universität wien

DISSERTATION

Titel der Dissertation

„Trucks in Movement:

Hybridization of Exact Approaches and Variable Neighborhood Search for the Delivery of Ready-Mixed Concrete“

Verfasserin

Mag. Verena Schmid

angestrebter akademischer Grad

Doktorin der Sozial- und Wirtschaftswissenschaften
(Dr. rer. soc. oec.)

Wien, im November 2007

Studienkennzahl lt. Studienblatt	A 084 157
Dissertationsgebiet lt. Studienblatt	Internationale Betriebswirtschaft
Betreuer	o.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

Concrete is needed almost everywhere. In order to build offices, commercial, residential or retail buildings, factories, industrial or agricultural buildings, etc., some kind of building material is needed. Concrete is one of the most used construction materials (see ERMCO, 2000).

Ready mixed concrete is a prosperous market. The total amount of concrete produced in the European Union has increased from 318.4 million m^3 in 2002 to 369.6 million m^3 in 2005, which refers to a percental increase of 16.08%. In the United States the total amount of concrete produced has risen 15% to up to 345 million m^3 per year (see ERMCO, 2004, 2005) in the same period. Emerging markets such as China and India push the raising demand for concrete even further.

Concrete itself is produced by blending cement, water and aggregates such as gravel and sand. Additionally certain admixtures, e.g. retarders and accelerators are added to affect the hydration (hardening) process of the material. Depending on what is the purpose of the construction to be built, other ingredients are used to improve certain properties, change color effects or water permeability. Concrete is a perishable good, in a sense that it hardens after a given period of time. During the blending process it is still smooth and can be transported for a limited amount of time if continuously in movement. After about two hours, depending on accelerators or retarders in use, concrete hardens and it will reach its durability and required strength.

As the name already suggests, ready-mixed concrete is not produced directly at construction sites, where the actual demand occurs. Concrete is produced at plants from where it is transported to construction sites using special types of vehicles designated for transporting it. Either concrete is mixed just-in-time for the loading operation of a truck. Or alternatively raw materials may be poured into the truck and are mixed on the way to the construction site.

Concrete needs to be delivered from plants to construction sites using a heterogeneous fleet of vehicles. Trucks need to be scheduled such that the demand at construction sites can be satisfied. Usually an order cannot be satisfied by just one single truck, therefore several consecutive unloading operations need to be scheduled. Typically concrete com-

panies operate various plants. So one additional degree of freedom refers to the choice of plant where a truck should be loaded before delivering concrete to a specific construction site. The objective consists of minimizing travel times of all trucks. All orders have to be satisfied in a feasible way, while also taking into account special types of unloading equipment that might be necessary. Consecutive deliveries at one and the same construction site cannot overlap. Any gaps in between will lead to a penalty in the objective function.

Linear Programming (LP) and formulations based on Mixed Integer Programming (MIP) will find - in case they converge - an optimal solution. One of the disadvantages of this approach might be that it would take way too long to solve those problems to optimality. On the other hand so called heuristics - or even more sophisticated metaheuristics - exist, that try to find good (but not necessarily optimal) solutions in a reasonable amount of time.

For solving large scale problem instances we decided to use hybrid approaches, combining the power of meta-heuristics with the strength of exact approaches to overcome the disadvantages of the two approaches if applied exclusively. Although it would be possible to state a complete mathematical formulation for the problem considered using a MIP-formulation it is not possible to solve it in a reasonable amount of time as the number of decision variables and constraints involved increases exponentially as the size of the instance considered increases. Therefore two hybrid approaches have been developed to overcome the problems mentioned and solving problems in a reasonable amount of time.

This thesis presents a broad range of different ways on how to solve the problem stated before. Various solution methods based on exact, heuristic, meta-heuristic and hybrid approaches have been developed.

Exact methods based on a formulation for the Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows (SDMDHVRPTW) have been implemented. In order to facilitate readability, this problem will be referred to as VRP*. The resulting problem formulation can be solved to optimality for very small instances. For real-world-sized instances, even with a steady increase in computational power, just “to MIP” is not the way to success. Hence an algorithm, that controls the solution process of the embedded MIP-formulation, has been developed to tackle larger problem instances as well. This *integrative hybrid* approach is based on Local Branching (LB) (see Fischetti and Lodi, 2003) which itself is guided by means of Variable Neighborhood Search (VNS) (see Hansen and Mladenović, 2001; Mladenović and Hansen, 1997; Hansen et al., 2006). Attention has also been paid to the development of valid inequalities and cuts to improve

the quality of lower bounds. VNS is a highly promising metaheuristic. See Glover and Kochenberger (2003) for a general overview on most of the existing metaheuristics. Rather than relying on commercial solvers such as XPRESS or CPLEX, that are mainly used as black box, VNS is able to efficiently search the solution space by iteratively applying both diversification and intensification strategies.

Another approach has been developed, that is based on a multi-commodity network flow (MCNF) model formulation. The two resulting model formulations themselves have been inspired by the model proposed in Durbin (2003) and Hoffman and Durbin (2007). Rather then having a comprehensive view on the problem, only subparts of the problem are considered and solved to optimality. So called *patterns* (options on how orders may be satisfied) are generated heuristically and serve as an input for the MCNF. This approach has first been presented in Schmid et al. (2006a) and Schmid et al. (2006b).

Moreover the entire problem may be tackled by just using VNS on its own. However the best results where obtained when combining the two last-mentioned approaches. Both methods used solely are capable of solving such problems. However, only the *cooperative hybrid* approach enables us to combine the strengths of both techniques and compensates their major drawbacks. Iteratively solutions obtained by MCNF serve as input for VNS which is going to (locally) optimize it. The resulting solution (in terms of patterns) is fed back into the MCNF problem, which is going to be optimized again. This approach was first introduced in Schmid et al. (2007b) and Schmid et al. (2007c).

It can be shown that both hybrid approaches and the embedded combination of two methods are by far more efficient then the use of any approach solely. Additionally we compare our algorithm with a software tool based on Simulated Annealing (SA) available in Austria. Our hybrid algorithms outperform results obtained by this tool.

Figures 1.1a and 1.1b present a basic outline of the integrative and cooperative approach. In the integrative hybrid approach LB serves as subordinate, supporting the solution process of the VRP*. Whereas the cooperative hybrid approach consists of two methods, where both of them may also be applied solely and are equally valuable during the search process.

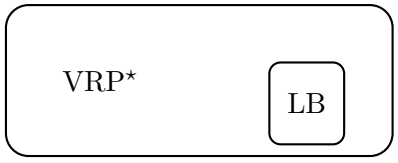

Figure 1.1a.: Integrative Hybrid Approach

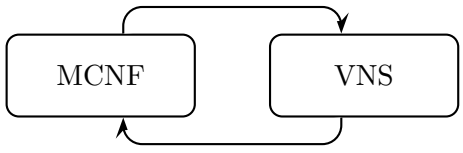

Figure 1.1b.: Cooperative Hybrid Approach

The remainder of this thesis will be organized as follows: Exact methods based on a formulation for the VRP* will be presented in Chapter 3. The integrative hybrid approach based on Local Branching (LB) is presented in Chapter 4. Chapter 5 states the formulations based on a MCNF model, the VNS will be presented in Chapter 6. The combination between the two last-mentioned approaches into a cooperative hybrid version will be introduced in Chapter 7. Chapter 8 finally compares the solutions obtained using the different approaches mentioned before.

Results and insights obtained during the development process of this PhD thesis have been incorporated in Schmid et al. (2007a) and Schmid et al. (2007d).

2. Delivery of Concrete

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. One needs to assign plants to deliveries and hence decide from which plant concrete will be transported the associated construction site. Next all resulting truck movements need to be scheduled accordingly. Additionally a large number of technical constraints dealing with the unloading operation itself 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 deliveries need to be scheduled to fulfill an order. At most one truck may unload at a time. Single deliveries to be executed at one construction site may not overlap. Satisfying orders on time is essential, consequently time windows need to be considered. Constructors require an uninterrupted supply of concrete, hence the time between consecutive deliveries will tried to be kept as small as possible.

Some vehicles may be used for the delivery of concrete only. Other vehicles, with specialized unloading equipment, may have to be present at a construction site and assist other vehicles with their unloading operation (some of them may also be able to transport concrete). If special unloading equipment is demanded by an order such vehicles need to arrive first at a construction site and remain at the construction site until the complete order has been fulfilled. It is not possible to displace the truck assisting others with their unloading operation. If an order requires special unloading equipment the first truck to arrive needs to stay until the last truck has finished its unloading operation. Any displace would be impractical, as it would take too much time, and hence disturb the unloading process.

Concrete is not a homogeneous product, rather many different recipes exists. As opposed to many other problems related to vehicle routing any truck may only service one

order at a time. It is not possible 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 the customer and trucks cannot serve two different orders with the same concrete in their loading space. Therefore all trucks basically commute between a plant, where they are going to be loaded, and construction sites where unloading operations are supposed to take place. After unloading concrete at the orders' construction site it was dedicated for, trucks drive to a plant again.

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

Trucks typically move between construction sites and plants. Loading operations take place at plants, afterwards the trucks need to unload concrete at construction sites. Every truck is assigned to a specific plant which is referred to as its home plant. Trucks start their daily tour at their home plant and need to come back there by the end of the day. The first loading operation per day is supposed to be executed at the corresponding home plant. All remaining loading operation throughout the day may be executed at any other plant as well and is not restricted to the home plant. In practice however schedulers tend to schedule all trucks for each plant individually. This restricted view may lead to sub-optimal solution. Using a global perspective however and by taking into account several plants simultaneously will dramatically help to improve the quality of solutions found. Figures 2.1a and 2.1b respectively depict a typical situation where profit can be obtained by not scheduling the movements of trucks for each plant separately.

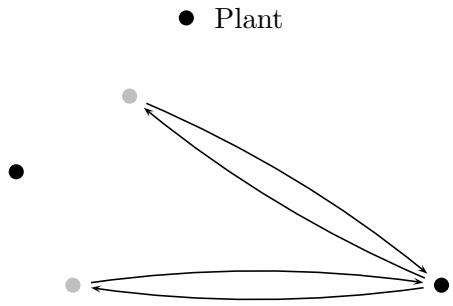

Figure 2.1a.: Loading restricted to home plant

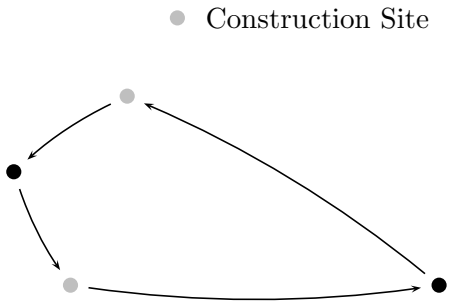

Figure 2.1b.: Loading at any plant

2.1. Related Work

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 to design a Decision Support System (DSS) for routing trucks distributing RMC within a dynamic environment. In his work he concentrates on the DSS. Routing is determined by using heuristics and the exact approach itself is only mentioned shortly. In comparison to our approach Matsatsinis splits the scheduling of pumps and concrete carrying vehicles, resulting in a two level approach. In our case however such decomposition is not possible, as some (hybrid) trucks are equipped with special unloading equipment such as a pump or a conveyor belt and may also be used for the delivery of concrete.

Naso et al. (2007) 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.

Durbin (2003) and Hoffman and Durbin (2007) developed a decision-support tool. In addition to a time-space network formulation they used a minimum-cost network flow model and a heuristic based on tabu search to solve the problem at hand. His foundation based on a time-spaced network representation mainly inspired our approach for solving

the problem at hand. The main idea of providing a set of different delivery options and finally choosing one alternative per order however distracts the pure network flow formulation.

A comprehensive approach for solving the problem refers to a formulation similar to Capacitated Vehicle Routing Problems (CVRPs). Vehicle Routing Problems (VRPs) are an extension of the classical and probably most extensively studied problem in logistics - the Traveling Salesperson Problem (TSP) (see Bellmore and Nemhauser, 1968). A fleet of vehicles, located at one single depot needs to be scheduled such that the customers demand can be satisfied, while visiting every customer exactly once. CVRPs additionally take into account the capacity of the vehicles that must not be exceeded at any point in time. Capacitated Vehicle Routing Problems with Time Windows (CVRPTWs) also take into account certain additional constraints concerning *when* customers should be visited. One further extension refers to the location of the vehicles. As soon as the vehicles are no longer located at one single depot but rather spread among various depots the problem under consideration is called MDVRPTWs. VRPs and its various extensions are a very active field in the research community and have been studied comprehensively in Toth and Vigo (2001).

An extensive overview on Vehicle Routing Problems with Time Windows (VRPTW) can be found in Bräysy and Gendreau (2005a,b). VRPs with multiple depots have been extensively studied by Chao et al. (1993) and Cordeau et al. (1997). MDVRPTWs have been tackled by Cordeau et al. (2001) and Polacek et al. (2004).

A common assumption for VRPs is, that every customer has to be visited *exactly once* and that the demand of any single customer is *less* than the capacity of any vehicle. In our case the demand of a single order typically *exceeds* the capacity of any single truck. Several deliveries need to be executed to completely satisfy those orders. Due to the fact that we consider a heterogeneous fleet of vehicles as well, we cannot pre-determine that exact number of deliveries to be executed.

The first extension to the classical VRP taking into account serving customers with more than one delivery has been developed by Dror and Trudeau (1989, 1990). They addressed the Split Delivery Vehicle Routing Problem (SDVRP), which - opposed to the classical formulation - drops the constraint that every customer has to be visited exactly once. They showed that by allowing deliveries to be split - and hence allowing that any customer may be visited more than once - substantial savings can be obtained. However they only considered the case in which the demand of any single customer is less than or

equal to the capacity of the vehicles. An IP formulation as well as valid inequalities has been developed in Dror et al. (1994). See Archetti and Speranza (2007) for an overview on SDVRPs.

The case where the demand exceeds the capacity also has been investigated. Archetti et al. (2006) developed bounds and performed worst-case analysis for a similar problem referred to as VRP$^+$, a variant of the VRP, where the demand (Q_i) of a customer could exceed the capacity C of the vehicle and every customer has to be visited exactly $t_i = \lceil Q_i/C \rceil$ times. Another extension referred to as SDVRP$^+$ also has been investigated, where the previous constraint is relaxed in a sense that every customer (obviously) has to be visited at least t_i times. Lower Bounds are explored in Belenguer et al. (2000).

As a further extension can be found in Archetti et al. (2005), where both the case of a homogeneous fleet of vehicles with a capacity of two units and of a mixed fleet of vehicles with capacity one and two respectively are considered. In case time windows need to be considered as well, the resulting formulation is called Split Delivery Problem with Time Windows (SDVRPTW). Different approaches based on Branch and Price and Tabu Search can be found in Feillet et al. (2002) and Ho and Haugland (2004) respectively.

Within these approaches the exact timing of several deliveries to be executed for one and the same customer is not important. Resulting gaps between consecutive deliveries have no effect on the quality of the solution. In our case however, as constructors require a preferably continuous inflow of concrete, consecutive unloading operations need to be scheduled such that resulting gaps are tried to be kept as small as possible.

Following this notation our problem can be classified as a Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows (SDMDHVRPTW$^+$), while also taking into account that fact that the demand of any single customer might exceed the capacity of any given truck available and that the exact number of visits cannot be pre-determined. Throughout the remainder of this thesis we will refer to this problem as the VRP*.

Our problem also seems to be related to the so called Vehicle Routing Problem with Full Truckloads (VRPFL). In the case of VRPFL a given number of full truckloads need to be shipped between specified pairs of locations. These problems have been addressed by Arunapuram et al. (2003). The decision to be made can be reformulated in terms of minimizing empty vehicle movements. An exact approach based on a formulation similar to the one used for asymmetrical VRPs under distance restrictions has been proposed by Desrosiers et al. (1988). For a comprehensive overview on VRPFL the interested reader is referred to Bodin et al. (1983) and Ball et al. (1983) respectively. Our problem however

does not specify shipments to be made between specific pairs of locations (i.e. plants and construction sites). Any order may be delivered from any plant that is not too far away. An assignment of plants to orders is not supposed to be done. Additionally one needs to keep in mind that the number of deliveries necessary cannot be determined in advance and that single deliveries associated with one and the same order should be scheduled just-in-time, in order to ensure a preferably continuous inflow of concrete.

VNS systems have proven their effectiveness in a multitude of problems. One of the first descriptions can be found in Mladenović and Hansen (1997) and Hansen and Mladenović (2001). Further efficient implementations for solving routing problems have been developed by Kytöjoki et al. (2007). The VRPTW has been tackled using VNS by Bräysy (2003), the vehicle routing problem with multiple depots and time windows (see Polacek et al., 2004) and for real world routing problems (see Polacek et al., 2007a). Other successful implementations include the capacitated arc routing problems with intermediate facilities (Polacek et al., 2007b) and periodic routing problems (Hemmelmayr et al., 2007). Recent successful implementations using VNS include the team orienteering problem (see Archetti et al., 2007), scheduling and flow shop problems (see Blazewicz et al., 2008), nurse rostering (see Burke et al., 2007), car sequencing problems (see Ribeiro et al., 2007) and problems concerning berth allocation (see Hansen et al., 2007).

Hybridization of (meta-)heuristic and exact approaches is new and very active field of research. Prandstetter and Raidl (2007) successfully implemented a hybridization combining VNS and integer linear programming for solving the car sequencing problems. Integer Linear Programming techniques are used within a general VNS framework to explore large neighborhoods. Blum (2005) combined Ant Colony Optimization (ACO) with beam search for solving problems related to open shop scheduling. Furthermore ACO has been combined with (modified versions) of classic algorithms such the one by Wagner Within and Silver Meal for solving lot-sizing problems (Pitakaso et al., 2007). Also multi-level capacitated lot-sizing problems have been solved using a hybrid approach. The problem itself is decomposed into subproblems using ant system. These can be solved using a solver such as CPLEX (see Pitakaso et al., 2006). The final solution itself can be obtained by wisely combing partial results.

Our solution approaches emphasize on a hybridization of well known and efficient methods. Methods that could be applied solely are combined in order to overcome the major drawbacks and concentrate their strengths. Inspired by a combination of VNS (see Mlade-

nović and Hansen, 1997; Hansen and Mladenović, 2001) and Local Branching (see Fischetti and Lodi, 2003; Hansen et al., 2006) we solve the problem based on the VRP* formulation. An exact approach based on a MCNF formulation (see Glover et al., 2003; Ahuja et al., 1993) is guided by means of VNS to quickly find good solutions. Both methods used solely are capable of solving such problems. However, only a cooperative hybrid approach enables us to efficiently combine the strengths of both techniques and compensate for their major drawbacks.

