# 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

December 21, 2022

**Abstract**

**Keywords:** Vehicle routing; scheduling; ready-mixed concrete; concrete delivery

## 1 Introduction

Construction projects involve a huge movement of equipment, people, and materials. Among the latter, concrete is one of the most used. It is a perishable product with many factors affecting its quality [Sinha et al., 2021]. It comes in two types: ready-mixed concrete (RMC) and site-mixed concrete (SMC). As its name implies, SMC is produced on the spot using raw materials (water, aggregates, and cement) stored on the construction site, while RMC is manufactured in a batch plant and delivered to the construction site. SMC can avoid delays caused by road traffic, but has a slower and more difficult production process, requires storage for mixing materials and equipment, and is suitable for low amounts of concrete. It is quite the opposite for the RMC, which has better quality and benefits from lower production cost [Muresan, 2019]. Nevertheless, the batch plant manager must employ a fleet of high-cost revolving drum trucks (concrete mixers) to dispatch the ready-mixed concrete. Hence, the key to reaping the RMC benefits is to ensure efficient and prompt delivery on the construction site.

Concrete delivery under the form of RMC is subject to many operational constraints that make it a challenging problem met in Operations Research and addressed by the Concrete Delivery Problem (CDP). In this paper, we study a variant of the Concrete Delivery Problem to schedule the daily production and dispatching of RMC for a company located in the province of Quebec, Canada. This company produces RMC in multiple batching plants with different production rates, using a fleet of concrete mixers of various capacities. Each plant has its specific fleet, but a truck can load elsewhere if deemed necessary, except it must return to its home plant at the end of the day. They own two types of trucks with different capacities but can call on an external fleet when needed. They serve a construction site from any batch plant, with the first delivery starting at the time specified by the customer. Loading and unloading a concrete mixer depends on the truck capacity, a plant's loading, and a construction site's unloading rate, respectively. The problem is like the one addressed in Schmid et al. [2009, 2010], except for the plant's production rate and the lack of unloading instrumentation in our case. Also, as per union rules, they assign a truck driver to a delivery task based on his seniority. 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 concrete delivery problem combines vehicle routing with scheduling issues to plan routes to deliver concrete from batch plants (depots) to customers' construction sites. Ready-mixed concrete is an on-demand product with a short life cycle from production through end use. It cannot be stored and cannot stay too long in a truck, or it will harden. Hence, concrete mixers must deliver RMC at the planned construction site shortly after its production. RMC production and delivery is then an example of a Just-In-Time (JIT) production system in construction [Tommelein and Li, 1999]. A customer quantity requirement is often greater than the truck size and must be fulfilled by multiple deliveries. In that sense, CDP is like the Vehicle Routing Problem (VRP) with split delivery (VRPSD) [Archetti and Speranza, 2008], except that the same truck may visit a customer more than once.

Since concrete hardens quickly, in case of multiple deliveries, back-to-back deliveries must be continuous or at least close in time to avoid the problem of cold joint that negatively affects the quality of the concrete. Customers request to be served within a specific time window, complicating truck loading schedules when a plant can only load one truck at a time. Likewise, often 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 Concrete Delivery Problem is a complex problem to solve and has been proven by Asbach et al. [2009] to be NP-hard. With a single batching plant, the objective is to maximize the service level (serve all customers and avoid delays between subsequent deliveries and truck queuing). With additional production centers to choose from, a minimization of the total travel time is added to the objective.

The remainder of the paper is organized as follows. Section 2 presents an overview of the CDP-related literature. Section 3 provides a formal description of the problem. Then in Section 4, we describe the Greedy Randomized Search Procedure (GRASP) algorithm and the constructive heuristics we design to solve this variant of the CDP. Computational experiments and our conclusions follow in Sections 5 and 6, respectively.

