Hybridization of Very Large Neighborhood Search for Ready-Mixed Concrete Delivery Problems

Verena Schmid[(1)], Karl F. Doerner[(1)],
Richard F. Hartl[(1)], Juan-José Salazar-González[(2)]

(1) Department of Business Administration, University of Vienna,
Bruenner Strasse 72, 1210 Vienna, Austria
{Verena.Schmid, Karl.Doerner,
Richard.Hartl}@univie.ac.at

(2) Facultad de Matemáticas, Universidad de La Laguna,
38271 La Laguna, Tenerife, Spain
jjsalaza@ull.es

Abstract

Companies in the concrete industry are facing the following scheduling problem on a daily basis: concrete produced at several plants has to be delivered at customers' construction sites using a heterogeneous fleet of vehicles in a timely, but cost-effective manner. The distribution of ready-mixed concrete (RMC) is a highly complex problem in logistics and combinatorial optimization.

This paper proposes a hybrid solution procedure for dealing with this problem. It is based on a combination of an exact algorithm and a Variable Neighborhood Search approach (VNS). The VNS is used at first to generate feasible solutions and is trying to further improve them. The exact method is based on a Mixed Integer Linear Programming (MILP) formulation, which is solved (after an appropriated variable fixing phase) by using a general-purpose MILP solver. An approach based on Very Large Neighborhood Search (VLNS) determines which variables are supposed to be fixed. In a sense, the approach follows a local branching scheme. The hybrid metaheuristic is compared with the pure VNS approach and the conclusion is that the new metaheuristic outperforms the VNS if applied solely.

Keywords: hybrid, ready-mixed concrete, variable neighborhood search, capacitated vehicle routing problem, very large neighborhood search.

1 Introduction

The following problem has to be solved by companies in the concrete industry on a daily basis: concrete produced at several plants has to be delivered at customers' construction sites using a heterogeneous fleet of vehicles in a timely, but cost-effective manner. The distribution of ready-mixed concrete (RMC) is a complex problem in logistics and combinatorial optimization. One needs to assign capacitated vehicles (trucks) to deliveries and also decide from which plant concrete will be transported to the associated construction site. Further all resulting truck movements need to be scheduled accordingly. Additionally a large number of technical constraints dealing with the time for unloading operations also need to be taken into account. Typically companies rely on skilled dispatchers that schedule truck assignments to single deliveries (their movements throughout the day) such that the total demand can be satisfied.

As the ordered quantity of concrete typically exceeds the capacity of a single vehicle, several consecutive deliveries need to be scheduled to fulfill a single order. At most one truck may unload at a time at a given construction site (i.e., single deliveries to be executed at one construction site may not overlap). Loading operations of several trucks in a plant is not a trouble in reality, and indeed for our problem the plants can be considered uncapacitated for those aspects. Satisfying customer orders on time is essential, thus time windows need to be considered. Constructors require an uninterrupted supply of concrete, hence the time between consecutive deliveries needs to be kept as small as possible.

Some vehicles may be used for the delivery of concrete only. Other vehicles, with specialized unloading equipment (pumps, conveyor belt), may have to be present at a construction site and assist other vehicles with their unloading operation. Some of the vehicles with specialized instrumentation may also be able to transport concrete in case they have loading space. If special unloading equipment is demanded by an order, a truck with specialized instrumentation needs to arrive first to a construction site and remains there until the complete order has been fulfilled. It is not possible to displace the truck assisting others with their unloading operations. If an order requires a special unloading equipment, the first truck to arrive needs to have such specialized instrumentation and it must stay there until the last truck serving that order has finished its unloading operation. Any displacement of the truck with specialized instrumentation is forbidden in practice because it could disturb the unloading process at a construction site.

Concrete is not a homogeneous product, rather many different recipes exist. As opposed to many other problems related to vehicle routing, any truck may only service one order at a time. It is not allowed to execute several deliveries without being loaded in between, even if the capacity of the truck under consideration would be large enough to execute several small deliveries. Hence trucks might only be partially loaded when serving deliveries. It is not possible to store RMC. Every load of concrete is made just-in time according to the specifications and requirements of each order. Therefore all trucks basically

commute between a plant (where they are loaded) and construction sites (where unloading operations are supposed to take place). Every truck, after unloading concrete at the construction site of the order where it was assigned, must drive to a plant again. By the end of the working period (day) the truck needs to return back to its home plant.

The objective is to minimize total cost, consisting of total travel cost and penalties for delays between any two consecutive unloading operations for an order.

Some related work on scheduling and dispatching trucks for the delivery of concrete can be found in the literature: An overview of the main characteristics related to the delivery and production of RMC can be found in Tommelein and Li (1999). Within their paper they considered RMC delivery as a prototypical example for a just-in-time production system, which is batched to specifications according to customers demand. Alternative forms of vertical supply chain integration were investigated, based on data from industry case studies.

Matsatsinis (2004) presents an approach for designing a decision support system for dynamic routing of trucks in order to distribute ready-mixed concrete. Matsatsinis concentrates on the decision support system; routing is done using heuristics. Different from our approach, Matsatsinis separates the scheduling of vehicles with special delivery equipment and concrete-carrying vehicles. Furthermore, vehicles fulfilling the same order all have to load at the same plant.

Naso et al. (2007a) implemented a hybrid approach combining a constructive heuristic based on a genetic algorithm (GA). During a preprocessing stage orders are split into several jobs (deliveries) based on a fixed vehicle capacity. Unlike our approach they decompose the problem. First jobs and all resulting loading operations are assigned to plants using a GA. The routing of trucks is executed at a second step. This is done by means of a constructive heuristic, which makes sure the overall schedule gets feasible, based on the assignment done by GA. Unlike our approach their fleet of vehicles is supposed to be homogenous in terms of their capacity and is used for the delivery of concrete solely. Hence the number of deliveries necessary to completely satisfy an order may be clearly determined. Specialized unloading equipment such as pumps or conveyor belts and the resulting assistance during the unloading operation of other trucks is not considered. Time windows need to be kept and a strictly uninterrupted supply of concrete is needed. To overcome potential bottleneck situations when many tight time windows need to kept they also consider outsourcing production and hiring trucks externally as an option. Their objective is threefold. Transportation costs, in terms of distance traveled, the time for loading and unloading waiting times as well as additional costs related to outsourced production, hired trucks and the drivers overtime work will be considered.

Their approach has been further extended in Naso et al. (2007b) taking into account even more realistic assumptions, such as increased plant capacities and dynamic truck speeds. A non-linear mathematical model has been introduced. Furthermore they propose an event-driven rescheduling approach in order to handle perturbations that might occur during the planning horizon.

Asbach et al. (to appear) developed a general mixed integer programming model which cannot be solved on instances of reasonable size. Hence a local search based approach has been implemented in order to handle real-world instances. Based on a given solution a certain number of customers and all associated deliveries are removed from the current solution. Their algorithm tries to sequentially reschedule all associated deliveries while the timing of all remaining loading and unloading operations stays unchanged.

Durbin (2003) and Hoffman and Durbin (2008) develop an optimization-based decision-support tool. They present a time-space network formulation and use minimum-cost network flow optimization techniques and tabu search to solve the problem. Their effective use of a time-space network formulation inspired us to include a integer MCNF component in our approach. Their approach is similar to the formulation presented in Schmid et al. (2008) and Schmid (2007).

The VNS as well as a cooperative hybrid approach for the problem described in this paper were proposed by Schmid et al. (2008). In contrast to the current paper the approach based on VNS was used as input for solving a formulation based on a multi-commodity network flow formulation (MCNF). For every order a number of patterns is generated. Every pattern uniquely specifies when and how unloading operations may take place, as well as which types of trucks are supposed to execute an unloading operation and their sequence to arrive at the construction site. All deliveries associated with one pattern would completely satisfy the requested demand of the corresponding order, as well as any requirements concerning special unloading equipment. Patterns however are only feasible from the orders point of view. The main goal of the MCNF is to choose one pattern per order while ensuring that the resulting truck movements are also feasible from the vehicle's point of view. Within the embedded hybrid framework VNS and MCNF alternate in trying to quickly find a good and feasible solution. Any solution which has been locally optimized by means of VNS serves as a (feasible) starting solution for MCNF. Additionally all incumbent solutions found during the process of VNS are added to the pool of patterns. Thereby the pool of patterns are enriched, yielding more alternatives and flexibility when it comes to solving the MCNF. The obtained solution quality is comparable to the one described in this paper. The solution quality for small and medium sized instances could be improved by the approaches described in the current paper.

Exact algorithms are desired when solving combinatorial optimization problems. In practice, however, only small sized problem instances may be expected to be solved to optimality in a reasonable computing time. Hence heuristic techniques are the immediate tool to quickly approach this type of problems, thus aiming to find good quality solutions (see, e.g., Glover and Kochenberger, 2003). One example of a metaheuristic is Variable Neighborhood Search (VNS), which allows for an intensive search in the solution space of the problem. VNS has been developed by Mladenović and Hansen (1997) and extended in Hansen and Mladenović (2001). It is a Local Search based improvement heuristic. In

contrast to population based approaches, VNS concentrates on one single solution only and does not incorporate any type of adaptive memory. An efficient search within the solution space is guaranteed by both diversification and intensification strategies in use. During shaking phases the incumbent becomes the new current solution, which is perturbed by means of different neighborhood structures, allowing the solution process to explore various regions of the solution space and (hopefully) to escape from any local optima.

A comprehensive overview on Very Large Neighborhood Search (VLNS) can be found in Ahuja et al. (2002). The aim of the current paper is to introduce and evaluate a new hybrid algorithm that combines the VNS based approach with an exact approach, which itself is guided by means of VLNS. The exact approach is based on a Mathematical Programming formulation that is tested to solve small instances. These experiments provide us also a measure of the quality of heuristic algorithms, like the one also presented in this paper.

The remainder of this paper is organized as follows. First we will give a detailed description to the optimization problem. Section 3.1 states the mathematical notation that is used in the paper. Section 3.2 discusses a fundamental issue of our approach: splitting each customer order into truck deliveries. The problem is formulated by a Mixed Integer Linear Programming (MILP) model in Section 3.3. Section 3.4 contributes with additional valid inequalities to strengthen the Linear Programming relaxation of the model, which are of fundamental help when a model like this is solved by a general-purpose MILP solver. Section 4 describes a new heuristic approach, which is a combination of an exact method guided by VLNS and a VNS algorithm. Computational results illustrating the limits of the exact approach, the utility of the valid inequalities, and the performance of the heuristic approach, are discussed in Section 5.

2 Problem Description

An instance of our problem is defined by a set of customer orders, a set of plants and a set of trucks. Furthermore traveling times between plants and all construction sites associated with the set of orders under consideration (in both directions) are also given.

Customer orders Each customer is located at a construction site and requires a known amount of a particular type of concrete. The requirement is called order, and it is known the day before the delivery. Orders are not supposed to be postponable to one of the following days, but need to be executed. The amount of concrete demanded per order typically exceeds the capacity of any single truck available. Hence several deliveries need to be scheduled in a row in order to completely satisfy the order of a single customer. However, also small orders are attended. Even if the capacity of a truck would permit serving several orders, trucks may only serve one order between visiting two plants. Hence, small orders imply that trucks will only be partially loaded.