2.2. Real World Issues

In reality however there exist some additional restrictions concerning recipes of concrete. First of all concrete is not a heterogeneous product, rather a multitude of different recipes is available. Even if the same recipe would be used, not all plants may produce the exactly same type of concrete. Some constructors may require the entire amount of concrete to be delivered from the very same plant. According to legal restrictions the resulting schedules might also be feasible in terms of working hours for the corresponding drivers. So far these requirements are not going to be considered at all. However the model itself could easily be extended to incorporate these types of conditions as well.

Due to its chemical characteristics concrete has a limited life time. It is perishable in a sense that it starts to harden until it forms a firm building material. Traffic jams and delays during the delivery process involve a certain risk. Usually especially designed chemical substances are used for controlling the hardening process or even delay it if necessary. Temperature and humidity are among some factors that affect the hardening process. But regardless of all chemical ingredients and trimmings concrete cannot be transported for a very long time.

The perishability itself is considered implicitly by means of a preprocessing stage taking place. Delivering concrete from a plant to a construction site where the resulting traveling time is above two hours is not permitted.

The problem under consideration here is supposed to be deterministic. All orders are supposed to be known at the start of the optimization process. In reality however it is very likely for orders to arise or change during the day. Changes that one might encounter refer to the exact amount of concrete to be demanded and the specific time window when unloading should start. It is not always straight forward for constructors to estimate the real demand of concrete per day beforehand. Concrete plants and their dispatchers are facing short-term modifications relating to the exact demand of concrete to be delivered.

This thesis however concentrates on a deterministic version where all instance related input parameters are known beforehand and do not change dynamically. We decided to tackle the deterministic version before considering the dynamic case. All knowledge gained while working on the deterministic version will help us to overcome problems associated with dynamic characteristics in the future. The solutions obtained by the deterministic version are of good quality and the required run times are reasonable. Any solution obtained could easily be used as a starting solution for reoptimizing the problem under consideration in case input data changed.

3. Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows (VRP*)

Exact methods based on a formulation for the Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows (SDMDHVRPTW) have been implemented. Our formulation additionally takes into account some problem specific characteristics such as special equipment that may be necessary. Trucks providing unloading equipment need to arrive first at the construction site and assist (i.e. stay) later arriving trucks with their unloading operation. Gaps between consecutive unloading operations are supposed to be small and will be penalized in the objective function. In order to facilitate readability, this problem will be referred to as VRP*.

Our formulation deviates from the classical MDVRPTW formulation. Customers place their orders. An order typically cannot be satisfied by just one vehicle. Rather more trucks need to be scheduled to arrive one after each other to satisfy the demand. One order splits into various deliveries. However the exact number of deliveries to be executed is not known beforehand. It depends on the capacity of vehicles assigned to deliveries for any certain order. Similarly the time windows are not clearly specified. Only one time window is given which refers to the start of the very first delivery per order. Starting the first delivery before the beginning of the time window is not permitted, any late start will be penalized in the objective function. 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.

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 basically commute between plants and construction sites corresponding to orders that are going to be served. Loading operations of trucks are not restricted to their home plants. Trucks may also be loaded at plants other than their 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 an orders construction site. This it not necessarily due to perishability. Rather concrete is such a heterogeneous product with various different kinds of recipes.

Figure 3.1 depicts the situation that is going to be modeled. Trucks leave their home plant and visit several customers (i.e. execute deliveries) throughout the day. By the end of the day they need to return back to their home plant where they started their tour from. The first loading operation is executed at the corresponding home plant. Between executing two deliveries trucks need to be loaded. However the loading operation itself is not explicitly shown in this graphic. In our case trucks are supposed to be loaded at the closest plant en route between the corresponding construction sites.

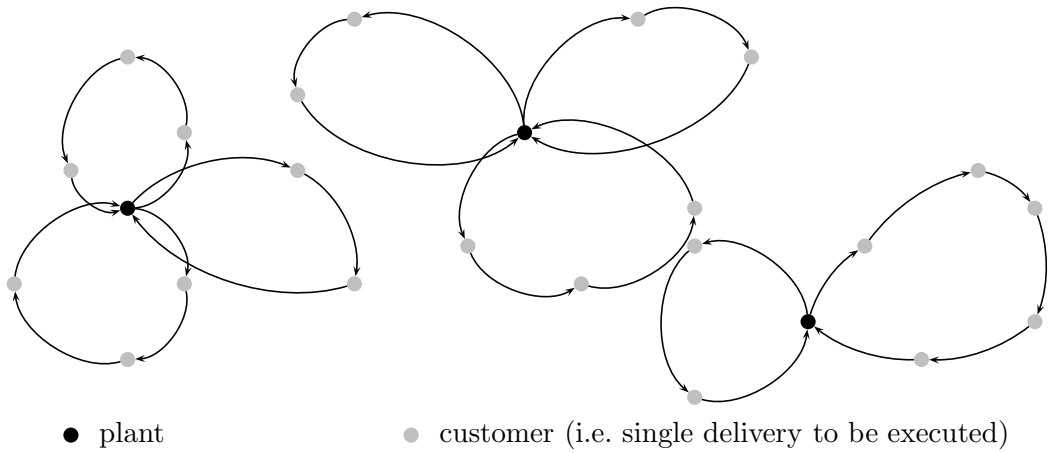

Figure 3.1.: Representation as VRP* (discarding loading operations)

To completely satisfy an order usually several truck loads are necessary, as the demanded quantity typically exceeds the capacity of any single truck. It is not known beforehand how many deliveries are needed to satisfy any order. Appropriate bounding strategies have been developed in order to overcome this problem and facilitate the optimization process. The minimum number of deliveries required is calculated in a sense that it is assumed that all unloading operations are executed by the largest truck available. Therefore the minimum number of deliveries needed in to satisfy order o with a demand of Q_o is given by $\lceil Q_o / cap_{max} \rceil$, where cap_{max} denotes the capacity of the largest truck available. In case the corresponding order requires special unloading equipment 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 $\lceil max(0, Q_o - cap_{max}^{oinstr_o})/cap_{max} + 1 \rceil$ deliveries are needed, where $cap_{max}^{oinstr_o}$ refers to the capacity of the largest truck being equipped with the instrumentation demanded by order o.

To strengthen the bounds and get an upper bound for the maximum number of deliveries necessary we consider the worst case where all deliveries are executed by trucks with the smallest capacity available. Only trucks that are equipped with a real loading space and have a non-zero capacity are considered in this case. The maximum number of deliveries necessary is given by $\lceil Q_o/cap_{min} \rceil$, where cap_{min} denotes the smallest capacity given any truck available. In case an order requires special unloading equipment the maximum number of unloading operations required is increased by one, taking into account a truck with special equipment that might need to arrive first, which possibly has no loading capacity at all.

This approach has been introduced in Schmid et al. (2007a), where an overview on the problem description, the development of bounds and the resulting integrative hybrid approach are described in more detail.

Section 3.1 gives an overview on the implemented model formulation. For getting good lower bounds several valid inequalities have been developed. Two different methods for adding them intelligently are presented in Section 3.2. Inspired by strategies based on LB and VNS imposed by Fischetti and Lodi (2003) and Hansen et al. (2006) we implemented an integrated hybrid approach for guiding the optimization process of the MIP formulation. Our approach is able to quickly generate good and feasible solutions. In the long run the implemented approach is capable of providing high quality and competitive solutions. An overview on the obtained solutions and bounds can be found in Section 8.2, 8.3 and B.1.

3.1. Model Formulation

A binary decision variable $z_{o,d}$ serves as an indicator whether or not a certain delivery d associated with order o is supposed to be executed. In case the d-th delivery associated with order o is going to be executed, $z_{o,d}$ is equal to 1, and 0 otherwise. The very first delivery per order is referred to as $d = 1$.

Trucks and their behavior respectively need to be linked to the deliveries to be executed. Hence the following decision variable is introduced: The binary decision variable $y_{o,d}^k$ indicates whether a certain truck k is going to execute delivery d of order o. $a_{o,d}$ ($b_{o,d}$)

refer to the start (end) of the unloading operation associated with delivery d of order o. $late_o$ 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 $[s_o, e_o]$, the decision variable will evaluate the length of the resulting gap. It captures late starts only. In case the first delivery starts within the time window it will be equal to 0. Starting the very first delivery before the start of the given time window is not permitted.

All truck movements are modeled in terms of three different types of decision variables. $xo2o_{o_1,d_1}^{o_2,d_2,k}$ is a binary decision variable evaluating to 1 if truck k, after having executed delivery d_1 of order o_1, will serve delivery d_2 of order o_2. When going from o_1 to o_2 truck k will drive from the construction site associated with order o_1 to a plant, it is going to be loaded there and finally drives to the construction site associated with order o_2. The choice of the plant where the loading operation in between is going to be executed is not left as an option to the model itself. Rather the plant is chosen deterministically. Between serving any two orders o_1 and o_2 the truck is going to be loaded at the closest plant en route, when going from the construction sites associated with order o_1 to o_2 respectively. For the very first and very last unloading operation per day $xp2o_{o,d}^k$ and $xo2p_{o,d}^k$ model the corresponding behavior respectively. $xp2o_{o,d}^k$ refers to the very first movement of truck k per day and will be equal to 1 if the first delivery to be performed along the planning horizon refers to executing delivery d of order o. Truck k will start its daily tour from its home plant, where it is also going to be loaded. Analogously $xo2p_{o,d}^k$ refers to the very last movement of truck k per day. In case truck k will go back to its home plant immediately after having served delivery d of order o, the associated decision variable will be equal to 1.

The aim of this optimization model is to minimize the total time traveled. Alternatively any other distance related measure could be used. Additionally gaps between consecutive deliveries or starting the first delivery of any order o after the end of the associated time window are tried to be avoided and will be penalized accordingly using β as penalty value. Formula 3.1 depicts the objective function. The travel time for going from truck k's home plant p_k to the construction site associated with order o and vice versa is denoted by $TT_{p_k,o}$ and TT_{o,p_k} respectively. The travel time necessary for a truck to drive from the construction site associated with order o_1 to order o_2 is denoted by TT_{o_1,o_2}. This does not include the time necessary for traveling directly. Rather it is made up of time necessary for driving to the closest plant en route and the time necessary for driving to the construction site associated with order o_2. The time for the loading operation itself however is not

included as part of the objective function.

$$\begin{aligned}\min \sum_{\substack{o_1,o_2\in O\\ d_1\in D_{o_1}\\ d_2\in D_{o_2}\\ k\in K}} xo2o_{o_1,d_1}^{o_2,d_2,k}\cdot TT_{o_1,o_2} + \sum_{\substack{o\in O\\ d\in D_o\\ k\in K}} xp2o_{o,d}^{k}\cdot TT_{p_k,o} + \sum_{\substack{o\in O\\ d\in D_o\\ k\in K}} xo2p_{o,d}^{k}\cdot TT_{o,p_k} + \\ \beta\sum_{o\in O}(late_o + \sum_{\substack{d\in D_o\\ d>1}}(a_{o,d}-b_{o,d-1}))\end{aligned} \tag{3.1}$$

In case a certain delivery is actually going to be executed (see associated decision variable $z_{o,d}$), one has to ensure that one of the trucks k is going to be assigned for executing this particular delivery. If $z_{o,d}$ evaluates to 1 exactly one truck needs to be assigned to executing delivery d of order o.

$$z_{o,d} = \sum_{k\in K} y_{o,d}^{k} \qquad \forall o\in O, d\in D_o \tag{3.2}$$

Due to symmetry reasons the very first delivery associated with any order needs to be executed by all means. Additionally consecutive deliveries may need to be executed. All deliveries starting from the very first ones need to be executed until the total amount of concrete demanded is met. Once a delivery is skipped no further deliveries can take place any more.

$$z_{o,d} \leq z_{o,d-1} \qquad \forall o\in O, d\in D_o, \text{where } d>1 \tag{3.3}$$

If truck k gets assigned to the execution of delivery d associated with order o, it somehow needs to get there. A truck can either get there directly from its home plant by means of executing its first delivery, or after having served any other delivery d_1 of order o_1.

$$y_{o,d}^{k} = xp2o_{o,d}^{k} + \sum_{\substack{o_1\in O\\ d_1\in D_{o_1}}} xo2o_{o_1,d_1}^{o,d,k} \qquad \forall o\in O, d\in D_o, k\in K \tag{3.4}$$

Additionally the following relationship needs to hold: In case truck k has to execute a certain delivery it also needs to leave the corresponding construction site and go somewhere afterwards. Either the truck just finished its last unloading operation and returns back home to its home plant. Alternatively the truck may serve any other delivery d_2 associated with order o_2 afterwards. The corresponding constraint looks like this:

$$y_{o,d}^{k} = xo2p_{o,d}^{k} + \sum_{\substack{o_2\in O\\ d_2\in D_{o_2}}} xo2o_{o,d}^{o_2,d_2,k} \qquad \forall o\in O, d\in D_o, k\in K \tag{3.5}$$

Some logical constraints also need to hold. A truck is allowed to leave its home plant (and execute its very first unloading operation per day) at most once (see Equation 3.6). Equation 3.7 states that in case a truck leaves its home plant it also needs to return there by the end of the day.

$$\sum_{\substack{o \in O \\ d \in D_o}} xp2o^k_{o,d} \leq 1 \qquad \forall k \in K \tag{3.6}$$

$$\sum_{\substack{o \in O \\ d \in D_o}} xp2o^k_{o,d} = \sum_{\substack{o \in O \\ d \in D_o}} xo2p^k_{o,d} \qquad \forall k \in K \tag{3.7}$$

Usually trucks execute more then just one single delivery per day. To ensure a feasible solution, the time difference between two consecutive unloading operations to be executed by the same truck needs to be big enough. There needs to be enough time for the truck to drive to the closest plant en route and for being loaded there. Usually trucks are free to leave a construction site immediately after having finished their unloading operation. Trucks bringing along special instrumentation and assisting other trucks during their own unloading operations may only leave the construction site after the order's demand has been met completely and the last truck has finished its unloading operation. It needs to stay and assist later arriving trucks with their unloading operation respectively.
Note that there are two cases that need to be distinguished: The first set of constraints (see Equation 3.8a) refers to the general case. The second set (see Equation 3.8b) however has been designed specifically for orders requiring special instrumentation and the trucks that are equipped accordingly. In this case the first truck to arrive is not allowed to leave the construction site immediately having finished its unloading operation. Rather it needs to stay there until the last truck has finished its unloading operation. The end of the last unloading operation is denoted by $b_{o_1,|D_{o_1}|}$.

$$a_{o_2,d_2} \geq b_{o_1,d_1} + xo2o^{o_2,d_2,k}_{o_1,d_1} \cdot TTL^k_{o_1,o_2} + M \cdot (1 - xo2o^{o_2,d_2,k}_{o_1,d_1}) \\ \forall o_1, o_2 \in O, d_1 \in D_{o_1}, d_2 \in D_{o_2}, k \in K, \text{where } \{o_1 \notin O' \vee d_1 \neq 1\} \tag{3.8a}$$

$$a_{o_2,d_2} \geq b_{o_1,|D_{o_1}|} + xo2o^{o_2,d_2,k}_{o_1,d_1} \cdot TTL^k_{o_1,o_2} + M \cdot (1 - xo2o^{o_2,d_2,k}_{o_1,d_1}) \\ \forall o_1 \in O', o_2 \in O, d_1 \in D_{o_1}, d_2 \in D_{o_2}, k \in K, \text{where } \{d_1 = 1\} \tag{3.8b}$$

The time necessary for driving from the construction site associated with order o_1 to order o_2 consists of the time necessary for driving to the closest plant en route, being loaded there and driving tot the construction site associated with order o_2. It is referred to as $TTL^k_{o_1,o_2}$.

At most one vehicle may be unloaded at any point in time. All deliveries d relating to the same order o are supposed to be non-overlapping. Any gaps between consecutive unloading operations will be penalized accordingly. One has to ensure that at most one vehicle can unload at any point in time. Hence the following restrictions need to hold.

$$b_{o,d-1} \leq a_{o,d} \qquad \forall o \in O, d \in D_o, \text{where } d > 1 \tag{3.9}$$

The first delivery has to start after the begin of the given time window s_o. A too early start is strictly not permitted.

$$a_{o,1} \geq s_o \qquad \forall o \in O \tag{3.10}$$

The first delivery of any order is supposed to start within a given time window. The late start of the very first delivery associated with order o will be captured by means of decision variable $late_o$. A late start of the first delivery of order o starts will 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 (e_o) of the corresponding time window.

$$late_o \geq a_{o,1} - e_o \qquad \forall o \in O \tag{3.11}$$

In order to ensure that the total quantity demanded will be delivered the following constraint has to hold: The total quantity of concrete ordered by order o is referred to as Q_o. One has to ensure that the cumulative capacity of all trucks serving order o may not fall below the ordered quantity Q_o.

$$\sum_{\substack{d \in D_o \\ k \in K}} cap_k \cdot y^k_{o,d} \geq Q_o \qquad \forall o \in O \tag{3.12}$$

In case special unloading equipment is demanded by an order the truck to execute the first delivery needs to be equipped accordingly. The type of special equipment demanded by order o and the type of instrumentation truck k is equipped with are referred to as $oinstr_o$ and $tinstr_k$ respectively.

$$\sum_{k \in K} tinstr_k \cdot y^k_{o,1} = oinstr_o \qquad \forall o \in O' \tag{3.13}$$

The time required for fully unloading truck k at the construction site associated with order o is denoted by U^k_o. The length of any delivery is determined by the following set of

constraints.

$$b_{o,d} - a_{o,d} \geq U_o^k \cdot y_{o,d}^k \qquad \forall o \in O, d \in D_o, k \in K \tag{3.14}$$

$$b_{o,d} - a_{o,d} \leq U_o^k + M(1 - y_{o,d}^k) \qquad \forall o \in O, d \in D_o, k \in K \tag{3.15}$$

3.2. Finding good Lower Bounds for the VRP⋆-Formulation

As expected the attempt "to MIP" the problem was not really working successfully. As a first step therefore we decided to work on the lower bounds. Instead of inconsiderately taking any (probably too) large number we try to find good values for the M's used by constraints defined in Section 3.1. Solving the relaxed problem as such already gives some kind of bound. Several sets of valid inequalities have been developed to improve the lower bound ever further. Moreover they can be used to guide the solution process of the relaxed problem formulation. These valid inequalities also have been proven to be highly useful when it comes to adding cuts and constraints as cutting planes. This section is dedicated to the implemented cuts. After an overview on the valid inequalities found, we will focus on how and where to use them to quickly obtain good lower bounds for the problem at hand. Some of the constraints are implemented using the M-method. Hence in order to get tighter bounds we try to find good and feasible values for them.

