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

March 4, 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 and generated CDP 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 batching 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 solution 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 batching 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 batching 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 batching 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 batching 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 batching 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. Sulaman et al. [2017] expand upon the SD heuristic proposed in Kinable et al. [2014], proposing a simulated annealing (SA) combined with a time-slot Heuristic (SATH). This method looks for a slot between existing visits of a truck to schedule a new delivery instead of assigning it to the time slot strictly after the truck's latest assigned delivery. The goal is to reduce the large time gaps that can be present in a schedule created with SD due to ignoring the intermediate available time slots. Experimental results indicated that SATH outperforms SD in speed and solution quality. 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 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 central center and assigned to one of the company's batching 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 batching plants.

A customer i requests one or more types of concrete to be delivered to his construction site on a particular day, starting at time a_i. We call an order o a request for a specific type of concrete. q_i is the sum of the demand 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 of O_i must have its first delivery start at a_i, while the others can start at any time after o is completed. Once a plant is selected to produce a customer's first delivery, it must be the supplier of all subsequent deliveries from that customer. To avoid cold joint problems with the concrete, the subsequent deliveries must be made in close succession. We define a maximum time delay γ_i after which no more deliveries will be allowed. The customer 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 incurred if a driver works 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. To fill multiple orders, a driver must restock at a plant between deliveries. After unloading the RMC, a driver takes β_k 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. The lower bound is the number of deliveries required if we only use trucks with capacity Q_{max}, while the upper bound is the number of deliveries required if we only use trucks with capacity Q_{min}.

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

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 rth delivered order. We will refer to $d \in \mathcal{D}_i$ ($d \in \mathcal{D}_o$) as the dth potential delivery of customer i (order o). $\mathcal{D} = \bigcup_{i \in \mathcal{C}} \mathcal{D}_i$ is the union of all delivery nodes.

The company owns a total of eight batching plants located in various geographical regions. Each plant has a loading dock that can only accommodate only one truck at a time, which leads to trucks lining up. Let $\mathcal{B}$ be the set of batching 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 or hire external vehicles if necessary. A batching plant can serve a construction site as long as the travel time between the two is less than the concrete's lifespan, which is represented by Δ. Let $l_{b,j}$ be the loading dock node associated with delivery node j at plant b. We represent each plant b by the set $\mathcal{L}_b = \{l_{b,j}, t_{bj} \leq \Delta, j \in \mathcal{D}\}$ of loading docks nodes. $\mathcal{L} = \bigcup_{b \in B} \mathcal{L}_b$ is the union of all loading docks.

A solution to the problem involves making decisions about truck loading schedules, driver assignments to different deliveries, and truck arrival times at construction sites for unloading. For a batching plant, the decision involves choosing which driver to load, when to load them, and which construction site they should deliver to. For a driver, the decision involves determining 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 his 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 $\{(i, d^0_{o_2}), i \in \mathcal{D}_{o_1}, i \geq n^{min}_{o_1}, o_1, o_2 \in \mathcal{O}_c, o_1 \neq o_2\}$. Arcs in A^L link all pairs of loading docks of the same batching plant.

$\delta^+(i) = \{(i,j,k) \in A\}$, and $\delta^-(i) = \{(j,i,k) \in A\}$ are the outcoming and incoming arc sets of any 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 $i \in \mathcal{D}$.

Let the binary variable x^k_{ij} be 1 if the driver k travels from node i to j. The binary variable y_o is 1 when the order o is completely served. v_i and w_i are the start and end of the loading (unloading) operation at node $i \in \mathcal{L} \cup \mathcal{D}$. The binary variable $u_{i,j}$ is 1 if node j is served just after i, the service being either an unloading or a loading operation. The binary variable σ_{ib} is equal to 1 if the orders of customer i are loaded by plant b. Variable q^k_j is the quantity to be loaded towards j with vehicle k. Let w^1_k 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. w^2_k is a continuous variable 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 the travel cost, the penalty costs incurred when customer demands are not fully met, the lateness of each customer's first delivery, the cost of drivers working less than the minimum hours, and the total overtime cost of drivers working beyond their scheduled hours. The mathematical model of this variant of the CDP is as follows:

$$\min \sum_{(i,j,k) \in A} t_{ij} x^k_{ij} + \beta_1 \sum_{o \in \mathcal{O}} (1 - y_o) + \beta_2 \sum_{i \in \mathcal{C}} g_i + \sum_{k \in K} \beta_3 * w^1_k + \beta_4 * w^2_k \tag{2}$$

$$\sum_{j \in \delta^+(s_k)} x^k_{s_k j} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{j \in \delta^-(e_k)} x^k_{j e_k} = 1 \qquad \forall k \in K \tag{4}$$

$$v_j \geq w_i + \alpha_i + t_{ij} - M\left(1 - x^k_{ij}\right) \qquad \forall i \in \mathcal{L}, j \in \delta^+(i), k \in K \tag{5}$$

$$v_j \geq w_i + \beta_k + t_{ij} - M\left(1 - x^k_{ij}\right) \qquad \forall i \in \mathcal{D}, j \in \delta^+(i), k \in K \tag{6}$$

$$w_i \geq v_i + \frac{q^k_j}{\tau^l_b} - M\left(1 - x^k_{ij}\right) \qquad \forall b \in \mathcal{B}, i \in \mathcal{L}_b, j \in \delta^+(i), k \in K \tag{7}$$

$$w_j \geq v_j + \frac{q^k_j}{\tau^u_c} - M\left(1 - x^k_{ij}\right) \qquad \forall c \in \mathcal{C}, j \in \mathcal{D}_c, i \in \delta^-(j), k \in K \tag{8}$$

$$w_j \leq v_i + \Delta + M\left(1 - x^k_{ij}\right) \qquad j \in \mathcal{D}, i \in \delta^-(j), k \in K \tag{9}$$

$$v_{d^0_o} \geq a_i \qquad \forall i \in \mathcal{C}, \forall o \in \mathcal{O}_i \tag{10}$$

$$g_i \geq v_{d^0_{o_1}} - a_i - M\left(\sum_{j \in \delta^-_D(d^0_{o_1})} u_{j,d^0_{o_1}}\right) \qquad \forall i \in \mathcal{C}, \forall o_1 \in \mathcal{O}_i, d = d^0_{o_1} \tag{11}$$

$$v_{d^0_{o_1}} \geq w_j - M\left(1 - u_{j,d^0_{o_1}}\right) \qquad \forall o_1 \in \mathcal{O}_i, j \in \delta^-_D(d^0_{o_1}) \quad (12)$$

$$v_{d^0_{o_1}} \leq w_j + \gamma_i + M\left(1 - u_{j,d^0_{o_1}}\right) \qquad \forall o_1 \in \mathcal{O}, j \in \delta^-_D(d^0_{o_1}) \quad (13)$$

$$\sum_{o_1 \in \mathcal{O}_i} \sum_{j \in \delta^-_D(d^0_{o_1})} u_{j,d^0_{o_1}} = |\mathcal{O}_i| - 1 \qquad \forall i \in \mathcal{C}, |\mathcal{O}_i| > 1 \quad (14)$$

$$\sum_{o_1 \in \mathcal{O}_i} \sum_{j \in \delta^+_D(d^0_{o_1})} u_{d^0_{o_1},j} = |\mathcal{O}_i| - 1 \qquad \forall i \in \mathcal{C}, |\mathcal{O}_i| > 1 \quad (15)$$

$$\sum_{j \in \delta^+_D(d^0_{o_1})} u_{d^0_{o_1},j} \leq 1 \qquad \forall o_1 \in \mathcal{O} \quad (16)$$

$$\sum_{j \in \delta^-_D(d^0_{o_1})} u_{j,d^0_{o_1}} \leq 1 \qquad \forall o_1 \in \mathcal{O} \quad (17)$$

$$\sum_{j \in \delta^+_D(d^0_{o_1})} u_{d^0_{o_1},j} + \sum_{j \in \delta^-_D(d^0_{o_1})} u_{j,d^0_{o_1}} \geq 1 \qquad \forall o_1 \in \mathcal{O} \quad (18)$$

$$v_j \geq w_{j-1} - M\left(1 - u_{j-1,j}\right) \qquad \forall o \in \mathcal{O}, j \in \mathcal{D}_o, j \geq 1 \quad (19)$$