## 2 Literature review

A text mining approach for reviewing the ready-mixed concrete literature [Maghrebi et al., 2015] showed that concrete technology and material science are the main core of research in this area. The first academic research on concrete batching and delivery began in the late 1990s. Tommelein and Li [1999] describe RMC as a prototypical example of a Just-In-Time production system in construction and identify two practices occurring when delivering it. An alternative is for the customer to haul the product from the batch plant with their concrete mixer. The other approach is where the batch plant delivers the concrete to the customer's location. This latter approach is the one studied in all related papers found in the literature.

Several works on the CDP mainly focussed on Simulation related methods, either standalone [Zayed and Halpin, 2001, Wang et al., 2001, Tian et al., 2010, Panas and Pantouvakis, 2013, Galić and Kraus, 2016], etc. or hybridized with optimization methods [Feng et al., 2004, Lu and Lam, 2005, Feng and Wu, 2006], etc. to schedule and dispatch concrete production and delivery. 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. With a combination of Genetic Algorithm (GA) and simulation process,

Feng et al. [2004] tried to minimize the total waiting time for trucks at a customer site. Loading trucks with identical capacities is carried out at the same batch plant. Loading and unloading durations are fixed. GA is used to find the most efficient and effective loading sequence of RMC trucks to be assigned to different construction sites. The simulation process determines the loading, arrival, departure, and waiting time of trucks and thus evaluates the cost of each dispatching sequence. They evaluated the method using data from a batch plant in Taiwan with up to nine customers served. Mayteekrieangkrai and Wongthatsanekorn [2015] solved 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 decide on the optimal number of concrete mixers to be deployed together with an optimal schedule for batching and delivering concrete. Their objective is to minimize the site crew idle times due to late concrete deliveries plus truck queuing time. The particularity of this setting is that mortar useful for unloading pump lubrication must be delivered on-site before concrete. Therefore, mortar batching and delivery are also modeled in the simulation. Finding the best RMC truck size was also the purpose of the discrete-event simulation model proposed by Panas and Pantouvakis [2013].

Besides simulation-based methods, we find in the literature works using metaheuristics [Faria et al., 2006, Misir et al., 2011, Maghrebi et al., 2016b, Yang et al., 2022], exact methods [Yan and Lai, 2007, 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 approach [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 sites to help the concrete unloading. Three plants are available, but vehicles fulfilling the same order must all load at the same plant. An order that cannot be executed may be postponed for the next day. The routing of the pumps is modeled as a multi-depot VRP with time windows. Naso et al. [2007] proposed a sequential GA method combined with constructive heuristics to solve a more general variant of the CDP. In this problem, besides batching concrete delivered to a customer site, the plant's production schedule must include orders picked up by the customers themselves. The algorithm first schedules the plant loading operations before scheduling truck job deliveries. A non-linear model minimizing transportation costs, waiting times, outsourced costs, and overtime work is also developed. The authors ran experiments using real-world instances of a concrete supply chain in the Netherlands. They found a reduction in the number of requests redirected to external companies. Yan and Lai [2007] also emphasized overtime considerations in their paper scheduling RMC for one batching plant with two loading docks. Overtime wages are paid for factory and construction site operations after 4 PM. A mixed integer programming (MIP) model on a time-space network is developed to minimize travel times and operating costs at both normal and overtime working hours at the plant and the construction sites. Real data consisting of 3 days of operation is tested using a two-stage algorithm. First, they solved the MIP relaxation with CPLEX. Then they simplify the original model by fixing some decision variables before solving it. This algorithm seems to improve the actual plant operation by 10%. A time-space network is the principal component of the real-time DSS developed by Durbin and Hoffman [2008] to solve a dynamic CDP every five minutes. The DSS can receive new orders, schedule them on the fly, and deal with unexpected events such as plant closures, truck breakdowns, delays in transportation times, etc. Combined with a Tabu Search (TS) heuristic to warm start CPLEX, the model is performant enough to solve instances with up to 1,500 loads per day with up to 250 trucks. The authors consider the case of a customer who places two orders, with the first being completed