3.2.1. Getting a small M

Feasibility of solutions from the trucks point of view is ensured by Constraints 3.8a and 3.8b respectively. They do guarantee that enough time is planned for driving from construction sites to orders, being loaded there and driving to the next construction site. These constraints have been implemented by using the so called "Big M-method". Instead of just picking any large number we try to take the smallest number possible instead. Therefore we will be able to get tighter bounds.

The two constraints mentioned only need to become affective in case truck k executes delivery d_2 of order o_2 right after having executed delivery d_1 of order o_1. However as usually in optimization we are not clairvoyant, we do not know a priori in which sequence trucks will execute deliveries. Therefore the constraints need to be imposed for every possible outcome. Without loss of generality all possible situations need to be handled, even situations where the above mentioned situation is not true.

The total planning horizon per day might be one reasonable value for the M used within those constraints. It can further be improved by evaluating the maximum time distance between the end (b_{o_1,d_1}) of any delivery d_1 of the associated order o_1 and the start (a_{o_2,d_2})

of delivery d_2 for order o_2.

At the earliest any delivery for order o can start at s_o (the start of the associated time window). Any delivery starting to be executed prior to this point will result in an infeasible solution. Suppose all previous deliveries d' (where $d' < d$) have been executed by the smallest truck at hand the earliest start for delivery d associated with order o is given by:

$$\underline{a_{o,d}} = \begin{cases} s_o + \lceil \frac{cap_{min}}{UR_o} \rceil \cdot (d-1) & \forall o \in O \setminus O', d \in D_o \\ s_o + \lceil \frac{cap_{min}}{UR_o} \rceil \cdot (d-2) + \lceil \frac{cap_{min}^{oinstr_o}}{UR_o} \rceil & \forall o \in O', d \in D_o, \text{where } d > 1 \\ s_o & \forall o \in O', d \in D_o, \text{where } d = 1 \end{cases} \quad (3.16)$$

The latest end ($max(b_{o,d})$) of any delivery d associated with order o however is not as easy to estimate. Let's assume however that all orders need to be satisfied by the end of the planning horizon. Resulting gaps between consecutive deliveries unpredictably postpone the end of any unloading operation. The planning horizon itself serves as the worst case for the end of any delivery.

Coming back to the "big Ms" for Constraints 3.8a and 3.8b respectively. Without loss of generality a good and feasible value is given by subtracting the two values accordingly. For constraints referring to one and the same order (i.e. where $o = o_1 = o_2$ and $d_1 < d_2$) an even tighter bound can be imposed. In case one and the same truck k is supposed to execute both deliveries d_1 and d_2 of order o the minimum amount of time in between is given by $TTL^k_{o,o}$, including the time for truck k for driving from the construction site associated with order o to the closest plant, being loaded there and driving back to order's o construction site.

For Constraints 3.15 $\lceil cap_{max}/UR_o \rceil$ - the maximum length of any unloading operation, regardless of the truck to execute it - is an appropriate choice.

3.2.2. Valid Inequalities

All valid inequalities have been implemented after solving the relaxed problem based on the VRP* formulation. The solution process will be guided towards a possibly integer solution, by the use of several additional constraints. When solving the MIP the constraints as such would not be needed at all, as the integrality itself would take care of these situations. When solving only the relaxed problem however additional constraints like the following ones help to guide the solution process and improve the quality of the

lower bound. The following section describes typical situations found when looking at the solution of a relaxed problem. The stated valid inequalities are implemented in to avoid exactly those situations and the resulting (integer infeasible) consequences.

If it turns out that for an order requiring special instrumentation ($o \in O'$) only one delivery is going to be executed (i.e. $z_{o,d} = 0$ for $d > 1$), the first delivery cannot be executed by a truck with no capacity. By all means the truck executing the first delivery needs to be equipped accordingly. In case no second delivery is foreseen, the first truck to arrive also needs to bring along enough capacity, otherwise the order's demand cannot be satisfied.

$$z_{o,2} \geq y^k_{o,1} \qquad \forall o \in O', k \in K, \text{where } cap_k = 0 \tag{3.17}$$

Within any feasible integer solution a truck after leaving an order for good cannot execute any other deliveries. In case the variable $xo2p^k_{o,d'}$ takes on a positive value - indicating that truck k returns back to its home plant after having executed delivery d' of order o - truck k cannot be foreseen for executing any later deliveries d ($d' < d$) for the very same order o.

$$y^k_{o,d} + \sum_{\substack{d' \in D_o \\ d' < d}} xo2p^k_{o,d'} \leq 1 \qquad \forall o \in O, d \in D_o, k \in K, \text{where } d > 1 \tag{3.18}$$

Trucks that do not leave their home plant cannot be foreseen for executing any single delivery. In case neither of the decision variables $xp2o^k_{o,d}$ for a fixed truck k evaluates to one or at least takes on a positive value, truck k cannot be assigned and hence move to any delivery.

$$\sum_{\substack{o \in O \\ d \in D_o}} xp2o^k_{o,d} \geq \sum_{\substack{o_1, o_2 \in O \\ d_1 \in D_{o_1} \\ d_2 \in D_{o_2}}} xo2o^{o_2,d_2,k}_{o_1,d_1} \qquad \forall k \in K \tag{3.19}$$

To avoid catch-22-alike situations a truck executing the first delivery of an order requiring special instrumentation cannot be scheduled for any other delivery within the same order. The first truck to arrive needs to stay at the construction site associated with order o and assist all later arriving trucks with their unloading operation. The truck is free to leave as soon as the truck scheduled for the last delivery has finished its unloading operation.

$$y^k_{o,d} \leq 1 - y^k_{o,1} \qquad \forall o \in O', d \in D_o, k \in K, \text{where } \{d > 1 \vee oinstr_o = tinstr_k\} \tag{3.20}$$

In case a truck is supposed to execute a delivery, decision variable $y^k_{o,d}$ for a fixed truck k takes a non-zero value, truck k needs to already have left its home plant sometime before.

$$y^k_{o,d} \leq \sum_{\substack{o_2 \in O \\ d_2 \in D_{o_2}: \\ \neg\{o=o_2 \wedge d_2 > d\}}} xp2o^k_{o_2,d_2} \qquad \forall o \in O, d \in D_o, k \in K \tag{3.21}$$

For avoiding subtours from the trucks point of view the following set of constraints has shown to be highly useful. Trucks that just came from their home plant (i.e. $xp2o^k_{o,d}$ takes on a non-zero value) and go back home immediately cannot be foreseen for any other delivery.

$$2 - (xp2o^k_{o,d} + xo2p^k_{o,d}) \geq xo2o^{o_2,d_2,k}_{o_1,d_1} \qquad \forall o, o_1, o_2 \in O, d \in D_o, d_1 \in D_{o_1}, d_2 \in D_{o_2}, k \in K \tag{3.22}$$

In case a truck is supposed to execute two different deliveries d_1 and d_2 (where $d_1 < d_2$) of one and the same order o, 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.

$$b_{o,d_1} \; + \; TTL^k_{o,o} \; \cdot \; (y^k_{o,d_1} + y^k_{o,d_2} - 1) \leq a_{o,d_2} \qquad \forall o \in O, d_1, d_2 \in D_o, k \in K, \text{where } d_1 < d_2 \tag{3.23}$$

Trucks are not allowed to commute between two deliveries. In case decision variable $xo2o^{o_2,d_2,k}_{o_1,d_1}$ takes on a non-zero value its counterpart $xo2o^{o_1,d_1,k}_{o_2,d_2}$ cannot take on a positive value as well. Rather truck k needs to go somewhere else instead (either return home to its home plant or to any other delivery). Usually Constraint 3.8a and 3.8b should take care about this situation. When solving only the relaxed problem and because of the use of the M-method in the previously mentioned constraints however certain features might not be captured. Even better bounds are achieved by the use of the following set of constraints.

$$xo2o^{o,d,k}_{o_1,d_1} \leq \sum_{\substack{o_2 \in O \\ d_2 \in D_{o_2}: \\ \neg\{o_1=o_2 \\ \wedge d_1=d_2\}}} xo2o^{o_2,d_2,k}_{o,d} + xo2p^k_{o,d} \quad \forall o, o_1 \in O, d \in D_o, d_1 \in D_{o_1}, k \in K \tag{3.24}$$

3.2.3. Cutting Planes

To solve the MIP all constraints stated in Section 3.1 are necessary in order to guarantee a feasible solution. Some families of constraints however result in a very huge number of constraints, most of which turn out to be not binding either. Therefore all families of constraints have been classified as either being fundamental or non-fundamental. Non-fundamental families of constraints typically result in a large number of constraints, mostly due to the number of indices involved. Alternatively they might also result in a large set of constraints, where most of which are non-binding and therefore not necessarily needed to be considered from the very first step on.

The constraints depicted in Equations 3.8a and 3.8b respectively indeed do result in a very large number of constraints due to the five indices involved. Therefore this family of constraints is considered as being non-fundamental. Moreover the families of constraints corresponding to Equations 3.15 and 3.14 also have been classified as being non-fundamental, as in relaxed solutions found most of them result to be non-binding anyway. All remaining familie of constraints are considered as being fundamental.

The initial LP based on the relaxed MIP formulation can be solved easily. The resulting bounds however are extremely week and can be strengthened by adding cuts to the original LP formulation. On demand (i.e. whenever violated) we are generating cuts that cutoff the current (i.e. fractional) solution. Inspired by the fundamental work on cutting planes and integer programming in Gomory (1958, 1960) we developed two different approaches for obtaining good lower bounds. For a more comprehensive overview on cutting planes and their application to integer and mixed integer programming the reader is referred to Schrijver (1980) and Marchand et al. (2002). When solving the model fundamental constraints only will be considered. All non-fundamental constraints and valid inequalities will be added on demand, i.e. when discovered to be violated. The main difference between the two variants is made up by *how* violated valid inequalities and non-fundamental constraints are going to be dealt with.

Variant 1: Within this approach the MIP, while imposing only fundamental constraints right from the beginning, is going to be solved once. Instead of using the standard cut manager embedded within XPRESS-MP we defined our own callback function. This routine is going to be called at each node in the tree. At every node the current solution is examined in terms of its actual feasibility. On request violated non-fundamental constraints, as well as violated valid inequalities are going to be added as *cuts*. The cut manager routine

will be called repeatedly at each node until no more cuts have been added. The resulting sub-problem is automatically optimized if any cuts have been added. An algorithmic description of the embedded customized cut manager is shown in Algorithm 3.1. Any cuts added are only good for the corresponding node where they have been generated, as in its descendant child nodes.

Algorithm 3.1 Callback Function for Cut Manager

```
nV ← number of violated valid inequalities and non-fundamental constraints
if nV > 0 then
    add violated non-fundamental constraints and valid inequalities as cuts
    optimize resulting sub-problem again
end if
```

Variant 2: A different framework has also been tested, within which the resulting relaxed problem is iteratively going to be solved. Again, only fundamental constraints are considered right from the beginning. The solution obtained after having solved the relaxed problem is examined in terms of potential infeasibilities. Violated constraints and valid inequalities are added to the model as *constraints* and the updated relaxed formulation is going to be solved again. An outline of this procedure can be found in Algorithm 3.2. As opposed to the previous variant all added constrains are globally valid. Once they have been added the have to be observed. The remaining run time is spent on solving the formulation as MIP. The same callback function is used for the cut manager, as probably not all of the non-fundamental constraints already have been added. To ensure feasibility, some more cuts might still need to be added.

Algorithm 3.2 Iterative Relaxation

```
set up the VRP* with fundamental constraints only
violated ← true
while violated and terminationCriteria not met do          ▷ Loop over LP
    solve relaxed MIP (LP)
    nV ← number of violated valid inequalities and non-fundamental constraints
    if nV > 0 then
        add violated valid inequalities and non-fundamental constraints as constraints
    end if
end while
solve MIP                                                  ▷ use remaining run time
```

4. Integrated Hybrid Approach for Solving VRP*

As expected solving the problem based on the VRP* formulation as a MIP, using any 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 LB (see Fischetti and Lodi, 2003) and inspired by VNS is used to assist the solver during its optimization.

This chapter gives a short overview on the methodology used. Starting with a basic outline of the framework we will focus on the problem specific knowledge in use and the resulting implicit hierarchic structure. Finally the shaking operator in use will be specified. All results obtained using this approach can be found in Section 8.4 and 8.10. A more detailed overview is given in Section B.2.

4.1. Basic Outline

The following approach tries to systematically *improve* any given feasible solution. In principle the integrated hybrid approach for solving the VRP* formulation is inspired by ideas coming from VNS. For a more detailed description of VNS please see Chapter 6, and Mladenović and Hansen (1997) and Hansen and Mladenović (2001) respectively. 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 respectively. Then the problem is solved again using any general purpose solver used as a black box. Only solutions that improve the current best 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 4.1.

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. Rather we keep most of them unchanged. Only *certain* decision variables are allowed to change. The choice which binary decision variables are allowed to flip their value depends on the *neighborhood structure* currently in use.

Algorithm 4.1 Basic outline of LB using VNS

```
x ← any (feasible) initial solution
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
        if accept(x') then          ▷ move or move not? (only accept better solutions)
            x ← x'
            κ ← 1                                  ▷ continue with 1st neighborhood
        else
            κ ← κ + 1                              ▷ continue with next neighborhood
        end if
    end while
end while
```

The VNS-based approach described in Chapter 6 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. 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 think about which truck shall 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 be determined by the current neighborhood structure, using some problem specific knowledge. All remaining variables corresponding to the exact timing of deliveries ($a_{o,d}$, $b_{o,d}$), will left unrestricted at all times. The same is true for those decision variables responsible for modeling the movement of trucks ($xp2o^k_{o,d}$, $xo2p^k_{o,d}$, $xo2o^{o_2,d_2,k}_{o_1,d_2}$).

4.3. 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 4.1. Their structure is going to be described in more detail within the following pages.

Table 4.1.: Set of Neighborhood Structures (integrated hybrid approach)

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

The first shaking operator (`FreeDelivery`, responsible for neighborhood $\kappa = 1 \dots 3$) no longer fixes the decision concerning which truck k is supposed to execute a certain delivery. Given the current solution the assignment of trucks to deliveries in up to $10\kappa\%$ cases is discarded. An algorithmic description of this shaking operator can be found in Algorithm 4.2.

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. Hence only orders are going to be considered, which are currently served by more deliveries then necessary (for the calculation of the lower bound on the number of deliveries per order see page 14). For all orders selected the last delivery is going to be 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. Algorithm 4.3 sketches the outline of this shaking operator.

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 then 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. (The calculation of the upper bound on the number of deliveries required can also be found on page 14). A short overview on this shaking operator can be found in Algorithm 4.4.

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.

Algorithm 4.2 Shaking Operator 1: `FreeDelivery`

Require: $1 \leq \kappa \leq 3$
  $D' \leftarrow$ random choice of $10\kappa\%$ of deliveries executed by current solution
  for all $d' \in D'$ **do**
    $o \leftarrow$ order corresponding to delivery d'
    $d \leftarrow$ delivery index corresponding to delivery d'
    $k \leftarrow$ truck currently executing delivery d'
    unfix $y^k_{o,d}$ ▷ set lower bound to 0
    for all $k' \in K$, where $k' \neq k$ **do**
      unfix $y^{k'}_{o,d}$ ▷ set upper bound to 1
    end for
  end for

Algorithm 4.3 Shaking Operator 2: `SkipOneDelivery`

Require: $4 \leq \kappa \leq 6$
  $O'' \leftarrow$ random choice of $10(\kappa - 3)\%$ of orders that could be served by one delivery less
  for all $o \in O''$ **do**
    $d \leftarrow$ last delivery currently executed for order o
    fix $z_{o,d}$ ▷ set upper bound to 0
    $k \leftarrow$ truck currently executing delivery d of order o
    fix $y^k_{o,d}$ ▷ set upper bound to 0
    $d \leftarrow$ randomly choose any other delivery of order o ▷ bias towards smaller trucks
    $k \leftarrow$ truck currently executing delivery d of order o
    unfix $y^k_{o,d}$ ▷ set lower bound to 0
    for all $k' \in K$, where $k' \neq k$ **do**
      unfix $y^{k'}_{o,d}$ ▷ set upper bound to 1
    end for
  end for

Algorithm 4.4 Shaking Operator 3: `AddOneDelivery`

Require: $7 \leq \kappa \leq 9$
  $O'' \leftarrow$ random choice of $10(\kappa - 6)\%$ of orders that could be served by one more delivery
  for all $o \in O''$ **do**
    $d \leftarrow$ last delivery currently executed for order o
    unfix $z_{o,d+1}$ ▷ set upper bound to 1
    for all $k \in K$ **do**
      unfix $y^k_{o,d+1}$ ▷ set upper bound to 1
    end for
    $d \leftarrow$ randomly choose any other delivery of order o ▷ bias towards larger trucks
    $k \leftarrow$ truck currently executing delivery d of order o
    unfix $y^k_{o,d}$ ▷ set lower bound to 0
    for all $k' \in K$, where $k' \neq k$ **do**
      unfix $y^{k'}_{o,d}$ ▷ set upper bound to 1
    end for
  end for

5. Multi-Commodity Network Flow Formulation

The RMC delivery problem can be modeled in terms of an integer multi-commodity network flow problem (MCNF) on a time-spaced method. This approach has been inspired by the model proposed in Hoffman and Durbin (2007) and Durbin (2003). Nodes represent construction sites and plants at various points in time. Different nodes indicate potential start (and end) points of loading operations at all plants respectively. Similarly from the construction site's point of view, nodes refer to the potential start (and end) of unloading operations. The flow refers to the movements of trucks trough the network. Trucks with the same capacity, home plant and equipment are aggregated into classes and handled as one single type of commodity. The resulting model formulation has been introduced in Schmid et al. (2007d).

All modeling approaches presented within this Chapter are based on a MCNF formulation. Two different formulations will be presented within this chapter, both of which may be used to solve problems concerning the delivery of concrete.