$$v_j \leq w_{j-1} + \gamma_i + M\left(1 - u_{j-1,j}\right) \qquad \forall i \in \mathcal{C}, \forall o \in O_i, j \in \mathcal{D}_o, j \geq 1 \quad (20)$$

$$u_{j-1,j} \geq u_{j,j+1} \qquad \forall o_1 \in \mathcal{O}, j \in \mathcal{D}_{o_1}, 1 \leq j \leq n_{o_1} - 1 \quad (21)$$

$$u_{j-1,j} \geq \sum_{l \in \mathcal{L}} x_{lj} \qquad \forall o_1 \in \mathcal{O}, j \in \mathcal{D}_{o_1}, j \geq 1 \quad (22)$$

$$v_j \geq w_i - M\left(1 - u_{i,j}\right) \qquad \forall i \in \mathcal{L}, j \in \delta^+_L(i) \quad (23)$$

$$\sum_{j \in \delta^+_L(i)} u_{i,j} \leq 1 \qquad \forall i \in \mathcal{L} \quad (24)$$

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

$$\sum_{j \in \delta^-_L(i)} u_{j,i} + \sum_{j \in \delta^+_L(i)} u_{i,j} \geq \sum_{k \in K} \sum_{j \in \delta^-(i)} x^k_{ji} \qquad \forall i \in \mathcal{L} \quad (26)$$

$$\sum_{k \in K} \sum_{j \in \mathcal{D}_o} q^k_j = q_o \qquad \forall o \in \mathcal{O} \quad (27)$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x^k_{ij} \leq 1 \qquad i \in \mathcal{L} \cup \mathcal{D} \quad (28)$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x^k_{ij} = \sum_{k \in K} \sum_{j \in \delta^-(i)} x^k_{ji} \qquad i \in \mathcal{L} \cup \mathcal{D} \quad (29)$$

$$\sum_{k \in K} \sum_{i \in \mathcal{D}_c} \sum_{j \in \delta^-(i)} x^k_{ji} \leq M * \sigma_{c,b} \qquad \forall c \in \mathcal{C}, b \in \mathcal{B} \quad (30)$$

$$\sum_{b \in \mathcal{B}} \sigma_{c,b} \leq 1 \qquad \forall c \in \mathcal{C}, b \in \mathcal{B} \quad (31)$$

$$w^1_k \geq M_T + H_k - v_{e_k} \qquad \forall k \in K \quad (32)$$

$$w^2_k \geq (v_{e_k} - H_k) - N_T \qquad \forall k \in K \quad (33)$$

$$w_{e_k} \leq H_k + O_T \qquad k \in K \quad (34)$$

$$0 \leq q^k_j \leq Q^k \qquad j \in \mathcal{D}, k \in K \quad (35)$$

$$x_{ij}^k \in \{0,1\} \qquad (i,j) \in A, k \in K \quad (36)$$
$$u_{ij} \in \{0,1\} \qquad (i,j) \in A^D, k \in K \quad (37)$$
$$\sigma_{ib} \in \{0,1\} \qquad i \in \mathcal{C}, b \in \mathcal{B} \quad (38)$$
$$y_o \in \{0,1\} \qquad o \in \mathcal{O} \quad (39)$$
$$v_i \geq 0, w_i \geq 0 \qquad i \in V \quad (40)$$
$$w_k^1 \geq 0, w_k^2 \geq 0 \qquad i \in V \quad (41)$$
$$g_i \geq 0 \qquad i \in \mathcal{C}. \quad (42)$$

β_1 to β_4 are the penalty coefficients of each component of the objective function (2). Constraints (3–4) state that a driver k leaves his start node exactly once a day and returns to his end node after completing his last unloading operation of the day. Constraints (5–6) 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 (Eq. 7). Similarly, the unloading service depends on the site's rate and q_j^k (Eq. 8). Unloading operations must end Δ minutes after loading begins (Eq. 9).

