_{d^j_o}, U^k_{d^j_o})$
  $q_i \leftarrow q_i - q^k_{d^j_o}$; $q_o \leftarrow q_o - q^k_{d^j_o}$
  if $q_o \neq 0$ **then**
    $CL \leftarrow CL \cup \{d^{j+1}_o\}$
  else if $q_i \neq 0$ **then**
    Select another order $o' \in \mathcal{O}_i$
    $CL \leftarrow CL \cup \{d^0_{o'}\}$
  Update CL
return S

4.4 Local Search

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

4.5 Path relinking

After the construction and local search steps, it is possible that some customers remain unscheduled in one solution while being accommodated in another. The goal of the path relinking procedure is to use each customer's scheduling information from existing solutions and combine them effectively to construct a solution that schedules all customers. This algorithm iterates through each element of a pool of solutions and attempts to insert each customer's schedule into a newly constructed solution.

5 Computational experiments

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

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 the 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	3,078.5-3,953.5	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].

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.

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	

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 stopping criterion is set to 20,000 iterations. 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

Table 4 shows the percentage of concrete delivered by varying the execution times of the algorithm. For small instances, all requests are delivered in less than five minutes, 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.

Table 4: RMC delivered within different runtimes

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

Table 5 shows the detailed results for each instance. The runtime is set to 3600 seconds. The table reports the requirements of each instance, the components of the objective function, the number of drivers used, and the execution time in seconds.

All daily orders are served with our algorithm, using less drivers than the available fleet. For the small size instances, all first deliveries are timely, except for one instance. The sum of the driver underutilization costs (DUC) is high for medium and large instances. This indicates underutilization of the scheduled fleet. Some drivers are not scheduled at

all or work very little. The sum of the 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 cost (TC).

FDD is the sum of the delays of the first deliveries. To get an idea of the actuel delay per customer, we report $mFDD$, which is the maximum delay for all first deliveries. This tells us that we have a maximum delay that varies between 0 and 12 minutes for small instances, 1 and 60 minutes for medium instances, and 50 to 60 minutes for large instances.

Table 5: Results with all instances

	Instance	Demand (m^3)	TC (min)	PUO	FDD (min)	$mFDD$ (min)	DUC (min)	DOC (min)	#Drivers	Time (s)
Small	C_13_11_12_1	226.5	1243.55	0.0	0.00	0.00	66.01	0.00	13	3022.06
	C_13_5_8_1	267.0	745.10	0.0	0.00	0.00	0.00	0.00	13	3027.39
	C_18_6_11_2	333.5	753.26	0.0	0.00	0.00	653.78	0.00	17	3004.51
	C_15_4_7_2	375.0	1013.11	0.0	0.00	0.00	56.79	244.61	15	3009.54
	C_19_7_8_2	388.0	1714.74	0.0	0.00	0.00	97.56	283.83	19	3020.89
	C_29_10_14_3	613.5	3724.29	0.0	12.41	12.41	246.04	454.51	28	3035.23
	C_31_8_11_3	776.0	2297.20	0.0	0.00	0.00	627.37	539.96	30	3033.85
	C_35_9_10_3	937.5	4077.20	0.0	0.00	0.00	720.00	866.98	31	3020.75
Medium	C_76_40_43_6	1160.5	5715.53	0.0	1.00	1.00	3566.49	771.23	69	3021.92
	C_104_42_47_8	1565.0	9114.62	0.0	23.00	16.24	4180.95	671.94	94	3047.99
	C_94_63_70_7	1746.5	13888.90	0.0	19.86	12.40	1086.60	1477.76	94	3025.72
	C_116_67_78_8	1839.5	13946.30	0.0	44.68	9.98	1578.90	639.28	116	3007.88
	C_116_71_82_8	2060.0	12461.20	0.0	68.38	20.06	3119.66	1478.63	107	3028.24
	C_117_57_70_6	2327.5	15066.80	0.0	83.79	16.44	1522.11	1892.64	117	3020.70
	C_137_79_83_7	2425.0	18811.40	0.0	252.94	41.65	2193.63	1963.05	136	3025.83
	C_127_66_80_8	2512.5	15177.80	0.0	2.00	2.00	1874.83	965.65	125	3007.70
	C_128_68_74_7	2595.0	19133.50	0.0	251.77	54.40	1523.19	2924.28	128	3053.96
	C_136_85_97_8	2673.0	16832.10	0.0	154.97	50.21	3507.40	1958.58	132	3028.15
	C_128_78_85_8	2685.5	16927.30	0.0	202.26	50.12	1492.50	2794.21	128	3030.32
	C_131_77_85_8	2893.5	17572.10	0.0	187.42	49.24	1803.22	1306.94	130	3040.74
	C_137_89_97_7	2939.5	21934.10	0.0	697.18	59.10	1220.15	3038.33	137	3025.45
	C_133_84_98_7	2971.0	22399.30	0.0	867.12	59.44	550.77	2887.51	133	3137.11
Large	C_132_98_109_8	3078.5	23102.40	0.0	193.01	49.44	450.57	3931.83	132	3018.05
	C_129_101_119_8	3229.0	18978.90	0.0	467.94	59.47	1084.78	2334.16	129	3179.81
	C_141_114_136_8	3350.5	21760.00	0.0	1131.39	59.58	1284.13	2360.19	141	3056.56
	C_140_114_132_8	3401.5	23259.00	0.0	1092.82	57.99	634.53	2705.26	140	3046.75
	C_143_101_123_8	3437.5	24560.70	0.0	451.36	56.61	1663.43	4359.64	142	3039.02
	C_137_112_129_8	3471.0	21526.10	0.0	252.62	58.09	624.87	3360.64	137	3018.76
	C_142_114_129_8	3499.5	21991.50	0.0	898.10	55.22	1750.60	4943.03	142	3011.54
	C_149_98_122_8	3513.0	20696.60	0.0	839.56	52.60	2465.87	2892.55	149	3072.58
	C_139_98_108_8	3541.0	20550.10	0.0	566.69	57.49	1438.11	2910.97	139	3011.84
	C_144_108_122_8	3670.5	23401.60	0.0	668.15	59.77	1438.60	3569.62	143	3056.51
	C_142_92_107_8	3684.5	22700.20	0.0	147.42	48.31	899.82	3702.01	142	3019.52
	C_138_114_136_8	3739.5	24670.70	0.0	750.04	55.88	518.63	5156.41	137	3021.26
	C_150_127_136_8	3946.5	25822.20	0.0	742.75	57.10	899.92	5101.58	150	3060.79
	C_148_112_131_8	3953.5	25455.30	0.0	868.81	49.15	500.68	3775.57	148	3005.96