The first formulation (see Section 5.2) skips the choice *where* trucks are supposed to be loaded before being able to deliver concrete to any construction site. Plants where loading operations are supposed to take place are chosen automatically. Not necessarily the closest plant of a construction site is chosen. Rather, in case the vehicle is going from one construction site to any other one, the closest plant en route is chosen. Capacity restrictions concerning the maximum number of vehicles to be loaded at a time are not going to be enforced within this simplified model version. The very first loading operation of any truck is always supposed to be executed at the home plant of the corresponding truck, hence the very first movement per truck initiates at its home plant. As the truck also needs to return back to its home plant by the end of the planning horizon the last movement will terminate there. Throughout the day however the movement of trucks is modeled in terms of orders. Trucks are said to move between orders and their corresponding construction sites. Time for driving to the closest plant en route plus the time for loading

there will be considered implicitly.

The second approach however (see Section 5.3) models all loading operations explicitly. It also takes into account the choice of the plant where vehicles are going to be loaded. The choice of the plant where any loading operation is supposed to take place is not predefined, but rather left to the optimization model itself. This formulation is extremely useful when the capacity at plants is limited. This additional degree of freedom especially makes sense when certain loading restrictions (i.e. the maximum number of trucks that can be loaded at a plant at any time) have to be considered. In this case the movement of trucks is modeled in terms of going back and forth between plants and construction sites associated with any order. Again, trucks start and end their daily tours at their home plants.

Within both formulations trucks are aggregated in terms of classes, whereas one class is considered as one single commodity. Trucks within the same class have the same capacity as well as instrumentation for unloading and are located at the very same home plant.

Both formulations are *not* extensive in a sense that *all* possibilities on how unloading operations could take place (i.e. which trucks are going to execute them, how many trucks are needed to completely satisfy an order, in which sequence are trucks supposed to unload). Rather a limited number of so called *patterns* are 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. 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.

This chapter is organized as follows. A detailed overview on patterns, their significance and generation is given in Section 5.1. Patterns serve as input for the MCNF formulation. The models themselves are supposed to choose one pattern per order and determine all resulting types of vehicle movements and all subsequent loading and unloading operations. The two different approaches are presented in Section 5.2 and 5.3 respectively. An overview on the results obtained can be found in Section 8.6. For a comparison of the two formulations the reader is referred to Section 8.5. A comprehensive outline can be found in Section B.3 and B.4.

5.1. Base Patterns

Rather than taking into account all possibilities *when*, *where* and *which* truck is (possibly) supposed to execute a certain delivery and the resulting consequences thereafter, only a limited number of patterns per order will be considered. Only given possibilities for every single order - so called patterns - will be considered on how unloading operations could take place, whereas the execution of all deliveries belonging to a pattern would completely satisfy one order. The aim of the resulting optimization models (see Section 5.3 and 5.2) consists of selecting at most one pattern per order, hence determining the exact timing of all resulting loading and unloading operations, as well as the movements of trucks between plants and orders, given certain patterns have been chosen to be executed.

Any pattern a for order o uniquely specifies all necessary unloading operations, such that the demand will be completely satisfied. The sequence and types of trucks to execute single deliveries, as well as the point in time when unloading is supposed to start will be specified. Additional requirements concerning unloading equipment will also be considered.

If an unloading operation for order o of a truck of class c is supposed to start in t according to fulfillment pattern a, the corresponding binary indicator $P_{o,a}^{c,t}$ will be equal to 1. The start (end) of the very first (last) unloading operation is denoted by $start_{o,a}$ ($end_{o,a}$), the truck which is supposed to execute the first unloading operation is denoted as $first_{o,a}$. Every pattern itself is feasible in a sense that subsequent unloading operations do not overlap, the cumulative capacity of all scheduled trucks is able to meet the demand and the first truck scheduled brings along special equipment if required by the order.

The set of pattern for any order o is denoted as A_o. A pattern is supposed to be feasible if the first unloading operation does not start before the beginning of the time window given for the corresponding order and all consecutive unloading operations are non-overlapping. Moreover the accumulated capacity of all trucks scheduled for unloading operations within any single pattern have to be able to meet the required demand. In case special unloading equipment is required for order o the first truck scheduled within any pattern $a \in A_o$ needs to be equipped accordingly.

A graphical representation of a valid pattern a for order o is depicted in Figure 5.1. A total number of 3 unloading operations is foreseen within this particular pattern a. The accumulated capacity of trucks c_1, c_2 and c_3 is sufficient such that the total demand can be satisfied. The first unloading operation is supposed to be executed by a truck of class c_1. The second and third unloading operation would have to be executed by a truck of class c_2 and c_3 respectively. All unloading operations are non-overlapping. There are

no gaps between consecutive unloading operations, as the unloading operation d starts immediately after operation $d-1$ is finished. The trucks serving an order within pattern a do not need to be chosen uniquely. Any type of truck (in this case c_1, c_2 and c_3) can be foreseen (even several times within the same pattern) for executing unloading operations.

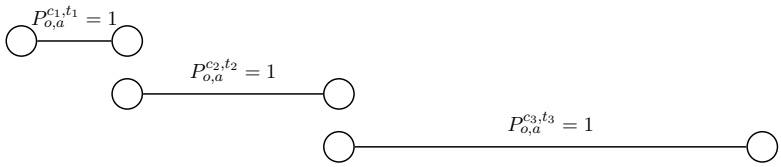

Figure 5.1.: Valid Pattern

Figure 5.2 also depicts a valid pattern. In this case there is a gap between two consecutive unloading operations. If such a pattern is going to be chosen it will be penalized accordingly. In this case there is a gap between the second and third unloading operation as the third unloading operation executed by a truck of class c_3 does not follow up the end of the previous unloading operation. In case this pattern would be chosen for order o a penalty accrues for the resulting gap.

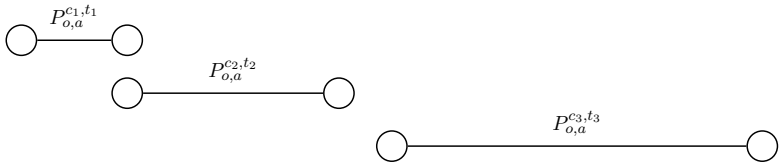

Figure 5.2.: Valid Pattern with Gap

In case special unloading equipment is needed, the first truck c_1 to arrive needs to bring along the demanded equipment. This truck first executes its own unloading operation and then needs to stay at the construction site (dotted line) and assist later arriving trucks with their unloading operations respectively. The truck is allowed to leave after the last truck has finished its unloading operation. Note that (see Section 2.2) trucks assisting others with their unloading operation cannot be released or replaced by any other truck, even if it would be equipped accordingly. The resulting rearrangement of trucks at the construction site would take too long and is undesired and hence not executed in practice.

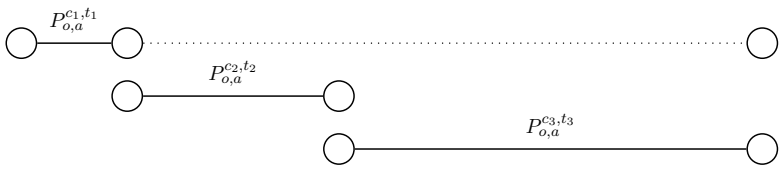

Figure 5.3.: Valid Pattern with Special Equipment

5.1.1. Generation of Base Patterns

Two different methods have been developed for generating base patterns which will be first fed into the available pool of patterns before starting the execution of MCNF for the very first time.

Brute Force

The first and rather myopic version is based upon a brute force approach. A comparatively large number of patterns are generated for every order. These are feasible from the orders point of view, but do not take into account all consequences concerning the resulting movements of trucks. As a first step a number of x base patterns - preferably without any gaps in between - will be generated. Patterns are generated sequentially, i.e. the type of truck foreseen for the first unloading operation and the point in time when its unloading operation is supposed to start will be determined first. Then further trucks will be added to the pattern sequentially until the total quantity delivered satisfies the required demand. All patterns generated for one order have to be unique.

The selection probability for choosing the first truck is proportional to the number of trucks available within a given class of trucks, taking into account requirements for unloading equipment. The first truck is supposed to start its unloading operation at the beginning of the time window. All subsequently scheduled trucks are supposed to start their unloading operation immediately after the previous one finished its unloading operation. The selection probability again is directly proportional to the number of trucks within a given class, but also taking into account if a truck can be scheduled next. The selection probability is inversely proportional to the length of the resulting gap.

After having generated x base patterns for every order the pool of patterns will expanded to $10 \cdot x$ patterns by adding gaps to existing patterns. In order to insert a gap into an existing pattern, a pattern and a delivery within will be chosen. The sequence of all trucks remains unchanged. Just an artificial gap will be inserted and the timing will be adjusted accordingly. The resulting pattern - which still needs to be unique per order - will then

be added into the pool of patterns.

This approach has been chosen for quickly generating a large number of possible options, while not taking into account the integration between various orders and the consequences upon the feasibility on patterns of different orders. Additional delays will be added on purpose. In case the resources - in terms of the capacity of trucks available - are limited and the schedule is tight this might be the only way to satisfy every order.

Compatible Base Pattern Generation

Contrary to the approach described above a more intelligent approach has been developed, also taking into account interdependence between patterns of different orders. Instead of randomly generating a huge number of patterns, less but more sophisticated patterns will be generated. Step by step one pattern per order is going to be generated. Note that all patterns generated at the same step do fit together and result in a feasible solution. Again patterns are constructed sequentially. The unloading operations of the first truck will be determined randomly, close to the time window given. The selection probability of trucks is directly proportional to the number of trucks available within any given class of trucks, the capacity of the trucks and inversely proportional to the distance from the truck's home plant to the construction site of the order. Gaps are tried to be kept as small as possible, if necessary at all.

5.2. Reduced Mathematical Formulation

The problem is modeled as an *integer multi-commodity flow problem* on a time-space network (with some similarities to the model proposed by Hoffman and Durbin (2007) and Durbin (2003). Each type of delivery truck is modeled as a separate commodity. Trucks with same capacity, home plant, and instrumentation are grouped into classes and considered as one single commodity.

Usually, trucks are free to leave a construction site immediately after having finished their unloading operation. However, if an order requires specialized unloading equipment, the first truck to arrive needs to supply the required equipment and remain at the site until the entire order has been fulfilled and the last truck has finished its unloading operation. That truck may depart as soon as the last unloading operation of a fulfillment pattern has been completed.

Only movements between orders and their corresponding constructions sites and all unloading operations taking place are modeled explicitly. All loading operations as well as

the choice of the plant where a loading operation could take place are implicitly given. As the capacity of the plants is supposed not be limited, the closest one will be chosen. When going from order o_1 to o_2 the loading operation will take place at the closest plant en route.

The objective of the optimization consists of minimizing traveling time as well as gaps. Gaps might occur either as a consequence of delays between consecutive unloading operations scheduled at the same construction site or because of starting with the very first unloading operation after the end of the corresponding time window.

Three different types of decision variables model the movements of trucks. The number of trucks of class c starting with their loading operation at their home plant in t for driving to the construction site corresponding to order o afterwards is denoted by the binary decision variable $moveP2O_o^{c,t}$. Similarly $moveO2P_o^{c,t}$ denotes the number of trucks of class c leaving the construction site corresponding to order o at time t, to travel back to their home plant. Please note that this decision variable is integer and not binary as the remaining ones for modeling truck movements. Two trucks of the same class might leave a construction site at the very same time in order to go back to their home plant. These two types of movements correspond to the very first and last movement of trucks per day, either coming or going back to their home plants. All other movements throughout the day are modeled by means of the decision variable $moveO2O_{o_1,o_2}^{c,t}$, which indicates the number of trucks of class c leaving the construction site associated with order o_1 to get to order o_2 in time, also being loaded meanwhile. The number of trucks of a given class c not leaving their home plant at all is denoted by $stayHome_c$.

The objective function Z (see Equation 5.1) basically consists of three terms. First of all we wish to minimize the total travel times. An additional term β_1 penalizes gaps between consecutive unloading operation or the delayed start of any delivery. Finally a (comparatively high) penalty term β_2 is needed in case no pattern can be chosen for an order. The home plant of trucks of class c is denoted by p_c. Travel times from a truck's home plant p_c to the construction site associated with order o (and vice versa) are denoted as $TT_{p_c,o}$ (TT_{o,p_c}). Travel times between two orders o_1 and o_2 (including the time necessary for going to the closest plant en route without being loaded there) are referred to as TT_{o_1,o_2}.

The set of orders is denoted by O. All patterns available in the pool of pattern for order $o \in O$ are denoted by A_o. C refers to the set of all classes of trucks. The time horizon is

denoted by T.

$$\begin{aligned} Z = & \sum_{\substack{o\in O\\ c\in C\\ t\in T}} moveP2O_o^{c,t} \cdot TT_{p_c,o} + \sum_{\substack{o\in O\\ c\in C\\ t\in T}} moveO2P_o^{c,t} \cdot TT_{o,p_c} + \\ & \sum_{\substack{o_1,o_2\in O\\ c\in C\\ t\in T}} moveO2O_{o_1,o_2}^{c,t} \cdot TT_{o_1,o_2} + \\ & \beta_1 \sum_{\substack{o\in O\\ a\in A_o}} choose_{o,a} \cdot delay_{o,a} + \beta_2 \sum_{o\in O}(1 - \sum_{a\in A_o} choose_{o,a}) \end{aligned} \tag{5.1}$$

For every order at most one fulfillment pattern can be chosen. The choice itself is modeled using a binary decision variable $choose_{o,a}$ which will be equal to one if pattern $a \in A_o$ is chosen for order o. If no pattern can be chosen for any given order an additional penalty term accrues.

$$\sum_{a\in A_o} choose_{o,a} \leq 1 \quad \forall o \in O \tag{5.2}$$

In case a certain fulfillment pattern $a \in A_o$ is chosen for order $o \in O$, one has to ensure that all single deliveries associated with it, are actually going to be executed. All trucks scheduled need to arrive just in time. An early arrival is not possible. Trucks will start their unloading operations immediately after arriving at the construction site associated with order o. $TTL_{p_c,o}^c$ denotes the time necessary for loading at the home plant of truck class c and driving to the construction site associated with order o. The time necessary for going from any order o_1 to o, while loading at the closest plant en route, is denoted by $TTL_{o_1,o}^c$.

$$\sum_{a\in A_o} choose_{o,a} \cdot P_{o,a}^{c,t} = moveP2O_o^{c,t-TTL_{p_c,o}^c} + \sum_{o_1\in O} moveO2O_{o_1,o}^{c,t-TTL_{o_1,o}^c} \quad \forall o \in O, c \in C, t \in T \tag{5.3}$$

Constraint 5.3 ensures that trucks scheduled within pattern a for order o arrive in time, either coming from their home plant or any order where they might have been before, if the fulfillment pattern a is chosen.

Trucks can leave a construction site either after having finished their unloading operation or - in the very special case of the first truck scheduled to bring along special equipment

- after the last truck associated with a pattern has finished its unloading operation. The time necessary for fully unloading a truck of class c at the construction site associated with order o is denoted by U_o^c. It is given by the truck's capacity and the order's unloading rate.

Trucks do not have to leave immediately after being able to leave. The number of trucks of class c not in use and waiting at order o in t is denoted by $wait_o^{c,t}$. Due to the fact that the choice of plants for loading operations, the loading operation at plants themselves and the movements there are not modeled explicitly, idle trucks are modeled in sense that they wait at a construction site after having finished there scheduled task awaiting their next loading and unloading request.[1]

Constraints 5.4 are valid for all orders $o \in O \setminus O'$ not requiring any type of special unloading equipment. Those are balance equations for every node ensuring that a truck of class c can only leave a construction site associated with order o in case it already finished its unloading operation or has been waiting there before. The first term on the left hand side catches trucks just finishing their unloading operation in t, the second one refers to trucks that have been waiting there idle since the last time period. Trucks could either remain idle at the construction site, move to any order or move back home to their depot.

$$\sum_{a \in A_o} choose_{o,a} \cdot P_{o,a}^{c,t-U_o^c} + wait_o^{c,t-1} = \sum_{o_1 \in O} moveO2O_{o,o_1}^{c,t} + moveO2P_o^{c,t} + wait_o^{c,t} \quad \forall o \in O \setminus O', c \in C, t \in T \tag{5.4}$$

Similarly, Constraints 5.5 have to hold for all orders $o \in O'$ requiring special unloading equipment. The first truck to arrive needs to be equipped accordingly, but this will already be taken care about when generating patterns for the corresponding order. Trucks again are allowed to leave after finishing their unloading operation, or later in case they spend some idle time at the construction site associated with order o. The first truck to arrive however needs to stay at the construction site and only is allowed to leave after the last truck associated with the pattern chosen has finished its unloading operation. We need to make sure that these vehicles stay at the construction site. The start of the first unloading operation associated with pattern $a \in A_o$ is denoted by $start_{o,a}$. The end of the last unloading operations is denoted by $end_{o,a}$ respectively. This leads to the following

[1] In reality however and because of space limitations vehicles are not likely to spend their idle times at construction sites after having finished their unloading operations. Without loss of generality, for modeling purposes and in order to keep the formulation small any waiting time happens at the location from which the vehicle departs. In case a vehicle is about to execute its first delivery on a particular day this would correspond to its home plant. Otherwise it would wait at the construction site it just finished its previous unloading operation before.

set of equations. The first term in the left hand side refers to trucks which have been scheduled as first one for an order requiring special instrumentation. They are allowed to leave (at earliest) when the last truck of the associated pattern has finished its unloading operation. Please note that we need to exclude $t = start_{o,a} + UL_{o,c}$ and $c = first_{o,a}$ unless $t \neq end_{o,a}$ for the special case where a pattern only consists of one single unloading operation. In this case the truck will be allowed to leave immediately after having finished its own unloading operation.

$$\begin{aligned} &\sum_{\substack{a \in A_o: \\ \neg\{c=first_{o,a} \wedge \\ t=start_{o,a}+U_o^c \wedge \\ t \neq end_{o,a}\}}} choose_{o,a} \cdot P_{o,a}^{c,t-U_o^c} + \sum_{\substack{a \in A_o: \\ \{t=end_{o,a} \wedge \\ c=first_{o,a}\}}} choose_{o,a} \cdot P_{o,a}^{c,start_{o,a}} + wait_o^{c,t-1} = \\ &\sum_{o_1 \in O} moveO2O_{o,o_1}^{c,t} + moveO2P_o^{c,t} + wait_o^{c,t} \qquad \forall o \in O', c \in C, t \in T \end{aligned} \tag{5.5}$$

One has to ensure that trucks start their daily tours from their home plants and return there after having executed their last unloading operation. Hence the following boundary conditions need to hold: the number of trucks of a given class c available at a given plant p (n^{p_c}) equals the number of trucks originating their tour their plus the number of trucks staying there throughout the day. All trucks need to return home analogously.

$$n^{p_c} = stayHome_c + \sum_{\substack{o \in O \\ t \in T}} moveP2O_o^{c,t} \quad \forall p \in P, c \in C \tag{5.6}$$

$$n^{p_c} = stayHome_c + \sum_{\substack{o \in O \\ t \in T}} moveO2P_o^{c,t} \quad \forall p \in P, c \in C \tag{5.7}$$

Unfortunately the resulting model has no longer a pure MCNF structure, as certain binary conditions (one pattern has to be chosen for every order, see Equation 5.2) disrupt the network structure. Nevertheless the complexity of the model and run-times can be reduced dramatically, compared to holistic approaches.