Constructors typically require a constant inflow of concrete to efficiently build their structures. Therefore all single deliveries should take place just in time. As soon as one truck has finished its unloading operation, preferably the next truck should already be available and ready to start its unloading operation. Gaps between consecutive unloading operations constitute a trouble for the constructor, therefore they should be avoided to the best extent possible. Unloading operations are supposed to be non-overlapping, hence only one truck can unload at a time. Due to space limitations at construction sites, trucks are supposed to leave the construction site immediately after having finished their unloading operations. Trucks can only spent idle time at plants if necessary. The time necessary in order to unload a truck is implicitly given by the construction sites' unloading rate.

When placing an order, constructors also indicate a time window within which their demand arises and the delivery of concrete should start. Therefore a time window is assigned to every order and the requested deliveries have to be executed using the fleet of vehicles available. During the given time window the first truck should arrive at the construction site and start its unloading operation. The time window however does not relate to any other deliveries other than the first one. An early start of the first delivery before the start of the time window is not allowed. In order to guide the solution process towards acceptable solutions, a late start (i.e. after the end of the time window) will be penalized accordingly.

Plants Every plant has its predefined loading rate at which concrete can be poured into the trucks. The time necessary to load a truck is implicitly given by the plant loading rate and the truck capacity. Any setup time between loading operations of two trucks at the same plant is considered as not being significant and hence neglected.

A plant's capacity is defined by the number of trucks that can be loaded simultaneously. Based on the real-world case motivating our work, resource restrictions in terms of availability of required raw materials are not relevant at this stage, and therefore they are not considered in our problem description. In other words, the plants are considered uncapacited.

Trucks Trucks start their daily tour at their home plant. By the end of the day every truck needs to return to its home plant where it started its daily tour from. Trucks do not have to be in use every single day (i.e., a truck might remain at its home plant and do not fulfill any single loading/unloading operation). Trucks basically commute between plants and construction sites corresponding to orders that are served. Loading operations of trucks are not restricted to their home plants. A truck may also be loaded at plants other than its home plant. Due to special characteristics of concrete a truck must not serve two customers (i.e. deliveries) with the same load of concrete. Rather it has to be loaded with (fresh) concrete before going to a construction site. This is not necessarily due to perishability; rather, concrete is such a heterogeneous product

with various different kinds of recipes. Only full truck loads are considered for the delivery of concrete.

The fleet of trucks is heterogeneous. On one hand their capacity limits might be different, on the other hand they might also differ in their instrumentation. Every truck is based at a particular plant called its home plant. A truck (with loading space) can be used for the delivery of concrete only, even if they have specialized instrumentation. Trucks with specialized instrumentation might only assist during unloading operations, providing a pump or a conveyor belt. There also exist hybrid vehicles which are used for delivery of concrete and providing the required equipment during unloading operations. Note that the problem itself cannot be decomposed into the scheduling of trucks transporting concrete only, and the scheduling of trucks equipped with unloading instrumentation. This is due to the fact that some trucks can also be employed for executing both types of tasks.

Usually trucks unload the concrete directly (without using any specialized instrumentation) in the construction site. If specialized instrumentation is required for unloading operations, they are disclosed at the same time that the order is placed. In these cases the first truck to arrive at the corresponding construction site needs to be equipped accordingly. These trucks first unload their own load of concrete when they have loading space. Afterwards they need to stay at the construction site and assist later arriving trucks in performing their unloading operations respectively. The first truck to arrive is only allowed to leave the construction site when the total demand of the order has been fully satisfied and the last truck scheduled for a delivery has finished its unloading operation. It is not possible to replace the truck assisting other trucks with their unloading operation. Any displacement would be impractical, as it would take too much time, and hence disturb the unloading process.

Trucks are considered suitable for executing a particular delivery if the following conditions apply: For orders requiring special instrumentation the first truck to be scheduled needs to be equipped accordingly by all means. The chosen truck must not be scheduled twice for the very same order. Trucks equipped with special instrumentation but no loading capacity may only be foreseen to arrive first at a construction site (of an order requiring their particular type of instrumentation) and assist other trucks with their unloading operations. In all other cases any truck can be scheduled for executing a particular delivery.

Raw material at plants is not considered as a restriction and any truck is not allowed to serve consecutively two customers without passing in-between by a plant. Although at first sight a reader may think that our problem is a particular case of what in the literature would be called *Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows* (see Schmid (2007)), the two hypothesis enforce a fundamental difference. More precisely, although in our problem each order will probably need to be split into several deliveries (whose number is unknown before the whole problem is solved), the problem does not have the implicit difficulty that the so-called *Split-Delivery Capacitated Vehicle Routing Problem* (SDVRP) faces. See, e.g., Archetti and

Speranza (2007) for a survey on SDVRP works. One should observe that in a SDVRP one needs to appropriately split the load of a vehicle along the sequence of its customers, so that each customer demand is satisfied. The difficulty of splitting the load of the vehicle into several customers is not present in our problem. On the other hand, this observation does not imply that our problem is simpler than the classical SDVRP because we also have to deal with other additional features of the problem (i.e., vehicles with specialized instrumentations, time windows, etc.).

To conclude the problem description, there are two criteria to be considered in the objective function. First, the total delivery time wants to be minimized. Second, starting the first delivery before the beginning of the time window is not permitted, and any late start will be penalized. All consecutive deliveries need to start afterwards and are supposed to be non-overlapping. Any gaps between consecutive unloading operations are not desired and will be penalized accordingly. With appropriated parameters, these aims may be measured in a single objective function to be minimized.

3 Mathematical Formulation

This section presents a Mixed Integer Linear Programming model of the above problem. It is used, after fixing appropriated variables, to drive a metaheuristic approach, thus leading to the hybrid approach proposed in this paper to find good feasible solutions to the problem. Before presenting the model, we first set up the notation that is used. A fundamental issue of our formulation is splitting each order into deliveries, each one to be performed by a truck. After presenting the model, we also propose additional valid inequalities, redundant for integer solutions but useful to cut-off fractional solutions of the linear programming relaxation.

3.1 Notation

This section contains all the notation needed for a precise formulation of the problem. Capital letters are used to indicate input parameters of an instance, while non-capital letters will be left for mathematical variables and indices (introduced in Section 3.3). Calligraphic letters are used for sets, and non-calligraphic letters are left for single numbers.

The set of orders is denoted by $\mathcal{O}$ and set of trucks is denoted by $\mathcal{K}$. The subset of orders requiring specialized instrumentation to unload trucks is denoted by $\mathcal{O}'$ and the subset of trucks with specialized instrumentation is denoted by $\mathcal{K}'$. The type of instrumentation required by an order $o \in \mathcal{O}'$ is denoted by I_o and the type of instrumentation available in truck $k \in \mathcal{K}'$ is denoted by I^k. The total amount of RMC associated with a customer order o is denoted by Q_o and the maximum amount (capacity) that truck k can transport is denoted by Q^k. For each $k \in \mathcal{K}$, let p_k be the home plant associated with truck k. For each order $o \in \mathcal{O}$, let E_o be the earliest time that the first truck should start

unloading at the construction site, and let F_o be the latest time that the last truck can start unloading in the construction site. The time interval $[E_o, F_o]$ is called the time window associated with order o. Driving truck k, let $T^k_{o_1,o_2}$ be the known time to go from the construction site of order o_1 to the construction site of order o_2, while also being loaded at the closest plant enroute, from which the construction site of order o_2 can be reached in less then two hours. Let $T_{p,o}$ be the known time to go from the location site of plant p to the construction site of order o, and $T_{o,p}$ be the known time to go from the construction site of order o to the location site of plant p. The time to unload truck k at the construction site o is denoted by U^k_o, depends on the truck and the construction site, but not on the amount of unloaded concrete. Also $T^k_{o_1,o_2}$ does not depend on the amount of concrete load in a plant for (partially) serving order o_2. In other words, $T^k_{o_1,o_2}$ is independent on the delivery that truck k will execute in order o_2.

Since the concrete request of an order may need more than one truck, each order o is associated with a set of potential deliveries $\mathcal{D}_o$. Each element of this set represents only the index in the sequence of trucks visiting the construction site of order o. Thus, for example, the first element of $\mathcal{D}_o$ is denoted by 1, and it must be associated with a truck with the specialized instrumentation required by o if $o \in \mathcal{O}'$. Indeed, delivery 1 of each order is a mandatory delivery that must not arrive before time E_o. Depending on the loading space of the truck associated to delivery 1 of order o, and the request Q_o, other deliveries may be necessary. Each element $d \in \mathcal{D}_o$ is sometime referred to as the d-th potential delivery of order o. Section 3.2 computes bounds to approximate how many deliveries an order may need. The size $|\mathcal{D}_o|$ is just an upper bound on this number, and it has a direct impact on the size of the mathematical model given in Section 3.3.

3.2 Splitting orders into deliveries

To completely satisfy an order usually several full truck loads are necessary, as the demanded quantity typically exceeds the capacity of any single truck. It is unknown beforehand how many deliveries are needed to satisfy any order. The difficulty is due to the heterogeneity of the truck fleet, the existence of different orders to be served, and to the timing of the deliveries. Appropriate bounding strategies are required in order to overcome this issue and help the optimization process by using tight bounds.

The minimum number of deliveries required can easily be calculated by assuming that all unloading operations are executed by the largest truck available. Therefore the minimum number of deliveries needed to satisfy order o with a demand of Q_o is given by $\lceil Q_o / \max\{Q^k : k \in \mathcal{K}\} \rceil$. In case the corresponding order o requires special unloading equipment I_o one also needs to take into account the largest truck being equipped accordingly such that it could execute the first unloading operation. Assume that the largest truck equipped accordingly executes the first unloading operation and all remaining deliveries are served by one of the largest trucks available at all. In this case at least

$\lceil \max(0, Q_o - \max\{Q^k : I^k = I_o\}) / \max\{Q^k : k \in \mathcal{K}\} + 1 \rceil$ deliveries are needed.

A bound for the maximum number of deliveries necessary to transport the concrete required by order o can be computed by assuming that all deliveries are executed by trucks with the smallest capacity available. Only trucks that are equipped with loading space are considered in this case. The maximum number of deliveries necessary is given by $\lceil Q_o / \min\{Q^k : k \in \mathcal{K}, Q^k > 0\} \rceil$. In case an order requires special unloading equipment the maximum number of trucks serving this order is increased by one, taking into account a truck with special equipment that might need to arrive first, possibly with no loading capacity.

3.3 Mathematical Model

We now present a Mixed Integer Linear Programming formulation for the problem, and to this end we start by introducing the mathematical variables.

All truck movements among customers are modeled in terms of three different types of 0-1 variables. Variable $x^k_{o_1,d_1;o_2,d_2}$ is equal to value 1 when truck k serves delivery d_2 of order o_2 immediately after serving delivery d_1 of order o_1. As stated in Section 2, when going from o_1 to o_2 truck k drives from the construction site associated with order o_1 to a plant p where it loads concrete, and then it drives to the construction site associated with order o_2. The choice of the plant p is not left as an option to the model itself, but it is automatically chosen depending on these two construction sites. Between serving two orders o_1 and o_2 the truck visits the closest plant enroute, while still ensuring the RMC is transported for at most two hours. This make sense on the hypothesis that plants are uncapacitated (i.e., no limit to the type and amount of concrete that a plant can provide every day).