We tried to reduce the maximum delay of the first delivery but found that some orders couldn't be fulfilled because of driver shifts and overtime constraints. We decided to remove these constraints and rerun our computation. We found that for small instances, the maximum delay dropped to zero, and for medium and large instances, it was reduced by almost 50%. This suggests that when we input a good work schedule, our algorithm works effectively to ensure that all orders are filled and that delays for first deliveries and travel costs are minimized. In Table 6, we can see that the average travel time decreases for all instances except the large ones. That's because the goals of minimizing travel cost and not

delaying the first delivery are sometimes in conflict: to ensure that the first delivery is not late, drivers assigned to one plant often have to travel to another. And since all plants are used in large cases, this can increase travel times.

Table 6: Comparaison with and without shift and overtime constraints

	Default parameters				$M_T = 0$, $N_T, O_T = \infty$, $H_k = 0$			
	Load (m^3)	TC (min)	FDD (min)	$mFDD$ (min)	Load (m^3)	TC (min)	FDD (min)	$mFDD$ (min)
Small	489.62	1946.06	1.55	1.55	489.62	1832.90	0.00	0.00
Medium	2313.86	15641.50	204.03	31.59	2313.86	14607.66	74.76	16.56
Large	3536.86	22748.24	647.90	55.48	3536.86	22293.04	210.14	29.00

The influence of the parameter γ^1 on the solution quality is demonstrated in Figure 3. Tests were conducted by varying γ^1 values between 10 and 25 minutes, without any driver shifts or overtime constraints. In small instances, all customers are served regardless of the value 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.