5.3. Extensive Mathematical Formulation

As soon as one might face additional capacity restrictions, where at most one truck can be loaded at a plant at any given point in time, the model mentioned before is no longer suitable. Loading operations were supposed to take place at the trucks' home plant (when initiating the daily tour) or on the closest plant en route between two construction sites visited one after each other. The choice of plant was not left over to the model, rather

the closest one was always chosen. Neither the movements of all trucks were tracked in a sense to find out when two of them are going to be loading simultaneously.

In case capacity restrictions at plant exists that limit the number of trucks that can be loaded simultaneously, some adjustments need to be made with respect to the previous formulation in order such that one additional feature can be incorporated: the choice of plant where unloading operations are supposed to take place, while making sure that no simultaneous unloading operations are going to take place.

Again the model is formulated as an integer multi-commodity flow problem on a time-space network. The aim is again to find a cost-effective schedule which minimizes travel times, while - given a set of possible patterns for every order - choosing possible good and matching patterns, such that time gaps between consecutive unloading operations at the same order can be avoided. Additionally the capacity restriction at plants needs to be taken into account.

The decision variables $moveO2P_{o,p}^{c,t}$ and $moveP2O_{p,o}^{c,t}$ are used to model truck movements between plant p and the construction site corresponding to order o starting at t and executed by any truck of class c, respectively. The start of any loading operation is modeled by use of a binary decision variable $load_p^{c,t}$, which will be equal to 1 if a truck of class c starts being loading at plant p at time t. Waiting (i.e. idle) time of trucks at plants still being unloaded (or already loaded) is modeled in terms of $waitBL_p^{c,t}$ ($waitAL_p^{c,t}$) respectively, indicating the number of trucks of a given class c which are waiting unloaded (loaded) at plant p in t.

To ensure the validity of the underlying network flow structure, we need to make sure that trucks somehow move through the given network. For a truck two options exist how to enter a plant node: the first possibility compromises the fact that the truck would have already been there in the previous time period and remains there idle and waiting before being loaded at the given plant. Alternatively the truck could just arrive there (empty) after having delivered concrete to a construction site. When leaving a plant node there exist two possibilities as well: either the truck stays (empty) at the plant or is about to start its loading operation. A graphical representation of plant nodes and the options considered is depicted in Figure 5.4.

Generally speaking the flow balance equation for the plant nodes looks like this, where $TT_{o,p}$ denotes the travel time necessary for getting from the construction site associated with order o to plant p. Flow conservation constraints for empty trucks at plants are

modeled by means of Constraints 5.8.

$$waitBL_p^{c,t-1} + \sum_{\substack{o \in O \\ t \geq TT_{o,p}}} moveO2P_{o,p}^{c,t-TT_{o,p}} = waitBL_p^{c,t} + load_p^{c,t} \qquad \forall p \in P, c \in C, t \in T, \text{where } t > 1 \tag{5.8}$$

A graphical representation of the embedded time spaced network from the plants' point of view can be found in Figure 5.4.

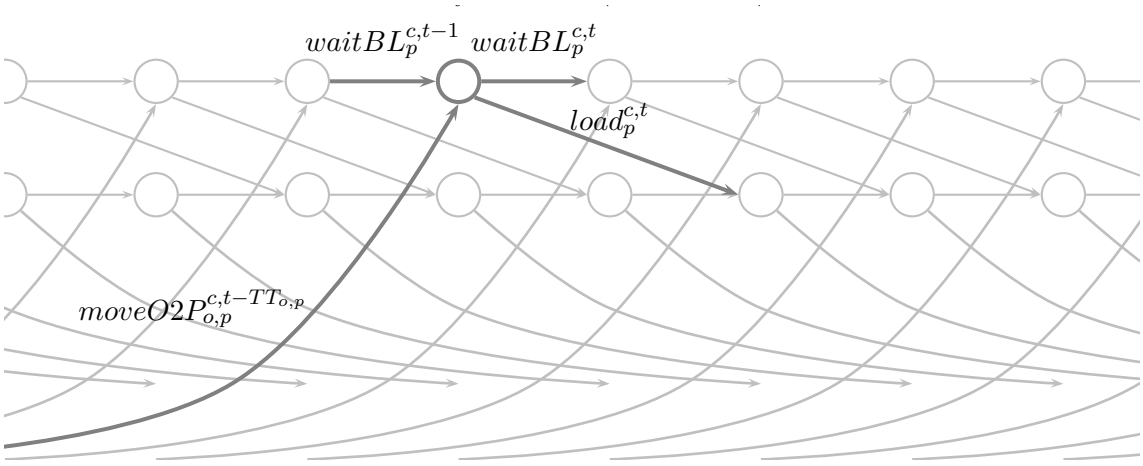

Figure 5.4.: Network Flow alike Presentation of Plant Nodes

As boundary conditions one has to consider that a truck can only start from its home location.

$$waitBL_p^{c,1} + load_p^{c,1} = n^{p_c} \qquad \forall p \in P, c \in C \tag{5.9}$$

Similarly within every layer of the network one can identify time-indexed nodes after loading operations (referred to as load nodes). Again, as this is a network flow formulation trucks somehow need to enter and leave the load nodes (see second row of nodes in Figure 5.5). There exist only two possibilities how a truck can enter a load node: either the truck started its loading operation at the corresponding plant previously and just finished with it, or it is already fully loaded and has been waiting there since the previous time period. When leaving the load node again two possibilities exist: either the truck remains (fully loaded) at the plant or alternatively drives to the construction site corresponding to some order. The flow balance equation for load nodes is formulated in Equation 5.10,

where L_p^c denotes the time necessary in order to fully load a vehicle of class c and plant p.

$$waitAL_p^{c,t-1} + load_p^{c,t-L_p^c} = waitAL_p^{c,t} + \sum_{o \in O} moveP2O_{p,o}^{c,t} \qquad \forall p \in P, c \in C, t \in T, t > 1 \tag{5.10}$$

.

Figure 5.5 depicts the situation graphically.

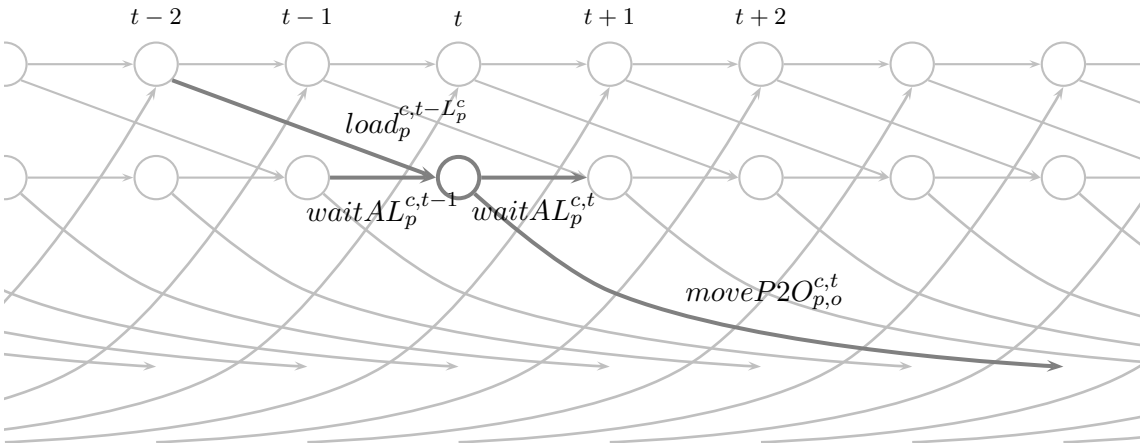

Figure 5.5.: Network Flow alike Presentation of Load Nodes

As the capacity for loading trucks at all plants is restricted such that at most one truck can be loaded at once, the following Restrictions 5.11 and 5.12 have to be imposed.

$$\sum_{c_2 \in C} \sum_{\substack{t_2 \in T \\ t < t_2 < t + L_p^c}} load_p^{c_2,t_2} = (1 - load_p^{c,t}) \cdot M \qquad \forall c \in C, t \in T, p \in P \tag{5.11}$$

$$\sum_{c \in C} load_p^{c,t} \leq 1 \qquad \forall p \in P, t \in T \tag{5.12}$$

Equation 5.11 states that as soon as any truck has started its loading operation at plant p in time t, no other truck is permitted to start its loading operation during the length of the loading process L_p^c. Equation 5.12 makes sure that at most one truck per plant can start its loading operation at any point in time.

For every order one alternative should be chosen. If no feasible assignment of a pattern

to an order can be made this will be penalized in the objective function accordingly.

$$\sum_{a \in A_o} choose_{o,a} \leq 1 \qquad \forall o \in O \tag{5.13}$$

If an alternative is going to be chosen (i.e. if $choose_{o,a} = 1$), all corresponding deliveries according to this pattern need to be realized. Trucks foreseen for a certain unloading operations (according to a pattern chosen) need to get to the construction site corresponding to that order just in time. And after having finished with their unloading operations (or finished with assisting during the unloading operation of other trucks) the trucks have to leave that site again. $TT_{p,o}$ denotes the travel time necessary for getting from plant p to the construction site associated with order o.

Equation 5.14 ensures that all trucks scheduled within a given pattern arrive at the construction site just in time if required by the chosen pattern.

$$\sum_{\substack{p \in P \\ t \geq TT_{p,o}}} moveP2O_{p,o}^{c,t-TT_{p,o}} = \sum_{a \in A_o} choose_{o,a} \cdot P_{o,a}^{c,t} \qquad \forall o \in O, c \in C, t \in T \tag{5.14}$$

Equation 5.15 and 5.16 respectively make sure that trucks also leave the construction site after having fulfilled their duties there. Usually trucks have to leave the construction site immediately after having finished their unloading operation. Just in case the order requires special instrumentation for its unloading procedures, the first truck scheduled and arriving at the construction site needs to be equipped with the corresponding instrumentation (the pattern already takes care about that). In this case the first truck arriving at the construction site needs to stay and assist later arriving trucks with their unloading operations. The first truck bringing along special instrumentation is only allowed to leave the construction site when the *last* truck scheduled within the chosen pattern has finished its unloading operation. Equation 5.15 handles the situation for all orders $o \in O \setminus O'$ not requiring any special type of unloading equipment. All trucks are allowed to leave the construction site immediately after having finished their unloading operation. The latter however deals with orders $o \in O'$ requiring unloading equipment. The first truck to arrive is not allowed to leave the orders construction site immediately.[2]

$$\sum_{a \in A_o} choose_{o,a} \cdot P_{o,a}^{c,t-U_o^c} = \sum_{p \in P} moveO2P_{o,p}^{c,t} \qquad \forall o \in O \setminus O', c \in C, t \in T \tag{5.15}$$

[2]Please note that in case a pattern for any order $o \in O'$ only consists of one single delivery the associated truck will leave immediately after having finished its unloading operation. There is no need to stay any longer.

$$\sum_{\substack{a\in A_o:\\ \neg\{c=first_{o,a}\wedge\\ t=start_{o,a}+U_o^c\wedge\\ t\neq end_{o,a}\}}} choose_{o,a}\cdot P_{o,a}^{c,t-U_o^c} + \sum_{\substack{a\in A_o:\\ \{c=first_{o,a}\wedge\\ t=end_{o,a}\}}} choose_{o,a}\cdot P_{o,a}^{c,start_{o,a}} =$$
$$\sum_{p\in P} moveO2P_{o,p}^{c,t} \qquad \forall o\in O', c\in C, t\in T, \text{where } t\geq U_o^c \tag{5.16}$$

A fragment of the embedded network flow structure corresponding to the two constraints explained before is shown below. Figure 5.6 shows that if a certain pattern is chosen, the designated trucks somewhere need to come from. The already have been loaded at any of the plants. Figure 5.7 demonstrates how trucks can leave a construction site after having finished their unloading operation.

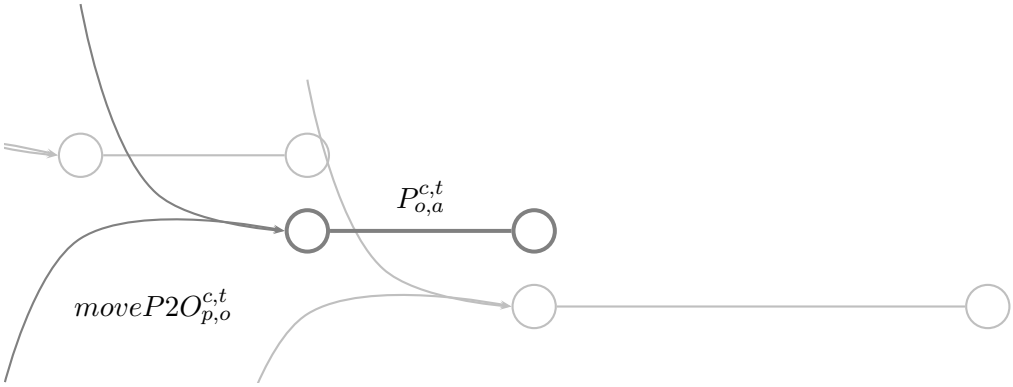

Figure 5.6.: Trucks getting to Order to execute Delivery

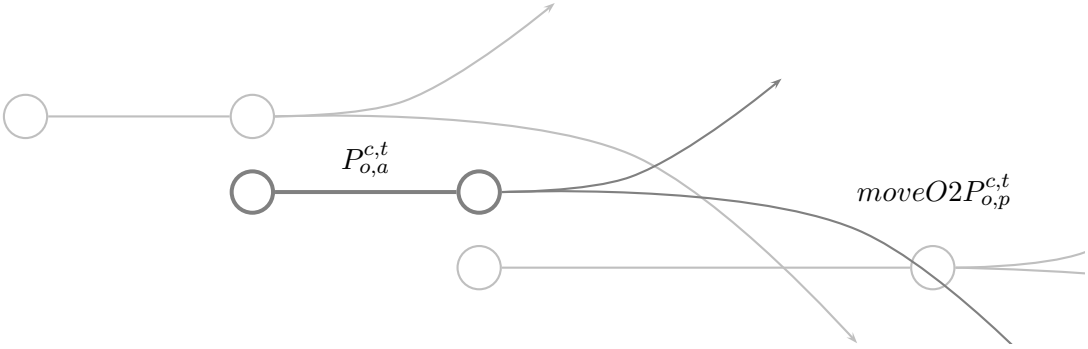

Figure 5.7.: Trucks leaving Order after executing Delivery

Finally we need to make sure that all trucks return to their home plant by the end of the planning horizon.

$$waitBL_p^{c,|T|-1} + \sum_{o \in O} moveO2P_{o,p}^{c,|T|-TT_{o,p}} = n^{p_c} \qquad \forall p \in P, c \in C \tag{5.17}$$

6. Variable Neighborhood Search

Exact algorithms are highly useful when it comes to solving combinatorial optimization methods to optimality. In practice however small sized problem instances only may be solved to optimality. Alternatively it might be possible to solve some real-world sized instances to optimality, but nevertheless it is still not meaningful as the resulting run time would end up being too high. Hence different types of heuristics or even more sophisticated metaheuristics emerged which are able to quickly solve problems and find good quality solutions (see Glover and Kochenberger, 2003). Due to the steady increase in computational power, sophisticated search procedures such as Variable Neighborhood Search (VNS) allow to search and intensify the search in the solution space. VNS is a highly promising metaheuristic which has been developed by Mladenović and Hansen (1997) and extended in Hansen and Mladenović (2001). It is a Local Search (LS) based improvement heuristic. In contrast to population based approaches it 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 current incumbent solution is perturbed by means of different neighborhood structures, allowing the solution process to explore various regions of the solution space and (hopefully) to escape any local optima. The following LS steps intensify the search and are going to improve it locally.

The previous approaches based on MCNF (see Chapter 5) restricted their view to a limited number of fulfillment patters given and let the MCNF choose one per order. The VNS component however is able to diversify the search without being restricted to certain patterns. During the solution process also unexplored regions of the solution space can be visited within a certain neighborhood structure.

This chapter is organized as follows. First we will present the basic implementation for the VNS. Afterwards design issues such as the shaking phase, the embedded LS operators, the evaluation and acceptance scheme in the context of RMC delivery are going to be explained. All trucks are handled individually and are no longer aggregated into classes. This facilitates the evaluation process and feasibility check for any solution found. The obtained results are depicted in Section 8.9. For a more detailed overview see Section B.5.

6.1. Basic Implementation

Any solution x found can further be improved by VNS. During the *shaking phase* several neighborhoods are used to explore the solution space more thoroughly and avoid potentially being trapped in local optima. Sequences of trucks designated to be sent to an order will be inverted and exchanged. The embedded LS locally optimizes every solution obtained after the shaking step. In the literature the LS phase is also referred to as iterative improvement phase.

A neighboring solution x' will be generated at random from neighborhood $N_\kappa(x)$. Starting from x' the current solution is going to be locally optimized by means of LS. The resulting solution is denoted by x''. If this solution improves the current solution x, the best incumbent solution x'' will become the new current solution x and shaking continues with the very first neighborhood. If no new best 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 though could easily be adapted to incorporate this feature as well.

A sketch of the basic steps of the implemented VNS can be found in Algorithm 6.1.

Algorithm 6.1 Basic Steps of VNS

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

6.2. Design Issues

6.2.1. Shaking Phase

In order to explore the solution space - in terms of potential patterns - more thoroughly, two shaking operators, resulting in six neighborhood structures have been implemented.