To ensure that all trucks leave and return to their home plants, we also consider two sets of 0-1 variables. Variable $u^k_{o,d}$ is 1 if and only if the first delivery to be performed by truck k along the planning horizon (day) refers to executing delivery d of order o. Truck k starts its route from its home plant, where it is loaded with the concrete for d. Analogously $v^k_{o,d}$ refers to the very last movement of truck k per day, and it is 1 if and only if delivery d of order o is its last assignment before returning to its home plant.

Another 0-1 variable $y^k_{o,d}$ indicates whether a certain truck k executes delivery d of order o. Furthermore a 0-1 variable $z_{o,d}$ serves as an indicator whether or not a certain delivery d associated with order o is supposed to be executed. The variable $z_{o,d}$ is equal to 1 if the d-th potential delivery associated with order o is executed, and 0 otherwise. Recall that $d \in \mathcal{D}_o$ is simply an index, so $d = 1$ represents the first delivery, that must be executed, and $d = |\mathcal{D}_o|$ represents a true or dummy delivery depending if it is executed or not respectively.

To control the time window requirements at each order, we introduce two sets of continuous (non-negative) variables. Variable $a_{o,d}$ ($b_{o,d}$) refers to the start (end) of the unloading operation associated with delivery d of order o. Remember that when $o \in \mathcal{O}'$, the first delivery $d = 1$ may only be assigned to a truck $k \in \mathcal{K}'$, which will stay at the construction site (in case $Q^k < Q_o$) to assist other trucks. Thus, $b_{o,d}$ coincides with the time when a truck leaves o if

and only if $o \notin \mathcal{O}'$ or $d > 1$ or $Q^k \geq Q_o$. On the other hand, $a_{o,d}$ coincides with the time when the truck assigned to this particular delivery starts its unloading operation at the construction site corresponding to order o.

In addition, another continuous variable l_o captures any delayed start of the first delivery associated with order o. In case the first delivery starts after the end of the given time window interval $[E_o, F_o]$, the decision variable l_o evaluates the length of the resulting gap. In case the first delivery starts within the time window it will be equal to 0. Remember that starting the very first delivery before the earlier time in the given time window is not permitted.

We are now ready to introduce the mathematical formulation. The main part of the objective function is to minimize the total time traveled by trucks. Additionally gaps between consecutive deliveries, or starting the first delivery of any order o after the latest time F_o of the associated time window, are not desirable and need to be penalized. To combine these features in a single objective function, we make use of the parameter β for penalizing gaps between consecutive deliveries (as also done in Schmid et al. (2008)). Then the objective function is as follows:

$$\begin{aligned} \min \sum_{\substack{o_1,o_2 \in \mathcal{O} \\ d_1 \in \mathcal{D}_{o_1} \\ d_2 \in \mathcal{D}_{o_2} \\ k \in \mathcal{K}}} T^k_{o_1,o_2} \cdot x^k_{o_1,d_1;o_2,d_2} + \sum_{\substack{o \in \mathcal{O} \\ d \in \mathcal{D}_o \\ k \in \mathcal{K}}} T_{p_k,o} \cdot u^k_{o,d} + \sum_{\substack{o \in \mathcal{O} \\ d \in \mathcal{D}_o \\ k \in \mathcal{K}}} T_{o,p_k} \cdot v^k_{o,d} \\ + \beta \sum_{o \in \mathcal{O}} \big(l_o + \sum_{d \in \mathcal{D}_o : d > 1} (a_{o,d} - b_{o,d-1}) \big). \end{aligned} \tag{1}$$

To reduce symmetric solutions, the very first delivery associated with any order needs to be executed by all means:

$$z_{o,1} = 1 \qquad \forall o \in \mathcal{O}. \tag{2}$$

Only consecutive deliveries may need to be executed:

$$z_{o,d} \leq z_{o,d-1} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o : d > 1. \tag{3}$$

When delivery d of order o is decided to be executed, a truck k must be assigned to it:

$$\sum_{k \in \mathcal{K}} y^k_{o,d} = z_{o,d} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o : o \notin \mathcal{O}' \text{ or } d > 1 \tag{4}$$

and

$$\sum_{k \in \mathcal{K}' : I^k = I_o} y^k_{o,1} = 1 \qquad \forall o \in \mathcal{O}'. \tag{5}$$

If truck k is assigned to the execution of delivery d associated with order o, the truck needs to go to the construction site of o:

$$y^k_{o,d} = u^k_{o,d} + \sum_{\substack{o_1 \in \mathcal{O} \setminus \{o\} \\ d_1 \in \mathcal{D}_o}} x^k_{o_1,d_1;o,d} + \sum_{d_1 \in \mathcal{D}_o : d_1 < d} x^k_{o,d_1;o,d} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o, k \in \mathcal{K}. \tag{6}$$

Analogously, after executing a delivery, a truck must go somewhere else:

$$y^k_{o,d} = v^k_{o,d} + \sum_{\substack{o_2 \in \mathcal{O}\setminus\{o\} \\ d_2 \in \mathcal{D}_o}} x^k_{o,d;o_2,d_2} + \sum_{d_2 \in \mathcal{D}_o : d_2 > d} x^k_{o,d;o,d_2} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o, k \in \mathcal{K}. \tag{7}$$

A truck is allowed to leave its home plant (and execute its very first unloading operation per day) at most once:

$$\sum_{\substack{o \in \mathcal{O} \\ d \in \mathcal{D}_o}} u^k_{o,d} \leq 1 \qquad \forall k \in \mathcal{K} \tag{8}$$

In case a truck leaves its home plant it also needs to return there by the end of the day:

$$\sum_{\substack{o \in \mathcal{O} \\ d \in \mathcal{D}_o}} u^k_{o,d} = \sum_{\substack{o \in \mathcal{O} \\ d \in \mathcal{D}_o}} v^k_{o,d} \qquad \forall k \in \mathcal{K} \tag{9}$$

Usually trucks execute more than one single delivery per day. To ensure feasibility, the time difference between two consecutive unloading operations assigned to the same truck needs to be big enough to allow the truck driving to the closest plant and being loaded there. When the two orders are different then:

$$a_{o_2,d_2} \geq b_{o_1,d_1} + T^k_{o_1,o_2} \cdot x^k_{o_1,d_1;o_2,d_2} - M \cdot (1 - x^k_{o_1,d_1;o_2,d_2}) \tag{10}$$

$$\forall o_1 \in \mathcal{O}, o_2 \in \mathcal{O}\setminus\{o_1\}, d_1 \in \mathcal{D}_{o_1}, d_2 \in \mathcal{D}_{o_2}, k \in \mathcal{K} : o_1 \notin \mathcal{O}' \text{ or } d_1 > 1 \text{ or } Q^k \geq Q_{o_1}$$

and

$$a_{o_2,d_2} \geq b_{o_1,|\mathcal{D}_{o_1}|} + T^k_{o_1,o_2} \cdot x^k_{o_1,1;o_2,d_2} - M \cdot (1 - x^k_{o_1,1;o_2,d_2}) \tag{11}$$

$$\forall o_1 \in \mathcal{O}', o_2 \in \mathcal{O} \setminus \{o_1\}, d_2 \in \mathcal{D}_{o_2}, k \in \mathcal{K}' : Q^k < Q_{o_1}, I^k = I_{o_1}$$

where M is a big value defined as the difference between the latest possible time when truck k could leave the construction site associated with order o_1 after having executed delivery d_1 and the earliest possible start of delivery d_2 for order o_2. Note that the second inequalities impose that the first truck arriving at the construction site associated with order $o_1 \in \mathcal{O}'$ requiring special instrumentation must wait until the last delivery of such order has been executed. The big value M is used to deactivate the inequality when $x^k_{o_1,1;o_2,d_2} = 0$.

When the two orders are identical then the previous requirements simplify as follows:

$$a_{o,d_2} \geq b_{o,d_1} + T^k_{o,o} \cdot x^k_{o,d_1;o,d_2} \tag{12}$$

$$\forall o \in \mathcal{O}, k \in \mathcal{K}, d_1, d_2 \in \mathcal{D}_o : d_1 < d_2.$$

Although redundant when the previous constraints are present, we also consider the following trivial inequalities for clarity. Two trucks cannot be unloaded at the same time in a construction site:

$$a_{o,d} \geq b_{o,d-1} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o : d > 1. \tag{13}$$

The first delivery of any order has to start after the begin of the given time window:

$$a_{o,1} \geq E_o \qquad \forall o \in \mathcal{O}. \tag{14}$$

A late start of the first delivery of order o needs to be penalized in the objective function accordingly. The first delivery of any order is considered as starting late in case the unloading operation is initiated after the end of the corresponding time window:

$$l_o \geq a_{o,1} - F_o \qquad \forall o \in \mathcal{O}. \tag{15}$$

To ensure that the required demand is totally delivered, the cumulative capacity of all trucks serving an order may not be smaller than such demand:

$$\sum_{\substack{d \in \mathcal{D}_o \\ k \in \mathcal{K}}} Q^k \cdot y^k_{o,d} \geq Q_o \qquad \forall o \in \mathcal{O}. \tag{16}$$

The time to execute any delivery is determined as a constant dependent on the truck and the construction site:

$$b_{o,d} - a_{o,d} \geq U^k_o \cdot y^k_{o,d} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o, k \in \mathcal{K} \tag{17}$$

$$b_{o,d} - a_{o,d} \leq U^k_o + M'(1 - y^k_{o,d}) \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o, k \in \mathcal{K} \tag{18}$$

where M' is a big value defined as $\max\{U^k_o : k \in \mathcal{K}\} - U^k_o$.

Finally, for completeness, the model also needs:

$$x^k_{o_1,d_1;o_2,d_2}\ ,\ y^k_{o,d}\ ,\ z_{o,d}\ ,\ u^k_{o,d}\ ,\ v^k_{o,d} \in \{0,1\} \tag{19}$$

$$a_{o,d}\ ,\ b_{o,d}\ ,\ l_o \geq 0 \tag{20}$$

for all indices.

The complexity of the problem justifies the size of this formulation. Indeed, the size is mainly due to the fact that an order o may be served by different deliveries by introducing the set $\mathcal{D}_o$, thus $|\mathcal{D}_o|$ affects directly the number of variables and constraints. If m is an upper bound of $|\mathcal{D}_o|$, then both the number of variables and the number of constraints in model (1)–(20) are magnitudes of the order of $m^2|\mathcal{O}|^2|\mathcal{K}|$. An advantage of the formulation, however, is that it is in a sense "compact", as sophisticated variable or constraint generation (i.e., neither pricing nor separation) procedures are not required. Still, the numbers are big even for small instances, and therefore basic ideas are fundamental to help a general-purpose solver with this model. To illustrate some of these basic ideas, for example, not all inequalities in (10)–(11) should be given all-together; instead, the solver should be called iteratively, and only violated requirements should be added at each iteration. To keep the model as small as possible, it is also fundamental to remove slack constrains that can later be added if they turn out to be violated again. The whole approach is called *Branch-and-Cut*, and we refer the reader to (for example) the manual of Xpress (Dash, 2007) for further technical details. A final remark, however, is that our main reason to provide a mathematical model for our problem is not to have an exact method for the problem itself, but rather to have a tool which (after fixing some variables) will drive a metaheuristic approach to find high-quality solutions.