before the second starts. The real-time planning and monitoring of CDP are studied more in detail by Garza Cavazos [2021] in his thesis. 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 illimited number of trucks simultaneously. Among the trucks, some have specialized equipment and must arrive first at certain construction sites to assist concrete-mixers unloading. 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 solve large problem instances with more than 60 orders per day quickly without encountering any memory issues. The same problem is addressed by Schmid et al. [2010]. The authors 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 been interested in minimizing travel time and delays between consecutive deliveries. Some authors, however, were more interested in maximizing only customer satisfaction. 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 also 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 relatively little 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 citekinable2014concreteand of two more compact models can be found in the thesis of Hernández López [2020]. In Kinable and Trick [2014], we find 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) with a Time-Slot Heuristic (TH). TH mechanism is to look 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.

# 3 Problem description

The problem studied within this paper is about the distribution of ready-mixed concrete from a Canadian company operating in the greater Montreal area. Concrete order from a customer arrives at a central center and is assigned to one of the batching plants from where the product will be produced and delivered to the customer. We define an instance of our problem on a set of batching plants, a set of customer orders, and a set of drivers.

The company owns and operates a total of eight concrete batching plants located in different geographical areas. The loading bays in each plant can only accommodate one truck at a time, resulting in trucks lining up. Let $\mathcal{B}$ be the set of batching plants. The plants are heterogeneous, as each plant $b$ has its hourly loading rate $\tau_b^l$ which influences the loading time duration. After loading, a driver takes $\alpha_b$ minutes to adjust the concrete in his truck before departing to the customer site. Any batch plant can serve a construction site if the traveled time between them is less than the concrete lifespan $\delta$. Each plant has its assigned fleet, but it can get a vehicle from another plant or even hire an external fleet. If $q_i$ is the quantity of RMC loaded to customer $i$ from plant $b$, we compute the loading time duration $LD_i^b$ as

$$LD_i^b = \frac{q_i}{\tau_b^l}. \tag{1}$$

A customer $i$ requests one or several types of concrete to be delivered to his construction site on a particular day for a total amount of $q_i$. When placing an order, he specifies the number of types of RMC $P_i \geq 1$, the quantity $q_i^p$ of concrete of each type $p$, the arrival time $a_i$ of the first concrete mixer, and the unloading rate $\tau_i^u$ at his site. $\mathcal{C}$ is the set of construction sites (customers) with a planned delivery for the day.

We refer to order $o_i^p$ as the request of customer $i$ to receive a type $p$ of concrete. If the amount of concrete exceeds any truck capacity, several deliveries are scheduled to satisfy an order. $n_i^p \geq 1$ is the number of deliveries needed to satisfy $o_i^p$. Let $d_{ij}^p$ be the $j_{th}$ visit with load $q_{ij}^p$ for order $o_i^p$. We represent the delivery of order $o_i^p$ by the visits to the ordered set of nodes $\mathcal{D}_i^p = (d_{i0}^p, d_{i1}^p, \cdots, d_{in_i^p}^p)$. All quantities of a type of concrete must be thoroughly delivered before moving on to another type. The orders $o_i^p$ $(1 \leq p \leq P_i)$ of a customer $i$ are not ordered. An order can start at any time, provided it is completed before delivering another. We represent the delivery of customer $i$ by the set $\mathcal{D}_i = (\mathcal{D}_i^{p^0}, \mathcal{D}_i^{p^1}, \cdots, \mathcal{D}^{p^{P_i}})$, where $p^l$ is the $lth$ order delivered. We define for the first node of $\mathcal{D}_i^{p^0}$ the time window $[a_i, b_i]$ to ensure that the first delivery of $i$ must be due at $a_i$. Once a plant is chosen to produce a customer's first order, it must be the provider of all subsequent orders of this customer.