Table 6.1.: Set of Neighborhood Structures (standalone VNS)

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

The first shaking operator tries to replace sequences of trucks within a pattern by trucks *not used* so far in any other pattern 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 pretty similar, but this time the new trucks to be inserted instead 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 up to $(\kappa + 1)/2$ patterns; neighborhood structures N_κ ($\kappa = 2, 4, 6$) relate to the second shaking operator and involve changes of up to $\kappa/2$ patterns. An overview on the set of neighborhood structures is given in Table 6.1. The position and the length of the sequence to be exchanged are determined randomly. The loss of unloading capacity associated with the vehicles being removed needs to be compensated. Hence new vehicles will be inserted into the pattern until the demand of the corresponding order can be satisfied again. Furthermore, if the first vehicle of the sequence is going to be exchanged and the order requires special unloading equipment, only vehicles equipped accordingly will be considered when selecting the replacement for the first vehicle of the pattern.

Any neighborhood operator should perturb the current incumbent solution x which might lead to a new best incumbent solution x'' after applying LS. The main idea of any shaking operator is to *replace* sequences of trucks within patterns belonging to up to $(\kappa + 1)/2$ (or $\kappa/2$ respectively) orders by a different sequence of trucks. The position and the length of the sequence to be exchanged are determined randomly. In order to compensate for the capacity of all trucks that are about to be dismissed, new trucks will be consecutively be inserted into the remaining schedule, until the total demand of the corresponding order will be satisfied again.

First, the start (the position of the first truck), as well as the length of the sequence

to be exchanged will be determined randomly. The total length (i.e. the total number of trucks) within pattern pat_o corresponding to order o is denoted by $l(pat_o)$. Let s' denote the position of the first truck within the pattern which is going to be replaced. l' stands for the number of trucks that will be replaced. Starting from position s' a total number of l' trucks will be replaced in any pattern. To compensate for the capacity of all trucks that are about to be dismissed, new trucks will be consecutively be inserted into the remaining schedule, until the total demand of the corresponding order (Q_o) will be met again.

Let $cumCap(pat_o)$ be the cumulative capacity of all trucks scheduled within the original pattern pat_o for order o so far. $cumCap(pat_o^-)$ denotes the cumulative capacity of trucks going to be replaced within pattern pat_o. The capacity of newly added trucks is denoted by $cumCap(pat_o^+)$.

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 going to be 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 pattern.

Shaking operators only have an effect upon the sequence of trucks to be scheduled to satisfy an order. The exact timing of all unloading operations as well as feasibility from trucks point of view are ignored. After having executed any shaking operator, the resulting patterns are supposed to be feasible from any single orders point of view. However, they do not constitute a feasible solution from the trucks' point of view, as 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 effects of any shaking operator, an evaluation function has been implemented, which is going to be described below. 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 a plant and being loaded there.

The first shaking operator `ReplaceByUnused` tries to replace sequences of trucks in patterns corresponding to up to $(\kappa + 1)/2$ orders. Trucks are going to be replaced by other trucks not in use far within the current solution. When selecting new trucks for a pattern, there is a bias towards choosing large trucks whose home plant is located close

to the construction site. The selection probability for a plant to be chosen is indirectly proportional to the travel time from the construction site to the corresponding plant. Additionally there is a bias towards choosing larger trucks, which might help to keep the number of trucks needed (and resulting travel times) low. The resulting pattern needs to remain feasible from the orders point of view (i.e. demand needs to be satisfied; requirements concerning unloading equipment have to be considered). The procedure is described in Algorithm 6.2.

Algorithm 6.2 Shaking Operator 1: `ReplaceByUnused`(κ)

$o_{max} \leftarrow rand(1, min(numberOfOrders, \frac{\kappa+1}{2}))$ ▷ number of orders to perturb
for any o_{max} orders $o \in O$ **do**
  $s' \leftarrow rand(0, l(pat_o) - 1)$ ▷ starting position of sequence to be exchanged
  $l' \leftarrow rand(1, l(pat_o) - s' - 1)$ ▷ length of sequence to be exchanged
  $cumCap(pat_o^-) \leftarrow$ capacities of trucks formerly scheduled at position $s' \ldots s' + l'$
  $cumCap(pat_o^+) \leftarrow 0$
  remove trucks scheduled at positions $s' \ldots s' + l'$
  while $cumCap(pat_o) - cumCap(pat_o^-) + cumCap(pat_o^+) < demand_o$ **do**
    $p \leftarrow selectPlant(o)$ ▷ select plant: bias towards closer ones
    $k \leftarrow selectUnusedTruck(o, p)$ ▷ select truck: bias towards larger ones
    $cumCap(pat_o^+) \leftarrow cumCap(pat_o^+) + Cap_k$
  end while
end for

The second shaking operator `ReplaceByAny`, unlike the first one, tries to replace sequences of trucks by 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 will be biased towards trucks based in preferably close home plants (with respect to travel times) to the corresponding construction site and trucks with larger capacity. Again, the resulting sequences of trucks need to remain feasible from the orders point of view. In case special unloading equipment is required by the construction site, the first truck scheduled needs to be equipped with the corresponding instrumentation. The first delivery may not begin before the start of the associated time window and the total quantity demanded has to be met. To avoid contradictory situations, the truck bringing along special instrumentation is not supposed to show up again within the same schedule or pattern.

6.2.2. Evaluation

LS and the shaking operators embedded in VNS locally disturb any given solution x. A solution consists of one pattern per order. After any disturbance the obtained solution needs to be reevaluated. This is done through heuristically determining the start of any single unloading operation. The resulting patterns have 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 for the first unloading operation per order is determined by the start of the associated time window. The result obtained then serves as input for backward termination, where every unloading operation is tried to be 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. Resulting patterns 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.

An algorithmic description of the implemented forward termination is depicted in Algorithm 6.3. 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.

Algorithm 6.3 Forward Termination

```
1: Initialize ordered list L with first unloading operations of orders
2: while L ≠ ∅ do
3:     l ← first(L)
4:     L ← L − l
5:     o ← l.order                                   ▷ order under consideration
6:     k ← l.vehicle                  ▷ vehicle to perform unloading operation
7:     t ← l.time                          ▷ start time of unloading operation
8:     if order o is not blocked then
9:         if vehicle k can reach construction site of o by time t then
10:            schedule unloading operation with vehicle k at time t for order o
11:            if there are remaining unloading operations for order o then
12:                l.order ← o
13:                l.vehicle ← vehicle to perform next unloading operation
14:                l.time ← t + U_o^k
15:                L ← L + l                  ▷ insert unloading operation in sorted list
16:            end if
17:            if first unloading operation of order requiring special equipment then
18:                for o′ ∈ O′ requiring truck k and not yet started do
19:                    block o′ because of o
20:                end for
21:            end if
22:            if last unloading operation of order o requiring special equipment then
23:                for orders o′ that have been blocked by order o do
24:                    unblock o′
25:                    if order o′ no longer blocked by any order then
26:                        l.order ← o′
27:                        l.vehicle ← vehicle to perform first unloading operation
28:                        l.time ← start time of first unloading operation
29:                        L ← L + l              ▷ insert unloading operation in sorted list
30:                    end if
31:                end for
32:            end if
33:        else
34:            l.order ← o
35:            l.vehicle ← k
36:            l.t ← earliest time vehicle k can reach construction site of order o
37:            L ← L + l                      ▷ insert unloading operation in sorted list
38:        end if
39:    end if
40: end while
```

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.

This relation can be illustrated by the following example: Imagine for instance the situation (see Figure 6.1) where we are about to schedule the timing for deliveries associated with two orders o_1 and o_2, both of which require special unloading equipment. In order to satisfy the demand two deliveries are necessary. Truck A (B) is supposed to execute the first (second) delivery of order o_1. For order o_2 the sequence of trucks is just vice versa. It is not known beforehand how long the trucks scheduled for the first delivery need to stay, as the rest of the pattern has not been scheduled yet. Imagine further that the start of the first unloading operations has already been determined for both orders. Now we are trapped into a deadlock situation. Truck A, which has been scheduled for the first delivery of order o_1 cannot be released until the second delivery has been executed by truck B. Hence we have to avoid the situation where any truck A waits for B, which itself cannot leave before A has executed its unloading operation respectively.

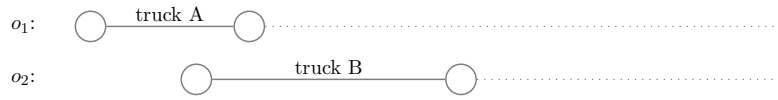

Figure 6.1.: Deadlock situation without blocking orders

Therefore we cannot start determining the timing for the first unloading operation of order o' (requiring special instrumentation) executed by truck k if there is another order o (where $o' \neq o$), also requiring special instrumentation, where the beginning of some unloading operations already has been scheduled, but the entire pattern is not yet finished and truck k still is supposed to be used for order o as well. The blocking will be released as soon as we finished scheduling the pattern for order o. Please note that any order may be blocked by more than one order. Scheduling the first unloading operation can only be started if it has not been blocked by any other order.

We finally schedule an unloading operation if order o is not blocked and truck k is able to get there in time t (taking into account where it as been scheduled before, the time it needs to get to order o, being loaded, etc). Usually it is known after having scheduled a

truck how long it lasts in order to execute its unloading operation, how long it needs to stay and how long it would take to get to order o. However, for trucks executing the first unloading operation of an order requiring special instrumentation, where the scheduling of the pattern has not been completed yet this information is not yet available. Therefore such trucks are temporarily unavailable and cannot be scheduled anywhere else in the meantime. This information will become available as soon as the last unloading operation for the corresponding order has been scheduled.

If the delivery for which the timing just was determined has not been the last delivery scheduled within the pattern of order o a new entry will be added into the event list L. The next unloading operation could start after the current one has been finished. In case a pattern for an order which requires special unloading equipment just has been scheduled entirely, orders that might have been blocked by order o can be unblocked. Again, it is important to note that an order may be blocked by several orders. Unblocking it once does not necessarily mean they are eligible to start scheduling their first unloading operation immediately.

In case the timing of an order cannot be determined yet because the truck is not available, a new entry will be inserted into the event list L stating the next possible time where truck k could be there.

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.

Backward termination works pretty much the same, just vice versa (see Algorithm 6.4). 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.

6.2.3. Local Search

For the embedded LS (see Hoos and Stützle, 2004) three different operators have been implemented in order to explore the solution space further and enrich the current pool of fulfillment patterns by new ones. One of the three LS operators (`Shrink`) has been introduced by ourselves. 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 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 pattern of a given order and inserts it again at any other possible position. The second operator, `InterPatternSwap`, exchanges two deliveries associated with patterns of two different orders. The last operator `Shrink` tries to remove unnecessary trucks from any pattern as the quantity ordered might be satisfied with all remaining deliveries.

In order to execute any of the three operations implemented within the process of LS the timing of all deliveries will be dismissed. The actual move, swap or deletion takes place, taking only into account the sequence of trucks from the pattern point of view. Afterwards the timing for the start of all individual unloading operations will be determined again 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.

Algorithm 6.4 Backward Termination

```
1: Initialize inversely ordered list L with end of last unloading operations of orders
2: while L ≠ ∅ do
3:     l ← first(L)
4:     L ← L − l
5:     o ← l.order                                   ▷ order under consideration
6:     k ← l.vehicle                  ▷ vehicle to perform unloading operation
7:     t ← l.time                                    ▷ end time of unloading operation
8:     if order o is not blocked then
9:         if vehicle k can reach construction site of o by time t then
10:            schedule unloading operation with vehicle k at time t for order o
11:            if there are remaining unloading operations for order o then
12:                l.order ← o
13:                l.vehicle ← vehicle to perform previous unloading operation
14:                l.time ← t − U_o^k
15:                L ← L + l                  ▷ insert unloading operation in sorted list
16:            end if
17:            if last unloading operation of order o requiring special equipment then
18:                for o′ ∈ O′ requiring truck k and not yet started do
19:                    block o′ because of o
20:                end for
21:            end if
22:            if first unloading operation of order o requiring special equipment then
23:                for orders o′ that have been blocked by order o do
24:                    unblock o′
25:                    if order o′ no longer blocked by any order then
26:                        l.order ← o′
27:                        l.vehicle ← vehicle to perform last unloading operation
28:                        l.time ← end time of last unloading operation
29:                        L ← L + l          ▷ insert unloading operation in sorted list
30:                    end if
31:                end for
32:            end if
33:        else
34:            l.order ← o
35:            l.vehicle ← k
36:            l.t ← earliest time vehicle k can reach construction site of order o
37:            L ← L + l                      ▷ insert unloading operation in sorted list
38:        end if
39:    end if
40: end while
```