3.4 Strengthening the LP relaxation

A general-purpose solver working on model (1)–(20) relies on its linear-programming (LP) relaxation. Therefore, to help a general-purpose solver with the best possible lower bound when solving LP relaxations, we now address additional inequalities. All the following inequalities are valid inequalities, redundant for integer solutions of model (1)–(20), and designed to cut-off fractional solutions observed when solving the benchmark instances from the application motivating our research. In other words, the inequalities listed in the section were observed to be useful in our implementation. Section 5 shows computational evidences to support this claim.

If the first vehicle cannot satisfy the order demand, a second delivery is necessary:

$$z_{o,2} \geq y^k_{o,1} \qquad \forall o \in \mathcal{O}, k \in \mathcal{K} : Q^k < Q_o. \tag{21}$$

Although these inequalities can be easily extended to the d-th delivery with $d > 2$, we did not find advantages of using them in our benchmark instances.

In case truck k returns to its home plant after having executed delivery d_1 of order o, then truck k cannot be foreseen for executing any later deliveries d_2 ($d_1 < d_2$) for order o:

$$y^k_{o,d_2} + \sum_{\substack{d_1 \in \mathcal{D}_o \\ d_1 < d_2}} v^k_{o,d_1} \leq 1 \qquad \forall o \in \mathcal{O}, k \in \mathcal{K}, d_2 \in \mathcal{D}_o : d_2 > 1. \tag{22}$$

Trucks that do not leave their home plant cannot be assigned to execute any single delivery:

$$\sum_{\substack{o \in \mathcal{O} \\ d \in \mathcal{D}_o}} u^k_{o,d} \geq \sum_{\substack{o_1, o_2 \in \mathcal{O} \\ d_1 \in \mathcal{D}_{o_1} \\ d_2 \in \mathcal{D}_{o_2}}} x^k_{o_1,d_1;o_2,d_2} \qquad \forall k \in \mathcal{K}. \tag{23}$$

A truck executing the first delivery of an order requiring specialized instrumentation cannot be scheduled for any other delivery within the same order:

$$y^k_{o,1} + y^k_{o,d} \leq 1 \qquad \forall o \in \mathcal{O}', k \in \mathcal{K}, d \in \mathcal{D}_o : d > 1. \tag{24}$$

In case a truck is supposed to execute a delivery, the truck needs to leave its home plant sometime before.

$$y^k_{o,d} \leq \sum_{\substack{o_2 \in \mathcal{O} \\ d_2 \in \mathcal{D}_{o_2}}} u^k_{o_2,d_2} + \sum_{d_2 \leq d} u^k_{o,d_2} \qquad \forall o \in \mathcal{O}, d \in \mathcal{D}_o, k \in \mathcal{K}. \tag{25}$$

For a given order o, if a truck k executes two deliveries d_1 and d_2 ($d_1 < d_2$) then the time between the end of the previous delivery (b_{o,d_1}) and the start of the latter one (a_{o,d_2}) needs to be sufficient such that the corresponding truck k can drive to the closest plant for being loaded there:

$$a_{o,d_2} \geq b_{o,d_1} \; + \; T^k_{o,o} \cdot \; (y^k_{o,d_1} + y^k_{o,d_2} - 1) \\ \forall o \in \mathcal{O}, k \in \mathcal{K}, d_1, d_2 \in \mathcal{D}_o : d_1 < d_2. \tag{26}$$

Only deliveries that are actually executed will take time:

$$b_{o,d} \geq a_{o,d} + \sum_{k \in \mathcal{K}} U_o^k \cdot \ y_{o,d}^k \quad \forall o \in \mathcal{O}, d \in \mathcal{D}_o. \tag{27}$$

The integer solutions of model (1)–(20) satisfy:

$$\sum_{\substack{o_1 \in \mathcal{S} \\ d_1 \in \mathcal{D}_{o_1} \\ o_2 \notin \mathcal{S} \\ d_2 \in \mathcal{D}_{o_2}}} x_{o_1,d_1,o_2,d_2}^k + \sum_{\substack{o_1 \in \mathcal{S} \\ d_1 \in \mathcal{D}_{o_1}}} v_{o_1,d_1}^k \geq y_{o,d}^k \tag{28}$$

for all $\mathcal{S} \subset \mathcal{O}$, $o \in \mathcal{S}$, $d \in \mathcal{D}_o$ and $k \in \mathcal{K}$. These type of inequalities are standard in routing problems and requires a sophisticated separation procedure based on solving min-cost flow problems. They enforce that a vehicle cannot be only visiting a subset of customers without ever returning to its home plant. In our case, constraints (10)–(11) ensures that the route of a truck connects its home plant with the construction site of the deliveries it must perform, thus imposing the above requirement. On fractional solutions some of the above inequalities may be violated, and in our experiments we found it useful to generate the ones with $|\mathcal{S}| = 2$. In addition we also generate the following inequalities, which are supported by the same argument:

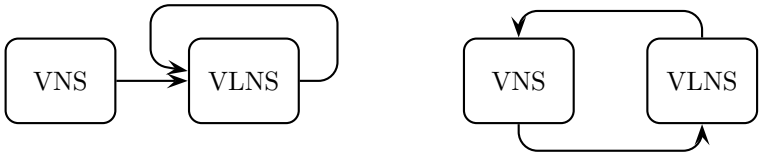
$$x_{o_1,d_1;o_2,d_2}^k + x_{o_2,d_2;o_1,d_1}^k \leq y_{o_1,d_1}^k \qquad \forall o_1, o_2 \in \mathcal{O}, d_1 \in \mathcal{D}_{o_1}, d_2 \in \mathcal{D}_{o_2}, k \in \mathcal{K}. \tag{29}$$

4 Solution Procedure

A hybrid solution procedure has been developed for solving the problem stated in Section 2. After having found an initial solution using VNS, the model (1)–(20) given in Section 3.3 is solved repeatedly using VLNS. In each iteration a general-purpose MILP solver is used to solve the formulation. Most variables are value fixed (i.e., removed). Some decision variables however remain unfixed and the optimization process starts over again. The choice of which decision variables are unfixed is left to a variable-fixing approach guided by means of VLNS. Optionally VNS can also be used to further improve any solution found after executing VLNS. VNS and its components are described in Section 4.1. The framework based on VLNS is discussed in more detail in Section 4.2. A graphical representation of the embedded framework is shown in Figures 1(a) and 1(b) respectively. Figure 1(a), referred to as the integrative hybrid approach, iteratively solves mixed integer problems using VLNS. Figure 1(b) shows the extended version, where VNS is used between the shaking steps of VLNS.

4.1 Variable Neighborhood Search

This section is organized as follows. First we present the basic implementation for the VNS. Afterwards design issues such as the initialization, the shaking



(a) Integrative Hybrid (b) Collaborative Hybrid

Figure 1: Different Hybrid Approaches

phase, the embedded Local Search (LS) operators for the iterative improvement step, the evaluation and acceptance scheme in the context of RMC delivery are explained. In the shaking phase or iterative improvement step the used operators are applied to the sequence of trucks to serve a given order. The exact timing, i.e. when all unloading operations start, is redetermined afterwards and the feasibility from the trucks point of view is ensured. A solution consists of one sequence per order. During the process of VNS these sequences are modified in the shaking phase and locally optimized in the iterative improvement step.

Any sequence of trucks obtained after the shaking process needs to be feasible from the orders point of view. That implies that any additional requirements concerning special equipment for unloading have to be considered and the accumulated capacity of all trucks scheduled is able to satisfy the demand of the corresponding order. If the first truck within a sequence is exchanged and the corresponding order requested special unloading equipment, only trucks bringing along equivalent instrumentation will be considered when selecting the new first truck within this sequence.

Shaking operators only have an effect upon the sequence of trucks to be scheduled to satisfy an order. Note that the exact timing of all unloading operations as well as feasibility from the trucks point of view are ignored. However, they do not yet constitute a feasible solution from the trucks' point of view, as the exact timing of all unloading operations, movements of trucks between plants and construction sites as well as their loading operations necessary in between, are not considered. In order to evaluate the solution after the shaking phase an evaluation function is used. The aim of the evaluation function is to determine the timing of all unloading operations to be performed, based on a sequence of trucks given per order. The resulting solution will not only be feasible from the orders' point of view, but also from the trucks point of view, i.e. giving enough time between consecutive unloading operations, to drive to the nearest plant and being loaded there.

4.1.1 Basic Implementation

During the *shaking phase* several neighborhoods are used to explore the solution space more thoroughly and avoid potentially being trapped in local optima. Partial sequences of trucks designated to deliver concrete to a specific construc-

tion site are inverted and exchanged. The embedded LS operator is used to locally optimize every solution obtained after the shaking step (in the iterative improvement step).

A sketch of the basic steps of the implemented VNS can be found in Algorithm 1. A neighboring solution x' will be generated at random from neighborhood $N_\kappa(x)$. Starting from x' the current solution is locally optimized in the iterative improvement step. The resulting solution is denoted by x''. If this solution improves the incumbent solution x, the current solution x'' will become the new incumbent solution x and shaking continues with the first neighborhood. If no new incumbent solution could be obtained, the search continues within the next neighborhood $\kappa+1$. Ascending moves, i.e. accepting deteriorating solutions, are currently not employed. The algorithm however could easily be adapted to incorporate this feature as well.

Algorithm 1 `Basic Steps of VNS`(x)

```
while stopping criterion not met do                    ▷ Time, Iterations
    κ ← 1
    while κ ≤ κ_max do
        x' ← Shaking(x, κ)
        x'' ← Iterative Improvement(x')
        if accept(x'') then                            ▷ move or move not?
            x ← x''
            κ ← 1                  ▷ continue with first neighborhood structure
        else
            κ ← κ + 1              ▷ continue with next neighborhood structure
        end if
    end while
end while
```

4.1.2 Design Issues

Initialization The VNS itself consists of improvement steps only. Hence an initial solution might needs to be constructed before the search process can be started for the very first time. If no feasible solution is available it can easily be constructed. One sequence is generated randomly per order. The resulting sequence needs to be feasible from the orders point of view, i.e. the total quantity delivered has to be sufficient in order to satisfy the demand, if special instrumentation is required the first truck to be scheduled needs to be equipped accordingly and must not appear twice within the very same sequence. Trucks with special instrumentation but no loading capacity may only be scheduled as first deliveries for orders requiring their equipment respectively. The selection probability for all remaining trucks to be inserted in a sequence is directly proportional to their loading capacity. Sequences will be constructed sequentially until the total demand can be met, one order at a time.

Table 1: Set of Neighborhood Structures (standalone VNS)

κ	Shaking Operator	max number of sequences changed
1	`ReplaceByUnused`	1
2	`ReplaceByAny`	1
3	`ReplaceByUnused`	2
4	`ReplaceByAny`	2
5	`ReplaceByUnused`	3
6	`ReplaceByAny`	3

Shaking Phase In order to explore the solution space - in terms of potential sequences - more thoroughly, two shaking operators, resulting in six neighborhood structures have been implemented. The first shaking operator replaces partial sequences of trucks within a sequence by trucks *not used* so far in any other sequence of the current solution. When selecting the trucks there is a bias towards trucks with larger capacities and whose home plant is located closer to the construction site associated with the corresponding order. The second shaking operator works almost the same, but this time the new trucks to be inserted are no longer limited to those not in use according to the current solution. Rather, new trucks are selected randomly among all trucks available.