Constraints (10–13) 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 (14–18) find the delivery sequence of all orders $\in \mathcal{O}_i$. Constraints (19–22) prohibit two trucks serving consecutive deliveries of the same order from unloading at the same time and enforce a maximum time delay between them. Similarly, constraints (23–26) ensure that two trucks cannot be loaded at the same time at a plant. Constraints (27) require that the cumulative load of all concrete mixers serving an order must equal the required quantities. Constraints (28) require that a driver can only visit a loading/delivery node once. Constraints (29) are degree constraints. Constraints (30) and (31) require that all orders from customer i must be loaded from the same plant. Constraints (32–33) calculate the difference between a driver's hours of service and the minimum and normal hours of service. Finally, constraints (34)–(42) 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 batching 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 to deliver. 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

We now 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

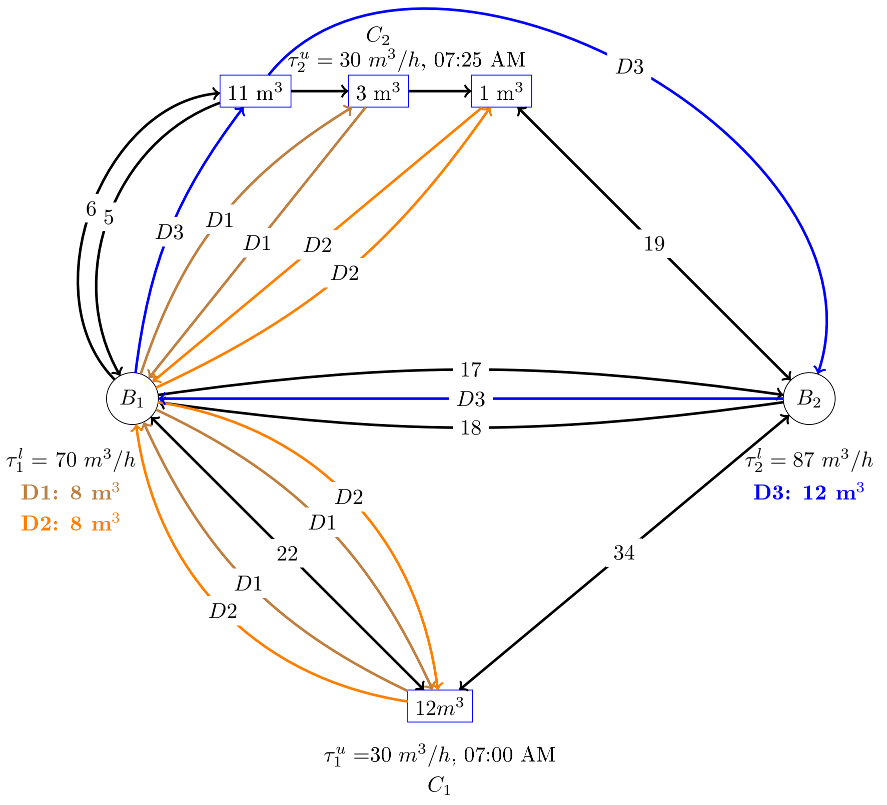

Figure 1: Solution of an instance with two plants, two construction sites, four orders, and three drivers.

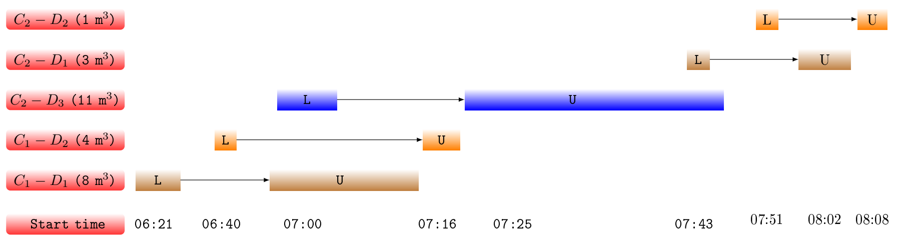

Figure 2: Schedule of the instance of Figure 1.

randomized heuristics and iteratively invoking these heuristics in the GRASP metaheuristic.

In this section, we represent the scheduling of a delivery node d as a pair of loading (L_d) and unloading (U_d) tasks. $L_d = (b, k, q_d, v_b)$ indicates that driver k starts loading q_d m^3 of RMC at plant b at time v_b. $U_d = (k, v_d, w_d)$ indicates that driver k is serving d between v_d and w_d. We then call a solution $S = \left\{\cup_{1\leq j\leq n_o}(L_{d_o^j}, U_{d_o^j}), \forall o \in \mathcal{O}\right\}$ the set of all loading and unloading tasks performed on a given day.