Algorithm 6.5 `LocalSearch`(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
```

A sketch of the solution procedure implied by LS can be found in Algorithm 6.5. Any given solution, based on one pattern per order (x), is tried to be optimized locally. The first operator to be executed is `Shrink`, which is supposed to eliminate unnecessary deliveries of single patterns within solution x. The two following operators `IntraPatternMove` and `InterPatternSwap` are executed on a first improvement basis. `InterPatternSwap` will be performed as soon as `IntraPatternMove` cannot improve the solution x any more. To complete the LS procedure one last iteration of `Shrink` will be executed again, just to make sure no unnecessary deliveries remain within the patterns. 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 pattern element of the current solution is going to be revised. Any single delivery scheduled is tried to be omitted. The resulting pattern needs to remain feasible. A delivery can only be omitted if the capacity of the remaining trucks within this pattern 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 pattern, 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 pattern. 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. Again the termination and timing of all deliveries associated with a single pattern will be dismissed and only the sequence of trucks counts. In order to evaluate any solution found this way, the timing will be determined using the approach described in Section 6.2.2. If an improvement could be found the resulting solution is accepted as the new current 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 patterns corresponding to two different orders. Any combination of two trucks originally scheduled for deliveries within a pattern associated with order o_1 and o_2 are tried to be exchanged. After any swap the timing of all deliveries to be performed is dismissed and new patterns (including timing) - based on changed sequences of deliveries - will be constructed accordingly using forward and backward termination. The operator itself is executed on a first improvement basis. Any swap that leads to a new best solution will replace the current incumbent solution and the procedure starts over again.

6.2.4. Acceptance Decision

After performing any search step within LS or shaking the newly obtained solution needs to be evaluated in order to be comparable with the current incumbent solution. There is no need to deal with infeasibilities, as infeasible solutions and/or pattern are not going to be generated at all. Any solution will only be accepted if it improves the current 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.

So far we do not allow any ascending moves, although they constitute one additional way to escape local optima, as we tried to keep the number of parameters as small as possible. Permitting deteriorating solutions indeed makes sense in case VNS is applied solely. In the remainder of this thesis however we will focus on a cooperative hybrid approach (see Chapter 7). By combining the strengths of VNS and MCNF we are able to obtain a global perspective. Any incumbent solutions found during the stage of VNS will be added to

the pool of original patterns therefore enriching the choice process of the MCNF. The MCNF focuses on the pure selection of patterns and the scheduling of all underlying truck movements. Any solution found by MCNF serves as an input for VNS where it is going to be locally optimized.

7. Cooperative Hybrid Approach using MCNF and VNS

Linear Programming (LP) and formulations based on Mixed Integer Programming (MIP) will find - in case they converge - an optimal solution. One of the disadvantages of this approach might be that it would take way too long in order to solve those problems to optimality. On the other hand so called heuristics - or even more sophisticated meta-heuristics - exist, which attempt to find good, but not necessarily optimal solutions, in a reasonable amount of time.

To solve large scale problem instances we decided you use a hybrid approach, combining the power of meta-heuristics with the strength of exact approaches enabling us to overcome the disadvantages of the two approaches if applied exclusively. Therefore a hybrid approach has been developed in order to overcome the problems mentioned and solving problems in a reasonable amount of time. The resulting framework is also discussed in Schmid et al. (2007d).

An overview of the results obtained compared to all other approaches described previously can be found in Section 8.10, 8.8 and 8.9. A comparison with the tool based on SA can be found in Section 8.11. For a detailed overview on results obtained using various run time limits the reader is referred to Section B.6.

The implemented MCNF (see Section 5.2) model selects the pattern such that all routes from the trucks point of view are feasible, in the sense that there will be enough time between unloading operations for driving to a plant, being loaded with concrete and getting to the next order in time.

A representation of the solution procedure applied can be found in Figure 7.1. As an initial step, base patterns are generated for every order. Base patterns are generated randomly using the brute force procedure described in Section 5.1.1. Alternatively a more intelligent approach as been developed, trying to generate good patterns where overlap with previously generated patterns is tried to be avoided (see Section 5.1.1). After solving the MCNF, a VNS locally improves the solution obtained and generates additional

patterns. The generation of these patterns is guided by the characteristics of the orders as well as by the patterns selected in the last solution to the MCNF formulation.

Within the embedded hybrid framework VNS and MCNF alternate in trying to quickly find a good and feasible solution. Any solution found by MCNF will be used as a starting solution for the VNS, whereas the patterns chosen by the MCNF serve as the initial solution for the VNS. In case - based on the current pool of patterns - a complete solution (i.e. a solution where one pattern for every order has been found) cannot be found by MCNF, missing pattern will quickly be generated in order to get a complete 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 will be added to the pool of patterns. Thereby the pool of patterns will be enriched, yielding more alternatives and flexibility when it comes to solving the MCNF.

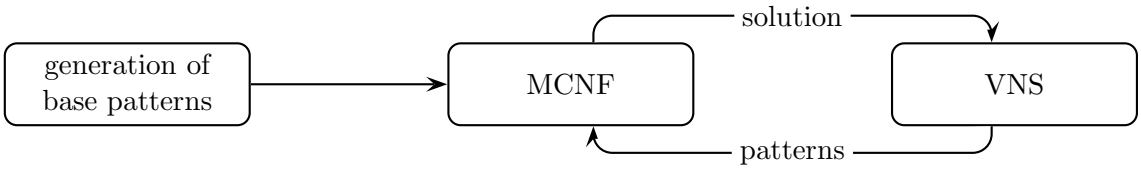

Figure 7.1.: Hybrid Solution Procedure

An overview on different means of communication between MCNF and VNS is given is Section 7.1.

7.1. Communication between MCNF and VNS

For the MCNF formulation each type of delivery truck was modeled as a separate commodity. Trucks with same capacity, home plant, and instrumentation are grouped into classes and considered as one single commodity. Analogously the generation of base patterns was also controlled by this aggregated view and all patterns were generated based on classes. Due to the aggregation of similar trucks into classes the size of the problem can be kept smaller. Furthermore additional flexibility can be gained through the use of class based patterns. For any pattern where a truck of class c is supposed to execute a delivery, any truck k within the class c is eligible to perform all associated actions.

All solutions used throughout the process of VNS and LS have a different view on patterns. To facilitate the evaluation of any given solution, all patterns are based on

individual trucks. These two views are perfectly interchangeable. Two transformation functions have been developed in order to transform any pattern from (individual) truck to class view (and vice versa). The transformation functions will be described below.

Please note, that when the MCNF is executed for the very first time based on the patterns initially generated, it is not guaranteed finding a solution where a pattern for every order can be selected. The obtained solution is still feasible, but a comparatively large penalty term in the objective function will penalize it accordingly. In this case, before communicating the solution (in terms of patterns) to VNS, additional patterns for orders where MCNF could not choose one, are generated. The generation of these missing patterns follows the principle for generating compatible base patterns described in Section 5.1.1. We try to generate patterns that preferably do not overlap with the ones part of the current solution.

7.1.1. Transformation: Class → Truck

In order to use patterns found by MCNF as an initial solution for VNS or LS, patterns need to be transformed into patterns based on individual trucks. During every transformation exactly one pattern per order (i.e. one complete solution) will be transformed at a time. In order to successfully transform a set of patterns, the combination of the set of patterns needs to be feasible from the trucks point of view.

As the patterns under consideration just have been retrieved from a MCNF solution, all individual truck movements can easily be derived by the values of the corresponding decision variables. The path of any individual truck can be followed through the underlying network flow structure. This knowledge helps to differentiate which truck is going to execute which delivery within every pattern under consideration and the transformation can be realized accordingly.

7.1.2. Transformation: Truck → Class

The other way round, in order to include patterns generated during the stage of VNS and LS into the pool of patterns used by MCNF, patterns need to be transformed from patterns based on individual trucks into patterns based on classes of trucks. This direction of transformation is much easier than the one described before. The transformation is straight forward, as every truck uniquely belongs to exactly one class of trucks. The exact timing of all scheduled deliveries remains unchanged.

8. Computational Experiments

8.1. Data Description

For our numerical results we use real data of a concrete company located in Italy for all orders placed between January and November 2006. 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. But they are also equipped with a pump or a conveyor belt respectively in order to assist during unloading operations. The fleet is also heterogenous 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 14 (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.

Figure 8.1 plots the location of their five plants as well as the location of all cities and towns had to be delivered to.

Instances: Real world data is used for solving the scheduling problems of a medium sized company operating in the concrete industry located in Alto Adige. All orders to be satisfied the next day are known the evening before. The schedule is calculated during the night. All orders and their characteristics are known beforehand. In the computational tests real world data of 2006 is used. One problem instance relates to one single day, taking into account all orders that had to be fulfilled that particular day. The available data material at hand refers to orders placed and executed between January and October 2006.

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

Figure 8.1.: Area in Alto Adige

will not be outsourced to other logistic providers.

Trucks: Movements of trucks can take place between any plant and construction site associated with any order. Generally trucks are going to be loaded at a plant's site, drive to the construction site associated with the corresponding order in order to unload. The next plant visited not necessarily needs to be the same one where the truck has been loaded right before. The only requirement for trucks is that they are located at specific

plants (their home plants).

Trucks have to initiate their daily tour at their home plant. After having executed their last unloading operation along their tour, every single truck is supposed to return back to its home plant. Trucks do not have to be in use every single day. They also might remain at their home plant and do not fulfill any single loading or unloading operation.

The fleet of truck is heterogeneous. On one hand their capacity varies, on the other hand they differ in their instrumentation. Every truck is based at a particular plant called its home plant. All trucks initiate their daily tour from their home plant and need to return there by the end of the day. Some trucks can only be used for the delivery of concrete only. Others 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.

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

Trucks start their daily tours at their home plants respectively. Their first loading operation will be performed at their home plant. Vehicles need to return there by the end of the day. A vehicle's loading operations are not restricted to its home plant. Throughout the day they can be loaded at any other plant as well.

A plant's capacity is defined by the number of trucks that can be loaded simultaneously. Resource restrictions in terms of availability of required raw materials are not considered within these modeling approaches.

Orders & Construction Sites: Typically orders are placed the day before. 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 an order.

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 operations, preferably the next truck should already be available and ready to start its unloading operation. Any gaps between consecutive

unloading operations will constitute a problem 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.

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 (s_o) is not allowed and does not result in a feasible solution. 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.

Usually trucks unload the concrete loaded directly (and on their own) into the construction site. Some of the trucks are equipped with special instrumentation which might be needed during the process of unloading. Types of special instrumentation include pumps and conveyor belts, which facilitate the unloading process. The type of unloading equipment needed will be disclosed at the same time the order is placed. If any kind of special instrumentation (pump or conveyor belt) is needed, the first truck to arrive at the corresponding construction site needs to be equipped accordingly. These trucks first unload their own load of concrete. 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 satisfied and the last truck scheduled for a delivery has finished its unloading operation. It is not possible to replace the truck assisting others with their unloading operation. Any displace would be impractical, as it would take too much time, and hence disturb the unloading process.

Only one truck may be unloaded at a construction site at any point in time. The time needed for loading and unloading trucks is implicitly given by the plants' loading and construction sites' unloading rate respectively. Even if the capacity of a truck would permit serving several orders, trucks may only serve one order at a time. Hence small orders imply that trucks will only be partially loaded.

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), to small, medium-sized and larger ones. An overview on the particularities of all instances and the aggregated classes is shown in Table 8.1.

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 are consisting of 13 to 18 orders respectively 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 8.1.: 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 results obtained. If not stated otherwise, all calculations have been executed on desktop PCs (3.2 GHz, 3 GB RAM) and XPRESS-MP (v. 2006B) was used for solving all problem instances. 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 different sets of initial fulfillment patterns in use and random choices within local search and shaking. Aggregated values presented per categories (mini/small/medium/large) correspond to average over the values obtained for all individual instances within that category.

In the following we describe some of the results for the variants we have designed and studied. Starting with lower bounds and the attempt "to MIP" the problem based on a VRP* formulation, results obtained by applying the integrative hybrid approach will be presented. Afterwards results obtained using the MCNF (using patterns generated brute force and compatible ones), the pure VNS and the cooperative hybridization will be presented. Finally we compare ourselves to a software tool available in Austria, whose embedded algorithm is based on SA.

8.2. Solving the VRP* formulation

The minimum and maximum number of deliveries necessary to satisfy all orders within any given instance are denoted in Table 8.2. At least (at most) D_{min} (D_{max}) deliveries are needed.

Table 8.2.: Minimum and Maximum Number of Deliveries per Instance

mini			small			medium			large		
n	D_{min}	D_{max}	n	D_{min}	D_{max}	n	D_{min}	D_{max}	n	D_{min}	D_{max}
1	17	32	6	58	117	11	89	163	16	101	190
2	18	37	7	40	78	12	76	135	17	91	164
3	32	65	8	53	96	13	70	137	18	96	187
4	30	62	9	62	118	14	95	183	19	110	213
5	28	57	10	65	123	15	85	169	20	102	201
avg	25	50.6		55.6	106.4		83	157.4		100	191

In order to keep the number of decision variables and constraints in use small only meaningful decision variables (i.e. decision variables that also might show up in an optimal solution) are going to be generated. The meaningless decision variables going to be excluded using problem specific knowledge are:

- The first delivery of an order requiring special equipment may only be executed by a truck which is equipped accordingly. All decision variables associated with trucks not being equipped accordingly will always be fixed to zero in any feasible solution anyway. Therefore for any $o \in O'$ decision variables like $y^k_{o,1}$, where $tinstr_k \neq oinstr_o$, will not be added to the model at all. Similarly all resulting decision variables associated with the first and last movement of a truck per day will not be generated respectively. If in a feasible solution a certain truck k will not be allowed to execute a certain delivery, it will not go there either. Consequently $xp2o^k_{o,1}$ and $xo2p^k_{o,1}$, where $tinstr_k \neq oinstr_o$, are not generated respectively. The same holds for the decision variables related to the movements of trucks between deliveries. $xo2o^{o,1,k}_{o_1,d_1}$ and $xo2o^{o_1,d_1,k}_{o,1}$ (for any $o_1 \in O, d_1 \in D_{o_1}$) will not be added to the model either.

- Similarly in any optimal solution a truck with no capacity, which is only equipped with some kind of special instrumentation, should only be scheduled for executing the very first delivery of an order requiring the corresponding equipment respectively. Scheduling these particular trucks for any other kind of delivery makes no sense and will never be required by an optimal solution.

The same is true for all types of constraints accordingly. If a truck is not supposed to execute a certain delivery all corresponding degree equations are not necessary either. Although general purpose solvers such as XPRESS-MP or CPLEX possess the capability to detect such unnecessary constraints and decision variables during their pre-processing stage known as presolve, we decided to exclude them a priori, hence trying to keep the size of the resulting model or matrix small and not running into out of memory problems at small instances.

Table 8.3 indicates that number of decision variables (cols) and constraints (rows) in use, when *all* or only *meaningful* ones are generated. By creating only meaningful ones the number of decision variables and constraints on average can be reduced by 27%. During the process of presolve which is embedded in XPRESS the number of decision variables and constraints can further be reduced by a minor percentage. This mainly refers to decision variables such as $z_{o,d}$, where $d \leq d^o_{min}$ (indicating that a minimum number of deliveries is necessary by all means). We could have excluded them as well. As the percentage of variables and space that could be saved is negligible, we decided to leave them in the model.

We tried to solve several instances on another computer with even more memory (3.2

Table 8.3.: Number of decision variables and constraints per instance (VRP*)

			only meaningful ones				% saved			
	all		before presolve		after presolve		by reduce		by presolve	
n	cols	rows	cols	rows	cols	rows	cols	rows	cols	rows
1	37,071	38,147	27,837	28,828	27,809	28,774	−24.91	−24.43	−0.10	−0.19
2	48,967	50,202	32,173	33,221	32,154	33,177	−34.30	−33.83	−0.06	−0.13
3	146,074	148,193	105,180	107,112	105,147	107,035	−28.00	−27.72	−0.03	−0.07
4	133,196	135,218	96,053	97,854	96,022	97,784	−27.89	−27.63	−0.03	−0.07
5	113,052	114,919	78,814	80,443	78,785	80,375	−30.29	−30.00	−0.04	−0.08
mini	95,672	97,336	68,011	69,492	67,983	69,429	−29.07	−28.72	−0.05	−0.11
6	463,700	467,458	343,927	347,334	343,867	347,201	−25.83	−25.70	−0.02	−0.04
7	208,758	211,288	156,625	158,917	156,584	158,816	−24.97	−24.79	−0.03	−0.06
8	313,955	317,052	235,080	237,894	235,025	237,766	−25.12	−24.97	−0.02	−0.05
9	471,564	475,351	354,173	357,603	354,110	357,464	−24.89	−24.77	−0.02	−0.04
10	511,844	515,795	368,146	371,678	368,079	371,522	−28.07	−27.94	−0.02	−0.04
small	393,964	397,389	291,590	294,685	291,533	294,554	−25.78	−25.63	−0.02	−0.05
11	893,455	898,668	675,504	680,265	675,410	680,041	−24.39	−24.30	−0.01	−0.03
12	615,247	619,576	437,315	441,095	437,236	440,915	−28.92	−28.81	−0.02	−0.04
13	633,408	637,802	449,383	453,199	449,312	453,026	−29.05	−28.94	−0.02	−0.04
14	1,123,860	1,129,706	845,941	851,215	845,844	850,986	−24.73	−24.65	−0.01	−0.03
15	959,813	965,224	700,426	705,220	700,338	704,997	−27.02	−26.94	−0.01	−0.03
medium	845,157	850,195	621,714	626,199	621,628	625,993	−26.82	−26.73	−0.01	−0.03
16	1,210,747	1,216,812	897,342	902,761	897,240	902,523	−25.89	−25.81	−0.01	−0.03
17	904,363	909,610	662,983	667,636	662,888	667,406	−26.69	−26.60	−0.01	−0.03
18	1,173,123	1,179,098	849,539	854,789	849,441	854,550	−27.58	−27.50	−0.01	−0.03
19	1,518,975	1,525,766	1,137,853	1,143,948	1,136,928	1,143,948	−25.09	−25.02	−0.08	0.00
20	1,353,813	1,360,231	955,347	960,910	955,242	960,653	−29.43	−29.36	−0.01	−0.03
large	1,232,204	1,238,303	900,613	906,009	900,554	905,867	−26.94	−26.86	−0.03	−0.02