Neighborhood structures N_κ ($\kappa = 1, 3, 5$) relate to the first shaking operator and involve changes of sequences corresponding to up to $(\kappa + 1)/2$ orders; neighborhood structures N_κ ($\kappa = 2, 4, 6$) relate to the second shaking operator and involve changes of up to $\kappa/2$ sequences. An overview on the set of neighborhood structures is given in Table 1. The position and the length of the partial sequence to be exchanged are determined randomly. The loss of capacity associated with the vehicles being removed needs to be compensated. Hence new vehicles will be inserted into the sequence until the demand of the corresponding order can be satisfied again. Furthermore, if the first vehicle of the sequence of any given order is exchanged and the order requires special unloading equipment, only vehicles equipped accordingly will be considered when selecting the replacement for the first vehicle.

Any neighborhood operator should perturb the current solution x which might lead to a new best solution x'' after the iterative improvement. The main idea of any shaking operator is to *replace* the allocation of some trucks within sequences belonging to up to $(\kappa+1)/2$ (or $\kappa/2$ respectively) orders resulting in a different sequence of trucks. The position and the length of the partial sequence to be exchanged are determined randomly.

The first shaking operator `ReplaceByUnused` replaces trucks in sequences corresponding to up to $(\kappa+1)/2$ orders. Trucks are replaced by other trucks not in use far within the current solution. When selecting new trucks for a sequence belonging to order o, there is a bias towards choosing large trucks whose home

plant is located close to the construction site. The shaking operator for order o can only be executed if there exists a suitable truck for the current delivery. First the plant will be chosen. The selection probability for a plant to be chosen is indirectly proportional to the travel time from the construction site to the corresponding plant and the return trip. The selection probability $\mathcal{P}_p$ for plant p is given by

$$\mathcal{P}_p = \frac{T_{p,o} + T_{o,p}}{\sum_{l \in P} T_{l,o} + T_{o,l}}. \tag{30}$$

After choosing a plant p a truck will be selected. When selecting the truck there is a bias towards choosing larger trucks, which might help to keep the number of trucks needed (and resulting travel times) low. For all suitable trucks located at plant p the conditional selection probability $\mathcal{P}_k$ for choosing truck k is given by

$$\mathcal{P}_k = \frac{Q^k}{\sum_{l \in K'_p} Q^l}. \tag{31}$$

, where K'_p contains all trucks located at plant p which are suitable for the delivery under consideration and currently not yet in use. In case no suitable truck located at plant p can be found a new plant will be determined and the procedure starts over again.

The second shaking operator `ReplaceByAny`, unlike the first one, tries to replace partial sequences of trucks by other trucks chosen randomly. This time the selection process is no longer restricted to trucks that are not in use so far within the current solution. The selection of new trucks again is biased towards trucks based in preferably close home plants (with respect to travel times) to the corresponding construction site and trucks with larger capacity. The selection probability for choosing the plants is also calculated according to Equation 30. The selection probability $\mathcal{P}_k$ for truck k is given by

$$\mathcal{P}_k = \frac{Q^k}{\sum_{l \in K''_p} Q^l}, \tag{32}$$

where K''_p refers to the set of suitable trucks located at plant p.

Evaluation After applying shaking operators or LS operators to any solution the solution needs to be reevaluated. This is done through heuristically determining the start of any single unloading operation. The resulting solution (consisting of sequences plus the exact timing for the start of all unloading operations) has to be feasible from the orders' point of view (i.e. unloading operations are non-overlapping). Additionally they also have to be feasible from the trucks' point of view. There has to be enough time, after a truck is allowed to leave a construction site, to drive to a plant, being loaded there and get to the next construction site just in time for its next unloading operation.

Two approaches based on forward and backward termination respectively have been developed in order to accomplish the evaluation of any solution.

Both approaches are executed consecutively, starting with forward termination. Forward termination tries to schedule every operation as early as possible while still taking into account the availability of all resources (i.e. the trucks to be scheduled). The earliest possible start time for the first unloading operation per order is determined by the beginning of the associated time window. The result obtained then serves as input for backward termination, where every unloading operation is scheduled as late as possible. The best solution obtained is accepted. These procedures have been inspired by the Critical Path Method (CPM) used in activity planning (see Moder et al., 1983).

However some modifications had to be implemented with respect to trucks providing special unloading equipment. Taking into account the fact that these might need to stay longer and assist later arriving trucks is essential, as deadlock situations might arise. In order to avoid them, the scheduling of certain unloading operations has to be delayed artificially. The resulting schedule can be improved by applying backward termination afterwards. However finding an improvement is not guaranteed, therefore the best solution obtained by any of the two procedures will be accepted.

When applying forward termination all orders, starting from the very first delivery, are scheduled in a consecutive manner. All deliveries are scheduled as early as possible. Only one delivery may be scheduled at a time. The first delivery of every order cannot start before the beginning of associated time window. Ties are broken arbitrarily. In order to avoid deadlock situations trucks, assisting during unloading operations, have to be blocked temporarily. The blocking is removed as soon as the construction of the entire pattern is completed.

In case we are about the schedule the first truck of an order requiring special equipment we need to check first whether order o is blocked (i.e. delayed) temporarily because of any other order. The reason for blocking orders at all is because we want to avoid deadlock situations. Deadlock-alike situations might arise where schedules cannot be fully determined. Trucks might not be released because they are waiting for each other.

Backward termination follows a similar principle. Starting from the last unloading operation per order all unloading operations are tried to be scheduled as late as possible. The starting time of the last unloading operation per order is adopted from the solution just obtained from forward termination. Everything is tried to be scheduled as late as possible. The end of the last unloading operation is given by the solution just obtained after applying forward termination. The timing of the last delivery of an order requiring special instrumentation cannot be determined unless both trucks (the first and the last truck) are available. Again, analogous to what had to be done during forward termination, orders might need to be blocked. The start of the last delivery of an order requiring special instrumentation cannot be determined if there is another order o' also requiring special instrumentation, where the timing of some deliveries (but not all yet) already has been determined and the truck under consideration still needs to be scheduled.

After backward termination it needs to be verified that restrictions concerning time windows are still observed. For any order o it is not permitted to initiate the first unloading operation before the start of the associated time window. However, such a behavior is not guaranteed after having applied backward termination. In case some orders are going to be delivered too early all timings will be shifted accordingly, making sure that all deliveries of any order o start after the start of the time window associated with the corresponding order o. This kind of shift is necessary to guarantee feasible solutions in terms of the time windows.

The resulting solution however might become worse. By shifting all timings back accordingly, some orders now start to be served after the end of the time window associated with it. Feasibility of the solution remains unchanged. The quality however might decrease, as starting too late will be penalized accordingly. Therefore at the end the previous solution found after having executed forward termination is compared with the one just obtained after having executed backward termination. The best solution obtained will be chosen and is used for evaluating the corresponding pattern.

Iterative Improvement For the iterative improvement step (see Hoos and Stützle, 2004) three different LS operators have been implemented in order to further explore the solution space. We introduced the (`Shrink`) operator, whereas the remaining two, `IntraPatternMove` and `InterPatternSwap`, are inspired by well known LS operators to be found in the vehicle routing literature. See Gendreau et al. (1997) and Kindervater and Savelsbergh (1997) for a more detailed discussion of LS operators for vehicle routing problems.

All operators are executed based on first improvement. `IntraPatternMove`, the very first operator tries to remove any single delivery within a sequence of trucks of a given order and inserts it again at any other possible position. The second operator, `InterPatternSwap`, exchanges two deliveries associated with sequences of two different orders. The last operator `Shrink` tries to remove unnecessary trucks from any sequence as the quantity ordered might be satisfied with all remaining deliveries.

In order to execute any of the three operations implemented within the iterative improvement the timing of all deliveries will be dismissed. The timing for the start of all individual unloading operations will be determined by applying *forward* and *backward termination*. This enables us to evaluate any solution found and make sure it is also feasible from the trucks point of view.

A sketch of the iterative improvement step can be found in Algorithm 2. Any given solution, based on one sequence per order (x), is locally optimized. The first operator to be executed is `Shrink`, which is supposed to eliminate unnecessary deliveries within sequences in any x. The two following operators `IntraPatternMove` and `InterPatternSwap` are executed on a first improvement basis. `InterPatternSwap` will be performed as soon as `IntraPatternMove` can-

Algorithm 2 `Iterative Improvement`(x)

```
improved ← true
x ← Shrink(x)                                   ▷ Shrink
while improved do
    x' ← IntraPatternMove(x)                    ▷ Intra Pattern Move
    if f(x') < f(x) then                        ▷ only accept better solutions
        x ← x'
    else
        x' ← InterPatternSwap(x)                ▷ Inter Pattern Swap
        if f(x') < f(x) then
            x ← x'
        else
            improved ← false
        end if
    end if
end while
x ← Shrink(x)                                   ▷ Shrink again
```

not improve the solution x any more. To complete the iterative improvement procedure one last iteration of `Shrink` will be executed again, just to make sure no unnecessary deliveries remain within the sequence. All three operators embedded will be described in more detail below.

Shrink The aim of this operator is to make sure no unnecessary deliveries scheduled within the current solution x. Any order might still be satisfied using less than the scheduled number of deliveries. Therefore any truck scheduled within the current solution is revised. Any single delivery scheduled is tried to be omitted. A delivery can only be omitted if the capacity of the remaining trucks within the resulting sequence still satisfies the total demand of the corresponding order. Additionally all requirements concerning unloading equipment need to be satisfied. In case a delivery can be dismissed without ending up in an infeasible sequence, it will be deleted from the original schedule and the solution will be evaluated again.

Intra Pattern Move This operator moves a truck associated with a single delivery within one and the same sequence. For any order o, every truck originally scheduled to fulfill a delivery at a certain position, is tried to be inserted at a later position. If an improvement could be found the resulting solution is accepted as the new incumbent solution and the procedure starts over again. Such a move can only take place if the resulting sequence of trucks remains feasible from the orders point of view. Any requirements concerning specific unloading equipment still need to be satisfied.

Inter Pattern Swap The aim of this operator is to swap single deliveries within sequences corresponding to two different orders. Any combination

of two trucks originally scheduled for deliveries for order o_1 and o_2 are tried to be exchanged. The operator itself is executed on a first improvement basis. Any swap that leads to a new best solution will replace the incumbent solution and the procedure starts over again.

Acceptance Decision After performing any search step within iterative improvement step or shaking the newly obtained solution needs to be evaluated in order to be comparable with the incumbent solution. There is no need to deal with infeasibility as infeasible sequences are not generated at all. Any solution will only be accepted if it improves the incumbent solution. The evaluation function returns a value in terms of total traveling time plus a penalty term for gaps between consecutive unloading operators or for starting the first unloading operation too late, after the end of the given time window.

The VNS stops after a given number of iterations or a given number of iterations within which no improvement has been found. Alternatively we stop the VNS after some maximum amount of time.

Extensive testing has been executed in order to evaluate the effect of deteriorating solutions. Improvements found were not significant. In order to keep the number of parameters involved small we decided not to accept deteriorating solutions.

4.2 Very Large Neighborhood Search

As expected solving the problem based on the MIP formulation (1)–(29), using any commercial solver as a black box, is not a promising idea. Especially for larger instances it does not guarantee finding good feasible solutions - if any - in a reasonable amount of time. Therefore a strategy based on local branching (see Fischetti and Lodi, 2003) is used to assist the solver during its optimization. The whole framework in the following will be referred to as VLNS.