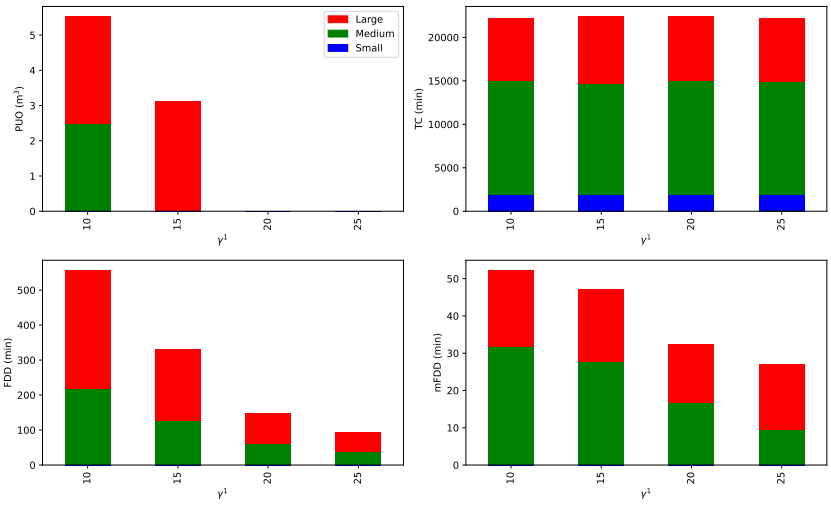

Figure 3: Influence of the maximum time delay between consecutive deliveries of the same order (γ^1)

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 7. 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: Influence of the local search step on the GRASP algorithm

	GRASP\LS							GRASP					
size	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

5.1.3 Comparisons with the real operations

- resultats avec les instances (8 au total) où uniquement la flotte de Unibéton a éét utilisée. enlver 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 servie, first delivery. - rouler les instances sans shift des livreurs. comparer la distance parcourue, et demande livrée. - rouler sans les livreurs externes. regarder la demande servie par jour.

6 Conclusion

Il serait intéressant de voir l'effet des paramètres lambda 1 et lambda 2 différents par clients et /ou ordre. Également le temps de service différent par client.

l'algorithme serait intéressant à utiliser dans le cadre d'un

Acknowledgments

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

References

C. Archetti and M. G. Speranza. The split delivery vehicle routing problem: A survey. In *The vehicle routing problem: Latest advances and new challenges*, pages 103–122. Springer, 2008.

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

J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, M. Sterna, and J. Weglarz. *Handbook on scheduling: From theory to practice*. Springer, 2019.

M. Durbin and K. Hoffman. Or practice—the dance of the thirty-ton trucks: Dispatching and scheduling in a dynamic environment. *Operations Research*, 56(1):3–19, 2008.

J. M Faria, C. A. Silva, J. MC Sousa, M. Surico, and U. Kaymak. Distributed optimization using ant colony optimization in a concrete delivery supply chain. In *2006 IEEE International Conference on Evolutionary Computation*, pages 73–80. IEEE, 2006.

C.-W. Feng and H.-T. Wu. Integrating fmGA and CYCLONE to optimize the schedule of dispatching RMC trucks. *Automation in Construction*, 15(2):186–199, 2006.

C.-W. Feng, T.-M. Cheng, and H.-T. Wu. Optimizing the schedule of dispatching rmc trucks through genetic algorithms. *Automation in Construction*, 13(3):327–340, 2004.

T. A. Feo and M. GC Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71, 1989.

M. Galić and I. Kraus. Simulation model for scenario optimization of the ready-mix concrete delivery problem. *Selected Scientific Papers-Journal of Civil Engineering*, 11(2):7–18, 2016.

J. Garza Cavazos. *Dynamic planning and real-time monitoring of ready-mixed concrete delivery problem*. PhD thesis, Universidad Autónoma de Nuevo León, 2021.

L. D. Graham, D. R. Forbes, and S. D. Smith. Modeling the ready mixed concrete delivery system with neural networks. *Automation in construction*, 15(5):656–663, 2006.

O. A. Hernández López. *Study of mixed integer programming models for the concrete delivery problem*. PhD thesis, Universidad Autónoma de Nuevo León, 2020.

J. Kinable and M. Trick. A logic based benders' approach to the concrete delivery problem. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 176–192. Springer, 2014.

J. Kinable, T. Wauters, and G. V. Berghe. The concrete delivery problem. *Computers & Operations Research*, 48:53–68, 2014.

M. Lu and H.-C. Lam. Optimized concrete delivery scheduling using combined simulation and genetic algorithms. In *Proceedings of the Winter Simulation Conference*. IEEE, 2005.

M. Maghrebi and S. T. Waller. Exploring experts decisions in concrete delivery dispatching systems using bayesian network learning techniques. In *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, pages 103–108. IEEE, 2014.

M. Maghrebi, V. Periaraj, S. T. Waller, and C. Sammut. Using benders decomposition for solving ready mixed concrete dispatching problems. In *The 31st International Symposium on Automation and Robotics in Construction and Mining*, 2014a.