To prevent cold joint in concrete, subsequent deliveries must be in just in time, or at least close in time. We define a maximum time lag $\gamma_i$ beyond which no next unloading operation should be allowed. The construction site unloading rate and the quantity to unload give the time necessary to discharge a truckload. The unloading time duration of $q_{ij}^p$ is

$$UD_{d_{ij}^p} = \frac{q_{ij}^p}{\tau_i^u} \tag{2}$$

The company has two types of trucks with capacities of 8 and 12 $m^3$. A driver $k$ is assigned to a plant from where he starts and ends his shift day, and drives one concrete mixer of capacity $Q_k$. The set of drivers is $K = \cup_{b \in \mathcal{B}} K_b$, $K_b$ being the set of drivers scheduled to start their shift at the batching plant $b$.

A driver must work between four and eight hours in normal time, plus optional overtime of two hours at most, during a scheduled day. A driver mainly loads RMC at his home plant but can still drive at other plants and load there if needed. Batching plant produces concrete on-demand with recipes specific to customers, which means that truck cannot hold orders for more than one client, even if it has spare capacity. Thus, a driver must refill at a plant between two consecutive deliveries. After unloading the RMC, a driver takes $\beta_k$ minutes to clean the concrete mixer before traveling to his next loading plant. When assigning a driver to a delivery, priority is given to the employees with the highest seniority.

For an order $o_i^p$, the number of deliveries $|D_i^p|$ is not known a priori, as we use a fleet with various capacities. However, we can compute the lower and upper bounds of $|D_i^p|$ using the capacities of the highest ($Q_{max}$) and smallest ($Q_{min}$) available trucks. The lower (upper) bound $\left\lceil \frac{q_i^p}{Q_{max}} \right\rceil$ $\left( \left\lceil \frac{q_i^p}{Q_{min}} \right\rceil \right)$ is the number of deliveries needed if we only use trucks of capacity $Q_{max}$ ($Q_{min}$).

A solution to our problem is a set of decisions about truck loading schedules, driver assignments for different deliveries, and truck arrival times at construction sites for unloading. For a batching plant, the decision is to determine which driver should be loaded, when, and for which construction site. For a driver, the decision is the sequence of loading depots and delivering sites. And for a construction site, the arrival time of all daily scheduled deliveries.

Let the binary variable $x_j^{bk}$ be 1 if a driver $k$ loads at plant $b$ to serve the delivery node $j$. Let $SLT_j^{bk}$ and $ELT_j^{bk}$ be the start and end loading time of driver $k$ at plant $b$. $SUT_j$, $EUT_j$ be the start and end unloading time. $LTS_j^{bk} = \left[SLT_j^{bk}, ELT_j^{bk}\right]$, and $UTS_j^{bk} = \left[SUT_j^{bk}, EUT_j^{bk}\right]$ stand for the loading and unloading operations timeslots. $AT_j$ is the arrival time of a driver at the node location.

$$ELT_j^{bk} = SLT_j^{bk} + LD_j^b \tag{3}$$

$$EUT_j = SUT_j + UD_j \tag{4}$$

For each delivery node $j$, we define an expected delivery time $EDT_j$ of unloading operation, the driver waiting duration $W_j^k$, and the node waiting time $W_j$.

$$EDT_{d_{ij}^{p^l}} = \begin{cases} a_i & \text{if } j = 0,\ l = 0 \\ E_{d_{ij-1}^{p^l}} & j \geq 1 \\ E_{in_i^{p^{l-1}}} & \end{cases} \tag{5}$$

$$W_{d_{ij}^p}^k = \max\{0, SUT_{d_{ij}^p} - AT_{d_{ij}^p}\} \tag{6}$$

$$W_{d_{ij}^p} = \max\{0, AT_{d_{ij}^p} - EDT_{d_{ij}^p} - \lambda_i\} \tag{7}$$

We keep track of the working time of each driver $k$ with the variable $WT_k$. Parameters $N_t$ and $O_t$ are the normal and overtime working hours. $M_t$ is the minimum working time for a day. $\Gamma_1$ and $\Gamma_2$ are the incurred penalties when a driver works less than $M_t$ and more than $N_t$.

$$\forall k \in K, WT_k \leq O_t \tag{8}$$

The objective is to minimize the traveled distance, of delivered concrete to the satisfied customers as in

$$\min \sum_{i,j} t_{ij} x_{ij} \tag{9}$$

The cost to travel between two nodes ($i$, $j$) is the travel time $t_{ij}$. COST...