This section gives a short overview on the methodology used. Starting with a basic outline of the framework we focus on the problem specific knowledge in use and the resulting implicit hierarchical structure. Finally the shaking operators in use will be specified. All results obtained using this approach can be found in Section 5.3.

4.2.1 Basic Outline

The following approach tries to systematically *improve* any given feasible solution. In principle the integrated hybrid approach for solving the formulation (1)–(20) is inspired by ideas coming from VNS and Local Branching. Given any initial solution where most of the variables are fixed, a certain amount of variables are unfixed by resetting their upper and lower bound to their initial values 1 and 0 respectively. Then the problem is solved again using any general purpose solver used as a black box. Only solutions that improve the incumbent solution will be accepted.

Given any feasible solution, three shaking operators resulting in nine different neighborhood structures have been implemented. The neighborhood structures $\mathcal{N}_\kappa$ as such are designed to grow as the neighborhood parameter κ increases, hence eventually enabling us to escape local optima. After disturbing any feasible solution by leaving most of the variables fixed and resetting certain bounds, the problem will be reoptimized and possibly ends up in a new (local) optimum. An outline of this procedure is depicted in Algorithm 3. According to a feasible solution all decision variables representing the realization of deliveries ($z^k_{o,d}$) and the assignment of trucks to deliveries ($y_{o,d}$) are going to be fixed according to their values in solution provided. Variables are fixed by resetting their upper and lower bounds to their value in the current solution. During the shaking phase - depending on the current neighborhood under consideration - some of the binary variables mentioned before will be unfixed again. Then the MIP is solved again. Depending on the hybrid variant in use optionally the current solution can further be improved by means of VNS. In case an improved solution was found it becomes the new incumbent solution and we continue with the first neighborhood. Otherwise one goes on with the next neighborhood. The procedure stops after a given time limit t_{max} is reached.

The two approaches described in Fischetti and Lodi (2003) and Hansen et al. (2006) do not explicitly state *which* decision variables are allowed to change. Actually all (however not at the same time) of their binary decision variables would be allowed to change. Rather their neighborhoods are defined in a sense that given any feasible reference solution, at most a certain number of binary decision variables can change (i.e. flip its value from 0 to 1 or vice versa). Their neighborhood structures have been implemented by means of a so called local branching constraint.

In our case we decided not to allow *all* decision variables to change, as the resulting neighborhood would turn out to be too large and it takes to long to restart the optimization process. Rather we keep most of them unchanged. Only *certain* decision variables are allowed to change. Due to the implicit hierarchical structure our partial unfixing mechanism works well, the resulting neighborhoods are of reasonable size and can be solved in an efficient way. The choice which binary decision variables are allowed to flip their value depends on the *neighborhood structure* currently in use.

The VNS-based approach described in Section 4.1 will be used for obtaining an initial solution. Alternatively any other (greedy) construction heuristics may be used. Instead one could also feed the model into any solver used as a black box and wait for any (e.g. the first) feasible solution to be found.

4.2.2 Hierarchy of Decision Variables

The decision variables may be represented using an implicit hierarchical structure. At the top level one has to decide whether or not a certain delivery d associated with order o shall be executed at all. The resulting decision is implemented by means of decision variable $z_{o,d}$. At a second stage one has to specify

Algorithm 3 `Basic outline of VLNS(x)`

```
fix all z_{o,d} and y^k_{o,d} according to x
while stopping criterion not met do                          ▷ run time limit t_max
    κ ← 1
    while κ ≤ κ_max do
        Shaking(x, κ)                                        ▷ unfix some decision variables
        x' ← solve MIP again
        x' ← VNS(x')          ▷ optionally: further improve solution using VNS
        if accept(x') then                                   ▷ move or move not?
            x ← x'
            κ ← 1                                            ▷ continue with 1st neighborhood
        else
            κ ← κ + 1                                        ▷ continue with next neighborhood
        end if
    end while
end while
```

the truck to execute it. Trucks have to be assigned to single deliveries. The choice of truck k for executing delivery d associated with order o is modeled in terms of $y^k_{o,d}$. Finally all truck movements as well as the exact timing for all single deliveries need to be determined and synchronized accordingly.

We observed that once the assignment of trucks to deliveries (and the choice whether or not a certain delivery should be executed at all) is fixed, the resulting model can be solved very quickly. Hence we decided to keep most decision variables $z_{o,d}$ and $y^k_{o,d}$ fixed and only release some of them. Which of those binary decision variables will be un-fixed (i.e. their lower and upper bounds will be set to 0 and 1 respectively) is determined by the current neighborhood structure, using our problem specific knowledge. All remaining variables corresponding to the exact timing of deliveries ($a_{o,d}$, $b_{o,d}$), will be left unrestricted at all times. The same is true for those decision variables responsible for modeling the movement of trucks ($u^k_{o,d}$, $v^k_{o,d}$, $x^k_{o_1,d_2;o_2,d_2}$).

4.2.3 Shaking Operators

The development of the shaking operators was influenced by our problem specific knowledge concerning the hierarchy of the decision variables in use described in the previous section. Three shaking operators have been implemented resulting in $\kappa_{max} = 9$ neighborhood structures $\mathcal{N}_\kappa$, where $\kappa = 1, \ldots \kappa_{max}$.

An overview on all nine neighborhood structures in use, the corresponding shaking operator in use and their sequence is depicted in Table 2. Their structure is described in more detail within the following pages.

The first shaking operator (`FreeDelivery`, responsible for neighborhood $\kappa = 1, 2, 3$) no longer fixes the decision concerning which truck k is supposed to execute a certain delivery. Given the current solution the assignment of trucks

Table 2: Set of Neighborhood Structures for VLNS

κ	Shaking Operator	percentage of items changed at most
1	`FreeDelivery`	10
2	`FreeDelivery`	20
3	`FreeDelivery`	30
4	`SkipOneDelivery`	10
5	`SkipOneDelivery`	20
6	`SkipOneDelivery`	30
7	`AddOneDelivery`	10
8	`AddOneDelivery`	20
9	`AddOneDelivery`	30

to deliveries in up to $10\kappa\%$ cases is discarded. By the use of this shaking operator and the three resulting neighborhood structures we provide more flexibility to the model when it comes to scheduling all trucks accordingly.

By the use of the second shaking operator (`SkipOneDelivery`, responsible for neighborhood $\kappa = 4 \dots 6$) we try to satisfy the demand by trying to serve orders with *less* deliveries then currently scheduled. Therefore randomly at most $10(\kappa - 3)\%$ of all orders where one delivery could be skipped are selected. For the calculations of bounds on the number of deliveries see Section 3.2. For all orders selected the last delivery is skipped. The shortage in the supply of concrete is overcome by choosing another (previous) delivery d' of the same order in return. The choice of the truck scheduled for executing delivery d' is discarded, allowing the model to find another (and possibly larger) truck instead. The choice of delivery d' itself is biased by the capacity of the truck executing it according to the current solution. The selection probability for any delivery d is inversely proportional to the capacity of the truck currently executing it, hence favoring the choice of deliveries being executed by smaller trucks. By using this shaking operator we attempt to reduce the number of deliveries necessary and hence decrease the total distance traveled.

The third shaking operator (`AddOneDelivery`, used in neighborhoods $\kappa = 7 \dots 9$) finally works almost conversely to the one described previously. Rather than trying to reduce the number of deliveries necessary this shaking operator tries to improve any given current incumbent solution by allowing *one additional* delivery. A potential oversupply in concrete has to be avoided. Therefore the assignment of trucks for one of the previous deliveries within the same order is abandoned. The choice of delivery for which a new truck shall be found is again biased. The selection probability for any delivery d' is directly proportional to the capacity of the truck foreseen to execute it based on the current solution. The described changes and the associated variable unfixing is executed for at most $10(\kappa - 6)\%$ of all orders which again are chosen randomly. Again, only orders are considered where the maximum number of deliveries that may be executed is not yet reached by the current solution. This shaking operator has

shown to be very useful especially when it comes to reducing the gaps between consecutive unloading operations. Although higher travel time may result from an increase in the number of deliveries for any order. However we might be able to compensate this by smaller or fewer gaps between consecutive deliveries.

Note that the shaking operators used within the VNS framework change the assignment of trucks to specific deliveries, whereas the number of deliveries remains unchanged. Trucks foreseen for selected deliveries are going to be replaced by specific ones. However the shaking operators used within VLNS result in a larger neighborhood, as the choice of a truck scheduled for executing a particular delivery is freed completely. Any other possibly suitable truck can be chosen instead.

Please note that the shaking operators and resulting neighborhood structures used in the context of VLNS are not identical to the ones used within VNS as described in Section 4.1. The shaking operators used within the context of VNS changes the assignment of trucks to particular deliveries in randomly chosen sequences of up to $(\kappa+1)/2$ $(\kappa/2)$ orders. Thereby the trucks originally scheduled will be replaced by specific trucks not in use so far and trucks randomly chosen respectively. The shaking operators used within the VLNS framework however completely free the choice of truck foreseen for particular deliveries. The assignment of trucks to deliveries will be unfixed. Any suitable truck could execute the delivery instead. The choice however is left to the MILP. Depending on the actual neighborhood structure under consideration the number of deliveries to be executed may also change: additional deliveries can be foreseen and the last delivery could also be eliminated. By freeing the choice of truck for previous deliveries the change in supply can be handled accordingly.

4.3 Hybrid Approaches

As part of the collaborative hybridization the two approaches described in Section 4.1 and 4.2 are combined. After having constructed an initial solution, VNS will be used to further improve until t_{init} seconds passed. Afterwards VLNS will be executed until the total run time limit is reached. Within VLNS decision variables will be fixed according to the current solution. After unfixing certain decision variables the formulation (1)–(20) will be solved again. The optimization process of the embedded MIP stops after the first solution improving the current one has bee found. Optionally VNS can be used to further improve the solution found for t_{inter} seconds. If this optional feature is enabled the resulting approach is referred to as the collaborative hybrid approach. Otherwise we refer to the resulting framework as the integrative hybrid approach. Anyway the resulting solution is returned back into the solver and the procedure starts over again. A graphical representation of this procedures can be found in Figure 1(a) and 1(b) respectively. The resulting frameworks have shown to work highly effective.

5 Experiments

In the following we describe some of the results for the integrative and collaborative hybrid approach proposed in Section 4. Section 5.1 gives a short description on the instances used for testing our algorithm. The lower bounds obtained will be presented in Section 5.2. For a detailed discussion of the results obtained the reader is referred to Section 5.3.

5.1 Data Description

To perform computational experiments we use real data of a concrete company located in Alto Adige, Italy, for all orders placed between January and November 2006. These are the same instances used in Schmid et al. (2008). On average 42.9 orders had to be served per day and an average amount of 514.39 cubic meters (m^3) of concrete daily had to be delivered. Their fleet of vehicles consists of 33 trucks, 14 of which are responsible for the sole delivery of concrete only. Two vehicles are equipped with unloading instrumentation only and do not have space for loading concrete. They cannot be used for transporting concrete. The remaining 17 trucks are hybrid vehicles, which can be used for the delivery of concrete as well. Some trucks are also equipped with a pump or a conveyor belt respectively in order to assist during unloading operations. The fleet is heterogeneous in terms of their loading capacity. The average loading capacity per truck is 8.6 m^3. The largest (smallest) truck can carry up to 11 (6.5) m^3 of concrete. On average 40.91% of all orders can be unloaded without any special needs and equipment. 56.64% (2.45%) of all orders require a pump (conveyor belt) at hand in order to execute all unloading operations respectively.