4.1 GRASP algorithm

The GRASP algorithm is an iterative suite of constructive and local search algorithms introduced by Feo and Resende [1995]. A feasible solution S is constructed using a greedy randomized algorithm at each iteration. Then a local search algorithm explores the neighborhood of S to find a local optimum. The pseudocode of the algorithm 1 shows that the procedure returns the best overall solution (for a minimization problem) after some stopping conditions (time limit or a maximum number of iterations) are met. In our implementation, we introduce at line 13 an additional local search step using the solutions found by all heuristics in the current iteration.

Algorithm 1 Pseudo-code of the GRASP algorithm

```
Input: $H$: List of constructive randomized heuristics
1  $S^* \leftarrow \emptyset \quad T \leftarrow \emptyset \; Cost(S^*) \leftarrow \infty$
2  while Conditions not met do
3      $Cost(S') \leftarrow \infty$
4      $T \leftarrow \emptyset$
5      for each $h \in H$
6          $S \leftarrow h(S)$
7          $S \leftarrow LocalSearch(S)$
8          $T \leftarrow T \cup \{S\}$
9          if $Cost(S) < Cost(S')$ then
10             $S' \leftarrow S$
11             if $Cost(S') < Cost(S^*)$ then
12                 $S^* \leftarrow S'$
13     $S^* \leftarrow LocalSearch(T)$
14 return $S^*$
```

4.2 Greedy randomized insertion algorithm

As mentioned in Resende and Ribeiro [2019], the general outline of a greedy randomized algorithm used in the GRASP framework works as described in Algorithm 2. Let's consider the set CL of candidate delivery nodes that have not yet been scheduled. We first initialize CL with the first delivery node of a randomly selected order, for each customer.

At each iteration, we create the set CL' with elements of CL from a given neighborhood. We then evaluate the increase in the cost function of incorporating each $d \in CL'$ into the incumbent solution S and construct a restricted candidate list RCL consisting of nodes whose incremental cost is less than a given threshold. From RCL, we randomly select the next delivery node to be incorporated into the incumbent solution and determine its loading and unloading schedule using the procedure described in Algorithm 3. Next, we update S and the remaining order and customer quantities. Then we add the next candidate delivery node to CL. If an order o is not yet fulfilled after visiting d_o^j, the candidate is the next delivery node d_o^{j+1}. Otherwise, if the customer has remaining orders, the candidate is the first delivery node $d_{o'}^0$ of another randomly selected order o'. Note that for our problem, not all unscheduled delivery nodes are candidates at each iteration, since deliveries of the same order are ordered, any order of a customer can be chosen and must be satisfied before switching to another order, and the number of visits to a customer is not known in advance.

Consider two customers whose services are required to start at the same time or in the same time slot. We could either finish scheduling one before the other, or schedule both at the same time. Obviously, each of these choices would result in a different solution. Similarly, in some cases, it would be better to schedule a customer that has a high demand and therefore requires more resources first. That's why we added the filtering part to impose some criteria on the algorithm. We can limit CL' to delivery nodes with the same demand, the same due date, or from the same customer. We can also decide if we want to schedule customers with the highest (or lowest) demand or the earliest (latest) due time first. In fact, this helps us to intensify the search in specific neighborhoods of the solution. To diversify the search, we simply remove the filter component. The greedy aspect of GRASP is the creation of the set RCL, the probabilistic aspect is the random selection in RCL, and the adaptive aspect is the updating of CL and the re-evaluation of the incremental costs.