Let us consider the small instance illustrated 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 $D_3$ of capacity 12 m$^3$. Figure 1 depicts the concrete mixers flow in the network. We schedule $D_1$ and $D_2$ to respectively deliver 8 and 4 m$^3$ to $C_1$. 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 lag of 20 minutes defined for this example.

# 4 Constructive heuristics and GRASP

We now present the solution approach we implement to solve this variant of the Concrete Delivery Problem. Our method consists in constructing feasible solutions for the CDP with randomized heuristics and iteratively calling these heuristics in the greedy randomized adaptive search procedure (GRASP) metaheuristic.

In this section, we represent the scheduling of a delivery node $d_{ij}^p$ as a pair of loading $l_{d_{ij}^p}$ and unloading $u_{d_{ij}^p}$ tasks. We then refer to a solution $S$ as a set of loading and unloading tasks performed by a set of drivers during a particular day.

$$l_{d_{ij}^p} = \left\{ (b, k, q_{ij}^p, LTS_{d_{ij}^p}^{bk}), b \in \mathcal{B}, k \in K \right\}$$
$$u_{d_{ij}^p} = (k, A_{d_{ij}^p}, UTS_{d_{ij}^p}), k \in K\}$$
$$S = \left\{ \cup_{1 \leq p \leq n_i^p} \cup_{1 \leq j \leq |D_i^p|} (l_{d_{ij}^p}, u_{d_{ij}^p}), i \in \mathcal{C} \right\}$$

## 4.1 GRASP algorithm

The GRASP algorithm is an iterative suite of constructive and local search algorithms introduced by Feo and Resende [1995]. For each iteration, a feasible solution $S$ is built using a greedy randomized algorithm. Then a local search algorithm investigates the neighborhood of $S$ to find a local optimum. The pseudo-code of Algorithm 1 shows that the procedure returns the best overall solution (for a minimization problem) after some stopping conditions (time limit or a maximal number of iterations) are met.

## 4.2 Greedy randomized insertion algorithm

As referred to in Resende and Ribeiro [2019], the general outline of a greedy randomized algorithm used in the GRASP framework works as described in Algorithm 2. At each