All orders to be satisfied the next day are known the evening before. The schedule is calculated during the night. One problem instance relates to one single day, taking into account all orders that had to be fulfilled that particular day.

The fleet of trucks, their instrumentation and capacity, as well as the number and location of plants are fixed and given according to the equipment available at hand. The fleet under consideration is large enough to satisfy all orders in a timely manner. Deliveries will not be outsourced to other logistic providers.

The algorithm and its variants have been tested on 20 chosen test instances. One instance refers to one day and all orders to be satisfied on this particular day. The instances can be grouped into four different classes, varying from very small (mini), small, medium-sized and larger ones. An overview on the particularities of all instances and the aggregated classes is shown in Table 3.

The number of orders per instance (day) is denoted by n_o. The total (average) amount of concrete to be delivered per day is denoted by $\sum Q$ (Q_{avg}). The ordered quantity corresponding to the smallest (largest) order per day is depicted by Q_{min} and Q_{max} respectively. The last column shows the standard deviation Q_σ of the ordered quantities per day.

The first five instances consist of 13 to 18 orders, respectively, and have been classified as *mini*-sized instances. The group of *small* instances contains

testcases with up to 39 orders. Instances with a total number of orders ranging from 50 to 60 are classified as *medium*-sized instances. Instances with a total number of orders between 65 and 76 are referred to as *large*.

Table 3: Properties of selected Instances

n	n_o	$\sum Q$	Q_{avg}	Q_{min}	Q_{max}	Q_σ
1	13	127.5	9.81	1.5	27	7.94
2	14	123	8.79	2	18.5	5.61
3	17	305.5	17.97	1	128.5	30.27
4	18	267.5	14.86	3	40	10.59
5	19	216	11.37	1	29.5	8.35
mini	16.2	207.9	12.56	1.7	48.7	12.55
6	27	554.5	20.54	0.5	97.5	28.92
7	28	305.75	10.92	0.75	48	12.42
8	33	413	12.52	0.5	101.5	19.52
9	34	535	15.74	0.5	98	19.67
10	39	498.5	12.78	1	178	29.71
small	32.2	461.35	14.50	0.65	104.6	22.05
11	50	736	14.72	0.5	172	30.15
12	50	502	10.04	0.5	48	11.05
13	55	491	8.93	1	36	8.67
14	55	824.5	14.99	1	104	21.15
15	60	648	10.80	0.25	66	15.18
medium	54	640.3	11.90	0.65	85.2	17.24
16	65	776	11.94	0.5	133.5	21.10
17	65	637.75	9.81	0.25	55	10.25
18	70	719	10.27	0.5	53	11.71
19	70	886	12.66	0.5	99	16.66
20	76	721.25	9.49	0.25	115	14.36
large	69.2	748	10.83	0.4	91.1	14.82

The following section shows the results obtained. If not stated otherwise, all calculations have been executed on desktop PCs (3.2 GHz, 3 GB RAM) and XPRESS-MP (v. 2007A) was used for solving all MIP models. All run times are given in seconds. Usually we present average and best results obtained per instance. All values presented are averaged over five independent runs with different seeds for the embedded random number generator. Variations within the results are the result of random choices within variable neighborhood search and local branching respectively. Aggregated values presented per categories (mini/small/medium/large) correspond to averages over the values obtained for all individual instances within that category.

5.2 Lower Bounds

Table 4 shows the lower bounds obtained at the root node (LB_o) as well as the time necessary for solving it (T_o) for the model formulation as described in Section 3.3. In other words, LB_o is the optimal value of the LP relaxation of

model (1)–(29). Furthermore the lower bound (LB) obtained after a total run time limit of 4800 seconds is shown. The total number of variables of the model under consideration as well as the number of nodes that have been explored within the run time limit given, are depicted in the last two columns. For the very small instances, on average, 1331.6 nodes can be explored within the given run time limit. For larger instances however time is only sufficient for solving the root node only, except for instances 14 and 19. By exploring other nodes of the branch-and-bound tree we have been able to improve the lower bound at the root node by 0.5% as compared to the solution obtained after solving the LP relaxation only.

Table 4: Lower Bounds

n	LB_o	T_o	LB	$\#var$	$\#nodes$
1	316.92	17.67	331.02	27837	2989
2	754.43	10.72	754.43	32173	2665
3	903.51	89.06	923.57	105180	154
4	1109.80	109.08	1170.12	96053	246
5	640.56	64.16	642.07	78814	604
mini	745.04	58.14	764.24	68011.4	1331.6
6	1297.62	934.41	1305.35	343927	16
7	1065.74	242.06	1079.65	156625	102
8	1416.50	402.25	1431.41	235080	28
9	1414.15	796.23	1414.73	354173	16
10	1704.40	713.03	1704.52	368146	28
small	1379.68	617.60	1387.13	291590.2	38.0
11	2156.50	2628.61	2159.10	675504	1
12	1813.15	3426.45	1813.97	437315	4
13	1831.17	1286.38	1839.73	449383	17
14	1921.33	–	1921.32	844314	0
15	1762.92	2766.66	1769.55	698975	3
medium	1897.01	3204.23	1900.73	621098.2	5.0
16	2025.03	3849.11	2025.03	897342	1
17	2032.70	3170.80	2037.78	662983	1
18	2093.73	3436.30	2131.17	849539	1
19	2450.63	–	2450.63	1136261	0
20	2023.07	4381.22	2023.07	955347	1
large	2125.03	3975.85	2133.54	900294.4	0.8

Table 5 indicates the number of valid inequalities that are added dynamically. The total run time limit was again set to 4800 seconds. All described inequalities are useful when solving the problem.

5.3 Computational Results

In order to show the efficiency of the hybrid frameworks the following experiment was set up: Both the integrative and the collaborative hybrid, as well as the VNS applied solely, were run for a total run time limit of $t_{max} = 3600$ seconds.

Table 5: Number of valid inequalities added dynamically

n	(18)	(17)	(10)+(11)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(29)
1	3504	1483	3195	25	93	3077	83	3244	485	77	5503
2	4185	1344	2037	38	77	1717	347	1949	461	98	4923
3	4254	1233	537	21	47	807	75	879	3038	65	1462
4	1815	1305	897	10	11	892	73	1002	481	43	2465
5	2226	1358	834	11	22	1141	106	1224	401	58	2544
mini	3196.8	1344.6	1500.0	21.0	50.0	1526.8	136.8	1659.6	973.2	68.2	3379.4
6	5640	1728	255	18	8	548	37	581	728	129	539
7	1653	1401	446	25	3	706	11	705	634	51	1098
8	2632	1336	417	14	23	747	36	774	1889	113	1036
9	4209	1788	335	20	6	658	22	673	1186	128	780
10	1777	1528	162	30	1	535	34	524	1040	124	623
small	3182.2	1556.2	323.0	21.4	8.2	638.8	28.0	651.4	1095.4	109.0	815.2
11	12315	2150	628	22	11	574	17	579	4304	151	617
12	2147	2058	236	24	2	615	19	612	627	143	876
13	1460	2226	66	43	0	436	11	452	462	134	731
14	7663	2573	101	24	2	724	19	693	4348	164	699
15	2697	2319	49	36	1	574	11	531	948	145	597
medium	5256.4	2265.2	216.0	29.8	3.2	584.6	15.4	573.4	2137.8	147.4	704.0
16	7644	2857	37	43	7	586	17	556	1907	204	593
17	1483	2486	97	23	0	607	20	611	152	199	944
18	2775	2990	130	46	1	753	13	734	758	157	714
19	5113	3305	76	36	1	601	21	565	1675	259	597
20	4782	3114	47	56	2	541	14	488	2137	171	601
large	4359.4	2950.4	77.4	40.8	2.2	617.6	17.0	590.8	1325.8	198.0	689.8

The time designated for generating an initial solution using VNS was set to 1200 seconds. For the collaborative hybrid approach 10 seconds were spend on VNS after each iteration. The results obtained are presented in Table 6. Columns heading z_{min} (z_{avg}) denote the best (average) solutions found over five runs. The average time when the solution was found is denoted by t_z. The last two columns indicate the best and average solutions found when the VNS is applied solely without any use of VLNS.

The best average solution among all three approaches is highlighted in bold. The best among all best solutions found is printed in italics. The results clearly show that both hybrid approaches (with and without the optional VNS component) work very well. On average for small, mini and medium-sized instances better solutions are found by the hybrid algorithms. They improve the solutions found if VNS is applied solely. This clearly demonstrates the effectiveness of VLNS.

For mini instances the collaborative hybrid works best on average, although the VNS is able to find some best solutions. The average solution quality found using the collaborative (integrative) hybrid approach is 801.5 (804.8). Whereas the average solution quality found by VNS is 807.8. All best solutions known

Table 6: Solutions obtained using hybrid approaches and pure VNS

	integrative hybrid			collaborative hybrid			VNS		
n	z_{min}	z_{avg}	t_z	z_{min}	z_{avg}	t_z	z_{min}	z_{avg}	t_z
1	*331.9*	335.4	2272.7	*331.9*	333.7	1568.6	*331.9*	**333.0**	105.9
2	*762.0*	762.3	1852.8	*762.0*	762.2	1281.5	*762.0*	**762.0**	104.4
3	*1007.7*	1052.3	2750.4	*1007.7*	**1043.7**	3042.0	1029.3	1078.4	2018.5
4	*1183.0*	1187.2	2584.3	*1183.0*	1185.6	1848.1	1183.0	**1184.1**	671.3
5	666.9	687.0	2776.1	666.9	682.2	2423.0	*666.9*	**681.6**	1147.0
mini	*790.3*	804.8	2447.3	*790.3*	**801.5**	2032.7	794.6	807.8	809.4
6	1529.0	1594.1	2209.9	1539.5	1578.9	2392.6	*1517.5*	**1567.3**	3080.9
7	*1118.0*	1154.3	2505.7	1119.5	**1148.1**	2545.2	1130.5	1156.0	2517.8
8	1528.5	**1548.3**	2287.9	*1512.0*	1550.7	2069.7	1519.2	1550.8	2211.9
9	1522.9	1560.4	2832.8	*1519.9*	**1552.3**	2775.7	1550.1	1599.3	2900.7
10	1948.9	2028.7	2735.3	1966.1	**2024.3**	2796.9	*1943.3*	2057.1	3256.5
small	*1529.4*	1577.2	2514.3	1531.4	**1570.9**	2516.0	1532.1	1586.1	2793.5
11	*2430.7*	2572.7	2189.9	2434.1	**2554.9**	2446.5	2448.2	2599.8	3190.6
12	1928.5	1990.7	2548.2	*1921.4*	**1984.1**	2048.3	1948.7	2036.1	3400.1
13	*1976.4*	2057.3	2397.4	*1976.4*	**2034.7**	2810.6	2036.3	2081.6	3290.2
14	2446.4	2676.6	1633.2	*2428.3*	**2608.4**	2183.1	2558.0	2804.7	3369.6
15	*2055.8*	2192.5	2961.1	2090.0	**2178.7**	2810.1	2172.1	2288.0	3313.5
medium	2167.6	2298.0	2346.0	2170.1	**2272.2**	2459.7	2232.7	2362.0	3312.8
16	n/a	n/a	n/a	n/a	n/a	n/a	*2492.4*	**2659.2**	3473.0
17	*2211.7*	2345.4	2126.7	*2211.7*	**2330.2**	2703.3	2250.2	2412.5	3291.9
18	*2394.9*	**2590.2**	1801.5	2469.6	2602.8	1811.3	2398.0	2723.8	3454.0
19	n/a	n/a	n/a	n/a	n/a	n/a	*3009.4*	**3317.4**	3336.8
20	4337.8	4337.8	1334.0	n/a	n/a	n/a	*2494.2*	**2644.7**	3382.6
large	n/a	n/a	n/a	n/a	n/a	n/a	2528.8	**2751.5**	3387.7