M. Maghrebi, V. Periaraj, S. T. Waller, and Claude Sammut. Solving ready-mixed concrete delivery problems: Evolutionary comparison between column generation and robust genetic algorithm. In *Computing in Civil and Building Engineering (2014)*, pages 1417–1424. 2014b.

M. Maghrebi, V. Periaraj, S. T. Waller, and C. Sammut. Column generation-based approach for solving large-scale ready mixed concrete delivery dispatching problems. *Computer-Aided Civil and Infrastructure Engineering*, 31(2):145–159, 2016a.

M. Maghrebi, S. Travis Waller, and Claude Sammut. Sequential meta-heuristic approach for solving large-scale ready-mixed concrete–dispatching problems. *Journal of Computing in Civil Engineering*, 30(1):04014117, 2016b.

M. Maghrebi, T. Waller, and C. Sammut. Matching experts' decisions in concrete delivery dispatching centers by ensemble learning algorithms: Tactical level. *Automation in Construction*, 68:146–155, 2016c.

N. F. Matsatsinis. Towards a decision support system for the ready concrete distribution system: A case of a greek company. *European Journal of Operational Research*, 152(2): 487–499, 2004.

N. Mayteekrieangkrai and W. Wongthatsanekorn. Optimized ready mixed concrete truck scheduling for uncertain factors using bee algorithm. *Songklanakarin Journal of Science & Technology*, 37(2), 2015.

M. Misir, W. Vancroonenburg, K. Verbeeck, and G. V. Berghe. A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete. In *Proceedings of the metaheuristics international conference (MIC 2011)*, pages 289–298. Udine, Italy, 2011.

F. Muresan. Comparing ready-mix concrete and site-mixed concrete, 2019. URL `https://www.ny-engineers.com/blog/ready-mix-concrete-and-site-mixed-concrete`.

P. K. Narayanan, D. Rey, M. Maghrebi, and S. T. Waller. Using lagrangian relaxation to solve ready mixed concrete dispatching problems. *Transportation Research Record*, 2498 (1):84–90, 2015.

D. Naso, M. Surico, B. Turchiano, and U. Kaymak. 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.

A. Panas and J.-P. Pantouvakis. Simulation-based Concrete Truck-Mixers Fleet Size Determination for On-Site Batch Plant Operation. *Procedia - Social and Behavioral Sciences*, 74:459–467, 2013.

F. Payr and V. Schmid. Optimizing deliveries of ready-mixed concrete. In *2009 2nd International Symposium on Logistics and Industrial Informatics*, pages 1–6, 2009.

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

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

A. Sinha, N. Singh, G. Kumar, and S. Pal. Quality factors prioritization of ready-mix concrete and site-mix concrete: A case study in indian context. In D. Singh, A. K. Awasthi, I. Zelinka, and K. Deep, editors, *Proceedings of International Conference on Scientific and Natural Computing*, Algorithms for Intelligent Systems, pages 179–187, Singapore, 2021. Springer.

M. Sulaman, X. Cai, M. Mısır, and Z. Fan. Simulated annealing with a time-slot heuristic for ready-mix concrete delivery. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 39–50. Springer, 2017.

X. Tian, Y. Mohamed, and S. AbouRizk. Simulation-based aggregate planning of batch plant operations. *Canadian Journal of Civil Engineering*, 37(10):1277–1288, 2010. Publisher: NRC Research Press.

I.D. Tommelein and A. Li. Just-in-time concrete delivery: mapping alternatives for vertical supply chain integration. In *Proceedings of the 7th Annual Conference of the International Group for Lean Construction*, volume 7, pages 97–108, University of California, 1999. Berkeley.

A. Tzanetos and M. Blondin. Systematic search and mapping review of the concrete delivery problem (cdp): Formulations, objectives, and data. *Automation in Construction*, 145: 104631, 2023.

S. Q. Wang, C. L. Teo, and G. Ofori. Scheduling the truckmixer arrival for a ready mixed concrete pour via simulation with @risk. *Journal of Construction Research*, 2(2):169–179, 2001.

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

J. Yang, B. Yue, F. Feng, J. Shi, H. Zong, J. Ma, L. Shangguan, and S. Li. Concrete vehicle scheduling based on immune genetic algorithm. *Mathematical Problems in Engineering*, 2022.

T. M. Zayed and D. Halpin. Simulation of concrete batch plant production. *Journal of Construction Engineering and Management*, 127(2):132–141, 2001.