Algorithm 2 Greedy randomized 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      $CL \leftarrow CL \cup \{d^0_{o_1}\}$
5  while $CL \neq \emptyset$ do
6      $CL' \leftarrow$ Filter (CL)
7      for each $d \in CL'$
8          $C(d) \leftarrow$ incremental cost of inserting $d$
9      $C_{min} \leftarrow min\,\{C(d),\ d \in CL'\}$, $C_{max} \leftarrow max\,\{C(d),\ d \in CL'\}$
10     $RCL \leftarrow \{d \in CL',\ C(d) \leq C_{min} + \alpha(C_{max} - C_{min})\}$
11     Select random delivery node $d^j_o$ of customer $i$ from $RCL$
12     $L_{d^j_o}, U_{d^j_o}$ = ScheduleTasks($d^j_o, S$)
13     $S \leftarrow S \cup (L_{d^j_o}, U_{d^j_o})$
14     $q_i \leftarrow q_i - q_{d^j_o}$; $q_o \leftarrow q_o - q_{d^j_o}$
15     if $q_o \neq 0$ then
16         $CL \leftarrow CL \cup \{d^{j+1}_o\}$
17     else if $q_i \neq 0$ then
18         Select another order $o' \in \mathcal{O}_i$
19         $CL \leftarrow CL \cup \{d^0_{o'}\}$
20     Update $CL$
21 return $S$
```

Algorithm 3 takes as parameters a delivery node d^j_o of the customer i and a partial solution and looks for a plant and a driver available for loading and unloading operations. For each driver k and plant b we simulate the loading and unloading operations while checking the concrete lifespan and cold joint constraints, and the plant and driver availability. For each node, we first determine the start loading time SLT by subtracting t_{bi}, α_b, and the loading duration $LD^b_{d^j_o}$ to the expected delivery time EDT. EDT is either the arrival time a_i for the first delivery of a customer or the end of the unloading of the precedent delivery. At line 15, we ensure with the procedure *FindDriverLoadingSlot* that the loading starts only if the driver is present at the depot. The presence of the driver does not guarantee that the loading dock is available, thus we use the procedure *FindDepotLoadingSlot* at line 16 to ensure that. Once we know SLT, we can easily deduce the arrival time (AT), start unloading time (SUT), driver waiting time ($W^k_{d^j_o}$), and client waiting time ($W_{d^j_o}$). We also keep track of the driver's work duration. The algorithm returns the best loading and unloading operations with the least cost. We keep track of all timeslots for a plant when its loading dock is busy, and for a driver when he starts loading until the end of the unloading service. Within *FindDepotLoadingSlot* (*FindDriverLoadingSlot*), we iterate through the timeslots of a depot (driver) to schedule the current operation at a free timeslot. These allow us to insert the scheduling of a node between existing schedules.

Algorithm 3 Schedule loading and unloading tasks

```
Input: Partial solution $S$, delivery node $d_o^j$
Function ScheduleTask($d_o^j, S$)
    $n_i$: last node visited for the customer $i$
    $\mathcal{LS}$ : Partial solutions list
    $bestSol$ Cost($bestSol$) $= +\infty$
    for each driver $k$
        $n_k$: current location of $k$ (plant or delivery node)
        for each plant $b$
            if $t_{bi} >= \delta$ then continue
            $EDT = SUT = a_i$
            if $n_i \neq null$ then
                $SUT = rand(EUt_{n_i} - \gamma/3, EUt_{n_i})$
                $EDT = EUT_{n_i}$
            $q_o^j = min\{Q_k, q_o\}$ // $q_o$ remaining demand of order $o$
            $SLT = EDT - t_{bi} - \alpha_b - LD^b_{d_o^j}$
            $SLT = FindDriverLoadingSlot(b, n_k, SLT, LD^b_{d_o^j})$
            $SLT = FindDepotLoadingSlot(b, SLT, LD^b_{d_o^j})$
            $AT = SLT + LD^b_{d^j o} + \alpha_b + t_{bi}$
            $SUT = max\{AT, EDT\}$
            $W^k_{d_o^j} = max\{0, SUT - AT\}$
            $W_{d_o^j} = max\{0, AT - EDT - \lambda_i\}$
            $EUT = SUT + UD_{d^p_{ij}}$
            $WT = WT_k + (EUT + \beta_k - SLT) + t_{n_k b} + t_{i,k}$
            $TC = t_{n_k,b} + t_{b,i}$
            if $Cost(bestSol) < Cost(S) + TC$ then
                $bestSol = (j, b, k, SLT) \cup (j, b, k, SUT)$
                $Cost(bestSol) = Cost(S) + TC$
    return $bestSol$
```

4.3 Local Search

5 Experimental results

5.1 Generation of instances

5.2 Data sets

5.2.1 Results for the generated instances

5.2.2 Results for the real-world datasets

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

We run our algorithm on the dataset provided in Kinable et al. [2014].

6 Conclusion

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. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

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.

M. G.C Resende and C. C Ribeiro. Greedy randomized adaptive search procedures: advances and extensions. In *Handbook of metaheuristics*, pages 169–220. Springer, 2019.

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.