were found by the integrative hybrid approach. For small instances however the integrative hybrid approach works slightly better. All three approaches obtain comparable solution qualities. The VNS requires the most amount of time for finding the solutions. Our hybrids clearly outperform the pure VNS variant for medium sized instances. Note that medium sized instances refer to the most common type of instances that appear in real world problems faced by our industry partners. For medium sized instances the collaborative hybrid approach works best. It provides the best average solutions over all instances and 3 out of 5 best known solutions have been found. In total 5 out of 5 best known solutions were found by one of our hybrids. The average solution quality found using the collaborative (integrative) hybrid approach is 2272.2 (2298.0). The average solution quality found by VNS is at 2362.0. Note that given a run time limit of 3600 seconds it takes on average 2459.7 (2346.0) for finding a solution using our hybrid approaches. Whereas the VNS consumes 3312.8 seconds. The best average solutions and most of the global best solutions can be found using the collaborative hybrid approach. For large instances however the VNS clearly dominates both hybrid approaches as we're encountering memory problems. Nevertheless for instances 17 and 18 the global best and average best solution can be found by our integrative hybrid approach.

Furthermore the results obtained have been statistically tested. For mini

and small instances statistical significance cannot be proven given a reasonable significance level. Please note that the resulting gaps to the best lower bounds found are already very low and both types of algorithms perform very well on these types of instances. For 2 instances the optimal solution could be found. Given a significance level of $\alpha = 0.01$ our hybrid methods perform better than the standalone VNS in 4 out of 5 medium-sized instances. For instance 17 and 18 the integrative hybrid approach clearly dominates the standalone VNS given a level of significance of $\alpha = 0.005$.

One can clearly see the strengths of our hybrid approaches. They are perfectly suitable for solving typical real-world-sized instances. Larger problem instances can easily be splitted into smaller ones or the solution procedure can be solved using a rolling horizon concept.

Table 7 displays the best bounds found after a total run time limit of 4800 seconds. Column with label LB is a copy of the same column in Table 4. The best solutions (z_{min}) found using our hybrid approaches (after 3600 seconds) as well as the resulting gaps ($\%z_{gap}$) are also shown below. Furthermore we tried to solve the formulation (1)–(20) on a more powerful computer with 3.0 GHz, 4 GB RAM and Cplex 11. The best lower bounds (denoted by LB') as well as the best solution (z_{min}) found within a run time limit of 500.000 seconds are shown in the last two columns respectively. The MIP formulation (1)–(20) cannot be solved for reasonable sized problem instances using a standard general purpose solver such as Cplex. Only 4 of the very smallest instances (mini) were solved in our experiments. When solving mini instances using the collaborative hybrid approach the resulting gap is as small as 2.89%. The performance of the VNS is slightly worse. The resulting gap observed is 3.27%. For the medium sized problems the gap to the lower bound using our hybrid approaches is 12.0% and 12.31% respectively, whereas with the pure VNS the resulting gap is 14.36% using the same runtime.

6 Conclusion

In this work we present an efficient hybrid metaheuristic for a highly relevant problem in distribution logistics - the delivery of RMC to construction sites. The problem is a complex scheduling problem on a daily basis. The concrete produced at several plants has to be delivered at customers' construction sites using a heterogeneous fleet of vehicles. Moreover, special equipment is required in order to unload the concrete at the construction site.

For approaching problems in distribution logistics, a large number of algorithmic strategies has been proposed in the last decades. Metaheuristics have proven to be highly useful in practice but do not yield performance guarantees with respect to solution quality. Exact algorithms are very often not applicable for medium to large size problems due to their excessive runtimes and memory requirements, although the performance of exact solution techniques has dramatically improved over the last years. Now, it is possible to combine the two

Table 7: Comparison

		VNS		integrative hybrid		collaborative hybrid		MIP	
s n	LB	z_{min}	$\%z_{gap}$	z_{min}	$\%z_{gap}$	z_{min}	$\%z_{gap}$	LB'	z_{min}
1	331.0	331.9	0.27	331.9	0.27	331.9	0.27	314.6	331.9
2	754.4	762.0	0.99	762.0	0.99	762.0	0.99	754.8	762.0
3	923.6	1029.3	10.28	1007.7	8.35	1007.7	8.35	889.9	n/a
4	1170.1	1183.0	1.09	1183.0	1.09	1183.0	1.09	1116.5	1293.6
5	642.1	666.9	3.73	666.9	3.73	666.9	3.73	639.5	729.7
mini	764.2	794.6	3.27	790.3	3.30	790.3	2.89	743.0	n/a
6	1305.3	1517.5	13.98	1529.0	14.63	1539.5	15.21	1297.6	n/a
7	1079.6	1130.5	4.49	1118.0	3.43	1119.5	3.56	1063.6	n/a
8	1431.4	1519.2	5.78	1528.5	6.35	1512.0	5.33	1411.1	n/a
9	1414.7	1550.1	8.73	1522.9	7.10	1519.9	6.92	1414.2	n/a
10	1704.5	1943.3	12.29	1948.9	12.54	1966.1	13.30	1675.2	n/a
small	1387.1	1532.1	9.05	1529.4	9.31	1531.4	8.86	1372.3	n/a
11	2159.1	2448.2	11.81	2430.7	11.17	2434.1	11.30	2137.3	n/a
12	1814.0	1948.7	6.91	1928.5	5.94	1921.4	5.59	1813.2	n/a
13	1839.7	2036.3	9.65	1976.4	6.91	1976.4	6.91	1829.7	n/a
14	1921.3	2558.0	24.89	2446.4	21.46	2428.3	20.88	1892.1	n/a
15	1769.6	2172.1	18.53	2055.8	13.93	2090.0	15.33	1753.6	n/a
medium	1900.7	2232.7	14.36	2167.6	12.31	2170.1	12.00	1885.2	n/a
16	2025.0	2492.4	18.75	n/a	n/a	n/a	n/a	2017.2	n/a
17	2037.8	2250.2	9.44	2211.7	7.86	2211.7	7.86	2028.9	n/a
18	2131.2	2398.0	11.13	2394.9	11.01	2469.6	13.70	2080.1	n/a
19	2450.6	3009.4	18.57	n/a	n/a	n/a	n/a	2445.7	n/a
20	2023.1	2494.2	18.89	4337.8	53.36	n/a	n/a	2016.7	n/a
large	2133.5	2528.8	15.35	n/a	n/a	n/a	10.78	2117.7	n/a

different solution philosophies and to take advantage of the synergy of both. For an overview of different hybridization schemes see Puchinger and Raidl (2005); Raidl (2006).

The developed new hybrid solution procedure is based on a combination of an exact algorithm and a VNS. The idea of hybridization is inspired by the seminal work of Hansen et al. (2006). The VNS is used to generate feasible solutions. Within VLNS the exact method based on a MILP formulation is solved respectively after an appropriate variable fixing.

The strength in the developed solution approach lies in the excellent solution quality of medium sized real world test instances. The complexity of the daily decision situation in most of the Austrian concrete companies is of that size. Even if the company is larger - the decision maker usually cluster not more than five concrete plants to simultaneously develop the schedule for a day. For the medium sized problems the gap to the lower bound is 12.0%, whereas with the pure VNS the gap is 14.36% using the same runtime. For individual instances the resulting gap using the standalone VNS increases up to 24.89%.

These efficient hybridization scheme should be applied to standard periodic vehicle routing problems, e.g. the periodic traveling salesman problem or the periodic vehicle routing problem (Hemmelmayr et al.).

Acknowledgements

This project was supported, in part, by the Austrian Science Fund (FWF) grant # P20342-NI3 and by the Spanish research project MTM2006-14961-C05-03. Academic Licenses of XPRESS-MP (v. 2007A) by Dash Optimization was used in our computational experiments.

References

R. K. Ahuja, Ergun, J.B Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1–3): 75–102, 2002.

C. Archetti and M.G. Speranza. An overview on the split delivery vehicle routing problem. In *Operations Research Proceedings 2006*, pages 123–127, 2007.

L. Asbach, U. Dorndorf, and E. Pesch. Analysis, modeling and solution of the concrete delivery problem. *European Journal of Operational Research*, to appear.

Dash. *Xpress-BCL Reference manual*. Dash Optimization Ltd, 2007.

M. T. Durbin. *The Dance of the Thirty-Ton Trucks: Demand Dispatching in a Dynamic Environment*. PhD thesis, George Mason University, 2003.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98 (1–3):23–47, 2003.

M. Gendreau, G. Laporte, and J.Y. Potvin. Vehicle routing: Modern heuristics. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*. John Wiley & Sons, New York (NY), USA, 1997.

F.W. Glover and G.A Kochenberger, editors. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Kluwer Academic Publishers, Norwell (MA), USA, 2003.

P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

P. Hansen, N. Mladenović, and D. Urošević. Variable neighborhood search and local branching. *Computers & Operations Research*, 33(10):3034–3045, 2006.

V.C. Hemmelmayr, K.F. Doerner, and R.F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*. to appear, available online since November 2007.

K. Hoffman and M. Durbin. The dance of the thirty ton trucks. *Operations Research*, 56(1):3–19, 2008.

H.H. Hoos and T. Stützle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann Publishers, San Francisco (CA), USA, 2004.

G.A.P. Kindervater and M.W.P. Savelsbergh. Vehicle routing: Handling edges exchanges windows. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*. John Wiley & Sons, Chichester, USA, 1997.

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. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

J.J. Moder, C.R. Phillips, and E.W. Davis. *Project Management with CPM, PERT and Precedence Diagramming*. Van Nostrand Reinhold, New York (NY), USA, 1983.

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, 2007a.

D. Naso, M. Suricom, and B. Turchiano. Reactive scheduling of a distributed network for the supply of perishable products. *IEEE Transactions on Automation Science and Engineering*, 4(3):407–423, 2007b.

J. Puchinger and G.R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *IWINAC 2005*, LNCS, pages 41–53, Berlin, 2005. Springer.

G.R. Raidl. A unified view on hybrid metaheuristics. In *Proc.Hybrid Metaheuristics 2006*, LNCS, pages 1–12, Berlin, 2006. Springer.

V. Schmid. *Trucks in Movement: Hybridization of Exact Approaches and Variable Neighborhood Search for the Delivery of Ready-Mixed Concrete*. PhD thesis, University of Vienna, 2007.

V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh, and W. Stoecher. A hybrid solution approach for ready-mixed concrete delivery. 2008. submitted.

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