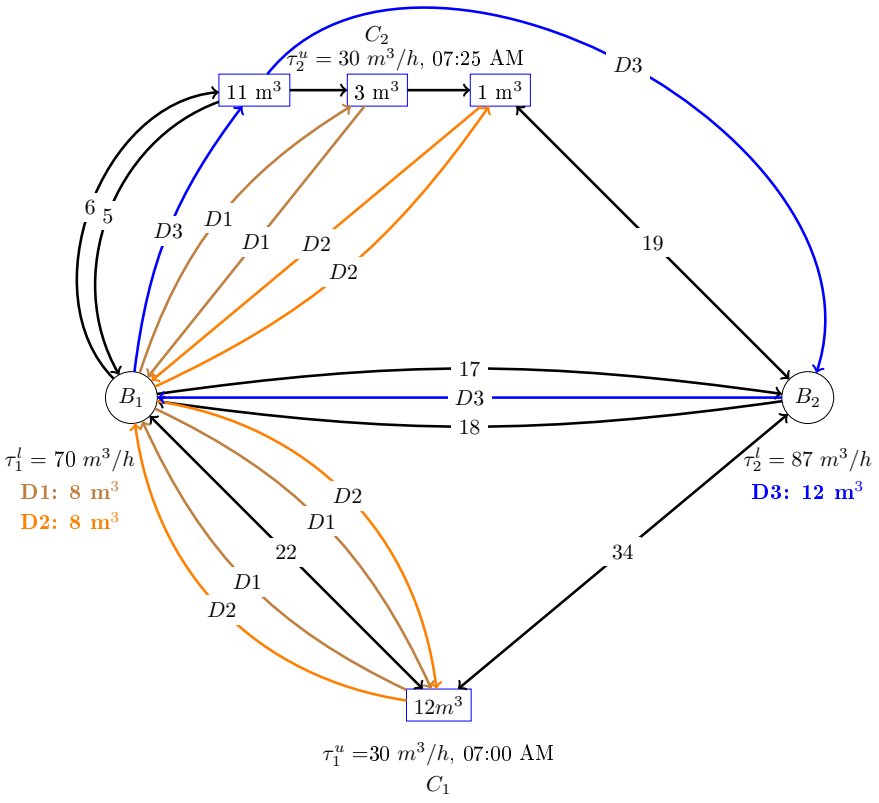


Figure 1: Solution of an instance with two plants, two construction sites, four orders, and three drivers.

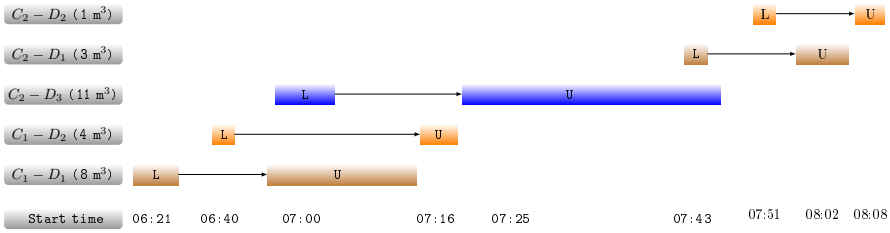


Figure 2: Schedule of the instance of Figure 1.

iteration, let's consider the set $CL$ of candidate delivery nodes that are not yet scheduled. We note that, for our problem, not all delivery nodes are candidates for insertion at each iteration, since deliveries of the same order are ordered, any order of a customer may be chosen, and it must be satisfied before switching to another order, and the number of visits to a customer is not known beforehand. We first initialize $CL$ with the first delivery of a random order for each customer. We create the set $CL'$ with elements of $CL$ we deem promising for a better solution. We evaluate the increase in the cost function when incorporating each $d \in CL'$ into the incumbent solution $s$ and create a restricted candidate

**Algorithm 1** Pseudo-code of the GRASP algorithm

**Input:** $H$: List of constructive randomized heuristics

```
1  S* ← ∅    Cost(S*) ← ∞
2  while Conditions not met do
3  |  Cost(S') ← ∞
4  |  for each  h ∈ H
5  |  |  S ← h(S)
6  |  |  S ← LocalSearch(S)
7  |  |  if Cost(S) < Cost(S') then
8  |  |  |  S' ← S
9  |  |  |  if Cost(S') < Cost(S*) then
10 |  |  |  |  S* ← S'
11 return S*
```

list $RCL$ formed by nodes whose incremental costs are less than a defined threshold. From $RCL$, we randomly select the next delivery node to be incorporated into the incumbent solution and we determine its loading and unloading schedule with the procedure described in Algorithm 3. Next, we update $S$ and the remaining quantities of the order and customer. Then we add to $CL$ the next delivery node candidate. If an order $o^p$ is not yet fulfilled after the visiting $d^p_{ij}$, the candidate is the next delivery node $d^p_{ij+1}$. Otherwise, if the customer has remaining orders, the candidate is the first delivery node $d^{p'}_{i0}$ of another randomly selected order $p'$.

The greedy aspect of the GRASP is the creation of the $RCL$ set, the probabilistic aspect is the random selection in $RCL$, and the adaptive aspect is the update of $CL$ and the reevaluating of the incremental costs. We use the additional set $CL'$ because we noticed that restricting $CL$ to delivery nodes with the same demand, delivery due time, or overlapping unloading timeslots helps us intensify the search. Using $CL'$ also helps improve the algorithm complexity by reducing the number of nodes to evaluate. To diversify the search we simply remove the filter component. We thus obtain four greedy randomized insertion heuristics for our GRASP framework.

Algorithm 3 takes as parameters a delivery node $d^p_{ij}$ 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 constraints, and the plant and driver availability. For each node, we first determine the start loading time $SLT$ by subtracting $t_{ij}$, $\alpha_b$, and the loading duration $LD^b_{d^p_{ij}}$ 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, start unloading time, driver waiting time, and client waiting time. 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 before existing schedules.

**Algorithm 2** Greedy randomized insertion algorithm

**Input:** $S$: empty solution $S$

$CL \leftarrow \emptyset$
**for each** customer i
  Select a random order $o_i^p$
  $CL \leftarrow CL \cup \{d_{i0}^p\}$
**while** $CL \neq \emptyset$ **do**
  $CL' \leftarrow$ Filter (CL)
  **for each** $d \in CL'$
    $C(d) \leftarrow$ incremental cost of inserting $d$
  $C_{min} \leftarrow min\{C(d),\ d \in CL'\}$, $C_{max} \leftarrow max\{C(d),\ d \in CL'\}$
  $RCL \leftarrow \{d \in CL,\ C(d) \leq C_{min} + \alpha(C_{max} - C_{min})\}$
  Select random delivery node $d_{ij}^p$ from $RCL$
  $l_{ij}^p, u_{ij}^p =$ ScheduleTasks$(d_{ij}^p, S)$
  $S \leftarrow S \cup (l_{ij}^p, u_{ij}^p)$
  $q_i \leftarrow q_i - q_{ij}^p$; $q_i^p \leftarrow q_i^p - q_{ij}^p$
  **if** $q_i^p \neq 0$ **then**
    $CL \leftarrow CL \cup \{d_{ij+1}^p\}$
  **else if** $q_i \neq 0$ **then**
    Select another order $o_i^{p'}$
    $CL \leftarrow CL \cup \{d_{i0}^{p'}\}$
  Update $CL$
**return** $S$

### Local Search

# 5 Experimental results

## 5.1 Data sets

# 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.

**Algorithm 3** Schedule loading and unloading tasks

**Input:** Partial solution $S$, delivery node $d_{ij}^p$

**Function** `ScheduleTask`$(d_{ij}^p, S)$
- $n_i$: last node of $i$ visited
- $\mathcal{LS}$ : 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**
      - $SUT = rand(EUt_{n_i} - \gamma/3, EUt_{n_i})$
      - $EDT = EUT_{n_i}$
    - $q_{ij}^p = min\{Q_k, q_i^p\}$ // $q_i^p$ *remaining demand of order* $o_i^p$
    - $SLT = EDT - t_{bi} - \alpha_b - LD^b_{d_{ij}^p}$
    - $SLT = FindDriverLoadingSlot(b, n_k, SLT, LD^b_{d_{ij}^p})$
    - $SLT = FindDepotLoadingSlot(b, SLT, LD^b_{d_{ij}^p})$
    - $AT = SLT + LD^b_{d^p ij} + \alpha_b + t_{bi}$
    - $SUT = max\{AT, EDT\}$
    - $W_d^k = max\{0, SUT - AT\}$
    - $W_d = max\{0, AT - EDT - \lambda_i\}$
    - $EUT = SUT + UD_{d_{ij}^p}$
    - $WT = WT_k + (EUT + \beta_k - SLT) + t_{n_k b} + t_{j,k}$
    - $TC_{k,j} = t_{n_k,b} + t_{b,j} - t_{pos_k,k}$
    - **if** $Cost(bestSol) < Cost(S) + TC_{k,j}$ **then**
      - $bestSol = (j, b, k, SLt) \cup (j, b, k, SUt)$
      - $Cost(bestSol) = Cost(S) + TC_{k,j}$
- **return** $bestSol$


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, S. T. Waller, and C. Sammut. Text mining approach for reviewing the ready mixed concrete literature. In *2nd International Conference on Civil and Building Engineering Informatics, University of Osaka, Tokyo, Japan*, pages 105–109, 2015.

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.

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.