GHz, 4 GB RAM). We were only able to solve very small instances to (proven) optimality. Solving the MIP using a stand-alone solver such as CPLEX as a black box was not able to tackle instances of reasonable size.

Table 8.4 reports the result we got after running some very small instances classified as *mini*. The maximum run time is varied between 1,000 and 1,000,000 seconds. The best bound (solution) found after a given maximum run time of t_{max} is denoted as b_{max} (z_{min}). Only the first and smallest instances can be solved to optimality. For instance 5 the gap after 1,000,000 seconds is still around 10%. We used the standard parameter setting provided. The test runs were executed using the default settings of CPLEX 10.1. A dash indicated that no solution was yet available. The penalty term β for gaps between

consecutive unloading operations, or starting the first delivery after the end of the time window, was set to 3.

Table 8.4.: VRP* solved as MIP

		n				
t_{max}		1	2	3	4	5
1,000	b_{max}	314.57	754.82	889.87	1116.43	636.78
	z_{min}	585.45	762.00	–	–	–
	gap in %	46.27	0.94	–	–	–
5,000	b_{max}	314.57	761.83	889.87	1116.45	637.51
	z_{min}	331.90	762.00	–	–	–
	gap in %	5.22	0.02	–	–	–
10,000	b_{max}	322.79	761.83	889.87	1116.72	637.51
	z_{min}	331.90	762.00	–	–	–
	gap in %	2.74	0.02	–	–	–
50,000	b_{max}	331.02	761.83	889.87	1118.20	641.01
	z_{min}	331.90	762.00	–	–	7766.60
	gap in %	0.27	0.02	–	–	91.75
100,000	b_{max}	331.02	762.00	889.87	1118.20	641.01
	z_{min}	331.90	762.00	–	–	719.70
	gap in %	0.27	0.00	–	–	10.93
500,000	b_{max}	331.02	762.00	889.87	1118.20	641.01
	z_{min}	331.90	762.00	–	–	715.47
	gap in %	0.27	0.00	–	–	10.41
1,000,000	b_{max}	331.02	762.00	889.87	1118.20	641.01
	z_{min}	331.90	762.00	–	–	715.47
	gap in %	0.27	0.00	–	–	10.41

8.3. Lower Bounds for VRP*

Rather than spending too much time waiting for getting a feasible integer solution when solving VRP* formulation as MIP we were able to achieve good bounds, using the two different variants described in Section 3.2.3. Based on the relaxed MIP formulation, considering fundamental constraints only, violated non-fundamental constraints as well as valid inequalities were added on demand. The only difference between the two variants refers to the *way* violated non-fundamental constraints and violated inequalities are going to be handled with. Variant 1 adds them as cuts during the process of optimization. Within Variant 2 the relaxed formulation is iteratively going to be solved. All violated valid inequalities and non-fundamental constraints were added as constraints. The total run time was set to t_{max}. The best bounds obtained after a total run time of 4800 seconds using each variant are shown in Table 8.5. At every step at most cut_{max} cuts (constraints) were added at every step. If more than cut_{max} constraints or valid inequalities were violated cut_{max} were chosen randomly among all violated ones. This approach was chosen in order to avoid adding too many constraints (cuts) simultaneously and hence ending up with memory issues. The best bounds obtained using each variant are highlighted in bold. A more detailed overview on bounds can be found in Section B.1.

Table 8.5.: Best Bounds after $t_{max} = 4800$ seconds

		Variant 1				Variant 2			
		cut_{max}							
class	n	100	500	1000	∞	100	500	1000	∞
mini	1	**331.02**	**331.02**	316.92	316.92	**331.02**	**331.02**	**331.02**	**331.02**
	2	**754.82**	**754.82**	754.43	754.43	754.63	754.65	754.65	754.65
	3	903.51	903.51	**923.69**	903.51	922.02	922.02	922.02	922.02
	4	1128.06	1130.67	1128.06	1128.06	1170.34	**1171.42**	**1171.42**	**1171.42**
	5	–	**642.07**	–	–	642.07	642.07	642.07	642.07
small	6	1298.32	1297.89	1298.24	1297.62	**1303.16**	**1303.16**	**1303.16**	**1303.16**
	7	**1070.99**	**1070.99**	**1070.99**	**1070.99**	–	–	–	–
	8	1416.50	1430.60	1429.72	**1430.79**	1422.29	1422.29	1422.29	1422.29
	9	1414.15	1414.15	1414.15	1414.15	**1414.73**	**1414.73**	**1414.73**	**1414.73**
	10	**1704.39**	**1704.39**	**1704.39**	**1704.39**	**1704.39**	**1704.39**	**1704.39**	**1704.39**
medium	11	2137.28	2159.68	**2163.22**	–	2157.27	2157.27	2157.27	2157.27
	12	1813.15	1813.15	1813.15	1813.15	**1813.97**	**1813.97**	**1813.97**	**1813.97**
	13	**1839.73**	1831.16	**1839.73**	**1839.73**	**1839.73**	**1839.73**	**1839.73**	**1839.73**
	14	1892.10	1921.26	1921.26	–	**1927.87**	**1927.87**	**1927.87**	**1927.87**
	15	1749.35	1769.91	1769.91	1769.91	**1785.26**	**1785.26**	**1785.26**	**1785.26**
large	16	2017.20	2025.04	2025.04	–	**2031.38**	**2031.38**	**2031.38**	**2031.38**
	17	**2046.67**	2032.69	2032.69	**2046.67**	2037.78	2037.77	2037.77	2037.77
	18	2080.05	2093.74	2093.74	2093.74	**2152.21**	**2152.21**	**2152.21**	**2152.21**
	19	2445.68	2445.68	2445.68	–	**2450.63**	**2450.63**	**2450.63**	**2450.63**
	20	2016.68	**2023.06**	**2023.06**	–	**2023.06**	**2023.06**	**2023.06**	**2023.06**

Note: adding at most cut_{max} cuts/constraints at every step.

A dash indicates that no results could be obtained due to memory issues. Variant 2 seems to work better. Compared to Variant 2, where cuts are iteratively added on a node level, on average stronger bounds can be obtained by iteratively adding constraints to the relaxed problem.

8.4. Integrative Hybrid Approach

The integrative hybrid approach (as described in Chapter 4) tries to improve any given solution by explicitly allowing certain decision variables to change with respect to the current incumbent solution. The initial solution is found by applying VNS (see Chapter 6) for a given time limit of t_{init} seconds. The remaining time of the total run time t_{max} is designated to repeatedly solving MIPs, while applying LB based on VNS.

Table 8.6.: Integrative Hybrid Approach

		t_{init}					
		1200		1800		2400	
class	n	z_{min}	z_{avg}	z_{min}	z_{avg}	z_{min}	z_{avg}
mini	1	**331.90**	331.90	**331.90**	331.90	**331.90**	331.90
	2	**762.00**	762.00	**762.00**	762.00	**762.00**	762.00
	3	**1007.73**	1038.56	**1007.73**	1019.89	**1007.73**	1034.46
	4	**1182.97**	1185.49	**1182.97**	1185.49	**1182.97**	1185.49
	5	670.42	681.27	**666.93**	676.06	670.42	676.76
small	6	1504.70	1522.89	**1473.23**	1503.69	1483.15	1500.39
	7	1135.88	1156.27	1131.92	1148.92	**1124.97**	1147.51
	8	1533.30	1542.12	1533.92	1541.99	**1532.42**	1541.76
	9	1531.58	1586.46	1527.28	1596.95	**1526.90**	1581.15
	10	2020.55	2083.81	**1966.52**	2039.22	1971.73	2024.79
medium	11	2481.57	2545.46	2469.95	2493.90	**2453.25**	2467.09
	12	**1939.60**	2001.48	1940.37	2008.85	1940.37	1988.17
	13	1979.87	2015.20	**1976.38**	2013.96	**1976.38**	2015.84
	14	2579.87	2777.94	2509.75	2610.40	**2462.18**	2582.00
	15	2098.30	2136.37	2003.62	2099.15	**2002.28**	2092.75
large	16	2293.63	2384.89	**2241.47**	2307.73	2273.98	2336.68
	17	2267.92	2299.36	2228.98	2268.45	**2215.23**	2262.07
	18	2532.10	2588.16	2505.95	2551.34	**2465.33**	2518.80
	19	–	–	–	–	–	–
	20*	–	–	–	–	**2266.32**	**2266.32**

* only one solution available for instance 20 (memory issues)

Table 8.4 presents the solutions values obtained when applying the integrative hybrid approach for a total run time limit of 4800 seconds. t_{init}, the time designated for the generation of an initial solution was varied between 1200 and 2400 seconds respectively.

The best solutions obtained (while varying t_{init}) are highlighted in bold. Columns heading z_{min} and z_{avg} represent the best and average solutions found. The algorithm can solve mini to medium sized instances without any problems. For two of the larger instances however we encountered memory problems. Instance 19 could not be solved due to memory issues.

A more detailed overview on the integrative hybrid approach using different total run times (t_{max}) and various parameter settings for the time designated for finding a good initial solution (t_{init}) can be found in Section B.2.

8.5. Reduced vs. Extensive MCNF Formulation

When using the extensive formulation one is able to consider additional constraints such as loading restrictions. The maximum number of trucks that can be loaded at any plant can be set to one. Additionally the choice of plant where loading operations should take place is no longer fixed but rather left to the optimization itself. However the resulting number of constraints and decision variable increases dramatically when using the extensive version (compared to the reduced formulation). Table 8.7 gives an overview on the number of decision variables and constraints in use given the reduced and extensive formulation for an exemplary input for 50 patterns per instance. The input patterns have been generated using the approach described in Section 5.1.1.

By using the reduced formulation the number of constraints and decision variables can be reduced by as much as 86% for mini-sized instances. The potential for reducing the matrix size decreases as the instance size increases. But still for large instances the number of decision variables (constraints) can be reduced by 39.98% (43.5%).

Table 8.8 plots the differences in solution runs obtained by using the reduced (see Section 5.2) and the extended (see Section 5.3) version for the MCNF formulation. In order to get a feeling for the consequences for a possible limitation in the number of trucks that can be loaded simultaneously the following setting was set up. A total number of 50 patterns were generated using the brute force approach. Then the MCNF formulation was solved using both approaches using the same pool of patterns. Run times can be decreased dramatically when as an additional restriction one does not have to make sure that at most one truck can be loaded at all plants at any point in time. For mini instances average run times could be decreased by 80%. As the size of problem instances increases this percentage even reaches 99.54% for large instances. For very small instances the consequences of this additional constraint have almost no effect on the objective function value. Few vehicle movements and resulting loading operations at plants tend not to overlap. As the size

Table 8.7.: Number of decision variables and constraints per instance (MCNF)

	extensive		reduced		% saved	
n	cols	rows	cols	rows	cols	rows
1	58,157	58,308	6,543	6,481	−88.75	−88.88
2	64,728	64,928	7,624	7,546	−88.22	−88.38
3	55,321	54,607	8,620	8,271	−84.42	−84.85
4	71,237	70,862	11,232	10,876	−84.23	−84.65
5	57,583	57,199	9,198	8,904	−84.03	−84.43
mini	61,405	61,181	8,643	8,416	−85.93	−86.24
6	66,971	65,164	16,918	15,682	−74.74	−75.93
7	58,037	57,034	14,565	13,900	−74.90	−75.63
8	68,737	67,347	19,384	18,310	−71.80	−72.81
9	72,388	70,559	24,254	22,822	−66.49	−67.66
10	69,528	67,570	27,049	25,273	−61.10	−62.60
small	67,132	65,535	20,434	19,197	−69.81	−70.93
11	70,888	67,950	28,457	25,961	−59.86	−61.79
12	74,724	72,611	33,347	31,026	−55.37	−57.27
13	72,099	70,022	30,530	28,016	−57.66	−59.99
14	80,362	76,966	38,607	34,652	−51.96	−54.98
15	78,897	76,021	36,599	33,071	−53.61	−56.50
medium	75,394	72,714	33,508	30,545	−55.69	−58.11
16	75,779	72,278	41,899	37,389	−44.71	−48.27
17	70,265	67,285	34,622	30,991	−50.73	−53.94
18	75,639	72,304	43,388	39,037	−42.64	−46.01
19	87,617	83,874	64,084	58,131	−26.86	−30.69
20	80,538	77,228	52,369	47,432	−34.98	−38.58
large	77,968	74,594	47,272	42,596	−39.98	−43.50

of the problem instance increases the effects of this additional restriction are noticeable. For medium instances relaxing this constraint on average involves an improvement of the solutions' quality by 4.92%, due to lower traveling times and less gaps between unloading operation. For medium and large instances the rate of improvement corresponds to 3.28 and 6.73% respectively. As practitioners told us that queues and bottleneck situations do not really seem to be a problem in reality we decided to use the reduced model formulation for all following test runs. The penalty term β_1 for gaps between consecutive unloading operations was set to 3. For penalizing orders for which no pattern has been chosen β_2 was set to 5000.

Table 8.8.: MCNF (reduced vs. extended version)

	extensive			reduced				
n	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	$\%z_{gap}$	$\%t_{gap}$
1	367.30	461.18	6.92	367.30	461.18	2.34	0.00	−66.13
2	928.08	1008.14	7.01	928.08	1005.59	2.46	−0.25	−64.95
3	1546.52	1602.55	20.81	1546.52	1594.44	2.94	−0.51	−85.87
4	1850.83	1875.05	18.78	1850.83	1875.05	3.55	0.00	−81.12
5	1050.95	1156.02	11.11	1050.95	1155.83	2.92	−0.02	−73.70
mini	1148.74	1220.59	12.92	1148.74	1218.42	2.84	−0.16	−74.35
6	7084.05	8204.05	570.55	7084.05	7295.48	7.84	−11.07	−98.63
7	1596.80	1707.88	38.86	1596.80	1707.88	5.18	0.00	−86.67
8	2551.23	3492.86	196.65	2551.23	3470.82	8.38	−0.63	−95.74
9	2746.90	3875.95	928.41	2746.90	3820.16	10.53	−1.44	−98.87
10	3638.05	6554.46	277.70	3405.92	5803.57	11.03	−11.46	−96.03
small	3523.41	4767.04	402.43	3476.98	4419.58	8.59	−4.92	−95.19
11	8554.95	11095.44	13509.55	8381.37	9401.47	29.89	−15.27	−99.78
12	3022.97	3182.70	3592.03	3022.97	3166.76	21.18	−0.50	−99.41
13	2964.65	3053.84	373.48	2964.65	3049.38	17.16	−0.15	−95.41
14	13165.50	14547.48	18568.33	13126.60	14246.68	81.27	−2.07	−99.56
15	8411.23	11822.75	3955.10	8408.03	11793.87	44.85	−0.24	−98.87
medium	7223.86	8740.44	7999.70	7180.72	8331.63	38.87	−3.65	−98.60
16	8401.47	9477.86	21142.32	8387.47	9464.01	130.62	−0.15	−99.38
17	8348.70	9487.36	4006.92	8348.70	9476.18	37.67	−0.12	−99.06
18	8819.95	10211.25	75041.09	8666.98	9887.88	152.20	−3.17	−99.80
19	5037.10	5207.28	19363.06	5006.97	5092.20	107.44	−2.21	−99.45
20	3918.62	5286.62	177177.30	3823.08	4711.64	71.36	−10.88	−99.96
large	6905.17	7934.08	59346.14	6846.64	7726.38	99.86	−3.30	−99.53

8.6. Initial Pattern Generation for MCNF

Two different methods in order to generate an initial pool of patterns have been implemented and are tested against each other. The two methods compared to each other are described in Section 5.1.1 and 5.1.1 respectively. The former approach generates a large number of patterns randomly. The latter version however only generates a smaller number of more intelligent (i.e. compatible) patterns. In both cases, based on the initial pool of base patterns, the MCNF was solved only once. To evaluate the quality of the set of fulfillment patterns on the performance on the MCNF - when it is going to be solved once - we compared the following settings. The algorithm has been tested on our 20 test instances. The MCNF was solved once with 100 base patterns generated randomly and 20 compatible patterns.

Table 8.9.: Comparison Initial Base Patterns

	random (r)			compatible (c)			c vs. r	
n	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	$\%z_{gap}$	$\%t_{gap}$
1	352.77	383.50	1.44	**333.13**	333.41	3.13	−13.06	117.82
2	811.62	853.17	1.74	**771.48**	819.04	3.09	−4.00	78.08
3	1450.65	1495.36	4.79	**1222.55**	1297.46	7.88	−13.23	64.57
4	1619.82	1659.10	4.78	**1327.47**	1403.63	6.12	−15.40	27.96
5	929.23	988.98	3.46	**796.15**	838.80	4.80	−15.19	38.89
mini	1032.82	1076.02	3.24	**890.16**	938.47	5.01	−12.18	65.46
6	2982.80	6151.30	74.23	**2168.05**	2236.54	118.29	−63.64	59.35
7	1407.12	1451.65	7.22	**1244.38**	1276.04	11.89	−12.10	64.68
8	2108.20	2178.06	12.41	**1922.65**	1947.91	22.06	−10.57	77.83
9	2284.02	2427.55	23.13	**2055.27**	2131.55	104.31	−12.19	350.93
10	2813.27	3047.74	30.22	**2480.60**	2519.20	59.97	−17.34	98.45
small	2319.08	3051.26	29.44	**1974.19**	2022.25	63.30	−23.17	130.25
11	3684.87	3873.28	371.87	**3136.22**	3173.99	1320.30	−18.05	255.05
12	2572.33	2645.08	33.03	**2376.78**	2506.69	279.25	−5.23	745.35
13	2501.37	2643.80	33.25	**2437.12**	2478.77	135.90	−6.24	308.68
14	4192.85	7514.34	3359.29	**3237.23**	3287.30	8064.79	−56.25	140.07
15	3249.75	5901.09	854.11	**2739.30**	2834.30	2003.82	−51.97	134.61
medium	3240.23	4515.52	930.31	**2785.33**	2856.21	2360.81	−27.55	316.75
16	3401.10	3520.92	1828.77	**2812.85**	2944.49	1802.65	−16.37	−1.43
17	3024.80	3128.63	259.18	**2652.65**	2767.51	1043.74	−11.54	302.71
18	3318.97	3535.87	557.42	**3179.18**	3260.93	3508.40	−7.78	529.40
19	3966.80	4133.45	5741.05	**3512.55**	3598.98	24195.52	−12.93	321.45
20	3110.68	3226.02	352.35	**2956.88**	2992.23	1715.96	−7.25	387.00
large	3364.47	3508.98	1747.75	**3022.82**	3112.83	6453.25	−11.17	307.83
all	2489.15	3037.94	677.69	**2168.12**	2232.44	2220.59	−18.52	205.07

Table 8.9 depicts the effects of the initial pool of patterns in use. The best (average) solution found using brute force or compatible patterns is denoted by z_{min} (z_{avg}). The necessary run times in seconds are given by t_{avg}. In the last two columns the deviation in terms of average solutions found ($\%z_{gap}$) and total run times ($\%t_{gap}$) is reported.

Using the compatible pattern generation instead of randomly generating a comparatively large number of initial base patterns dramatically helps to improve solution quality. On average the quality of the solution found can be improved by 18.52%. For mini (small) instances the solution can be improved by 12.18% (23.17%). For medium and large instances solution quality is improved by 27.55% and 11.17% respectively.

For a more detailed overview on solutions with different numbers of base patterns involved the reader is referred to Section B.3 and B.4 respectively. The number of brute force patterns was varied between 50 and 1000 patterns per order. On the other side 5 up to 20 compatible patterns were generated per order.

8.7. Cooperative Hybrid Approach

Two important parameters can be varied when executing our cooperative hybrid approach as described in Chapter 7. The number of iterations i used for consecutively executing the embedded MCNF and VNS component, along with the percentage of time p designated for the execution of the embedded MCNF component. The total run time will be equally split among all iterations. Within every iteration at most a fraction of p of the total run time designated to a single iteration will be spent for optimizing the MCNF. The remaining time is spent on VNS, which feeds back patters into the pool of pattern, enriches and diversifies it.

We tested different parameter settings for medium-sized instances and a run time limit of 600 and 1200 seconds respectively. The number of iterations i had been varied between 6 and 10, in steps of two. The percentage p of time designated for the embedded MCNF took values from 20% to 40%. The average results obtained are reported in Table 8.10. The numbers reported are averaged over all five medium-sized test instances (instance 11 to 15).

The algorithm is quite robust in terms of the parameter choice. Different settings lead to only to small variations in the average solutions found. More iterations given a fixed run time limit lead to a smaller amount of time designated for a single iteration. The more iterations we have given a fixed run time limit, the higher the percentage designated for the embedded MCNF component should be.

Table 8.10.: Parameter Study

	$t_{max} = 600$				$t_{max} = 1200$			
	p				p			
i	20%	30%	40%	avg	20%	30%	40%	avg
6	2354.25	2311.94	2331.90	2332.70	2266.41	2285.19	2318.37	2289.99
8	2355.00	2364.59	2320.95	2346.84	2281.67	2289.56	2299.03	2290.09
10	2367.64	2353.87	2339.53	2353.68	2309.02	2286.79	2281.53	2292.45
avg	2358.96	2343.47	2330.79		2285.70	2287.18	2299.64	

8.8. Cooperative Hybrid Approach vs. MCNF

The main difference between the cooperative hybrid approach (as described in Chapter 7) and the MCNF component (see Chapter 5), if applied solely, is the set and the generation of fulfillment patterns used during their optimization process. The cooperative hybrid approach in a sense depends on the embedded VNS component, which is responsible for generating good patterns and hereby enriching the pool of patterns to choose from. Whilst the MCNF can only rely on the set of fulfillment patterns generated initially by use of a greedy procedure. In order to evaluate the impact of the different pools of fulfillment patterns we set up the following test environment. First the hybrid method is executed for a given fixed maximum run time ($t_{max} = 150, \ldots, 4800$ seconds). Then we have a look at the number of patterns in the pool of patterns the procedure ends up with. The same number of patterns the cooperative hybrid method ends up with was generated (using compatible patterns) to initialize the MCNF model. The MCNF model is solved once with the same maximum run time limit imposed. Table 8.11 depicts the results obtained.

We report the best (z_{min}) and average (z_{max}) solutions found for both the MCNF component and our cooperative hybrid approach. The best results obtained are highlighted in bold. The improvement in percentage of the hybrid approach with respect to the MCNF component is indicated by $\%z_{gap}$.

Our hybrid approach clearly outperforms the MCNF component. Given the same number of patterns the best solutions are always obtained using our hybrid approach. Also on average the hybrid approach always finds better solution than the pure MCNF using the same number of patterns. This clearly shows the superiority of the embedded VNS component in order to generate good patterns and thereby enriching the pool of patterns. For mini instances the rate of improvement might not yet be so obvious, as the MCNF component with its greedy pattern generator is still able to achieve good quality solutions.

These experiments highlight the importance of the high-quality solutions generated by

Table 8.11.: Cooperative Hybrid vs. MCNF Approach

		MCNF		CHyb		
class	t_{max}	z_{min}	z_{avg}	z_{min}	z_{avg}	$\%z_{gap}$
mini	150	863.17	887.60	**800.08**	814.78	−8.20
	300	860.70	882.25	**793.77**	806.28	−8.61
	600	858.30	878.19	**794.67**	808.35	−7.95
	1200	857.79	872.69	**793.08**	803.66	−7.91
	2400	855.92	870.77	**791.10**	800.61	−8.06
	4800	850.19	862.76	**792.98**	798.66	−7.43
small	150	1900.55	2151.01	**1597.10**	1674.84	−22.14
	300	1870.40	1944.18	**1575.05**	1627.38	−16.29
	600	1868.81	1924.70	**1558.61**	1605.30	−16.59
	1200	1852.15	1909.95	**1560.06**	1600.45	−16.20
	2400	1827.22	1874.83	**1535.54**	1570.12	−16.25
	4800	1806.94	1859.73	**1546.52**	1578.15	−15.14
medium	150	2769.93	4190.80	**2550.09**	2729.95	−34.86
	300	2786.29	3606.75	**2322.64**	2457.30	−31.87
	600	2717.86	3355.17	**2268.40**	2364.59	−29.52
	1200	2641.14	2940.63	**2207.48**	2289.56	−22.14
	2400	2580.30	2739.30	**2190.75**	2260.75	−17.47
	4800	2599.96	2689.42	**2164.66**	2245.08	−16.52
large	150	3083.38	3765.57	**2922.04**	3195.76	−15.13
	300	2987.91	3692.10	**2732.46**	2842.27	−23.02
	600	2897.71	3044.75	**2465.38**	2612.09	−14.21
	1200	2857.92	2974.79	**2399.34**	2500.48	−15.94
	2400	2840.36	2951.44	**2408.09**	2493.39	−15.52
	4800	2817.14	2922.89	**2375.71**	2461.54	−15.78

the embedded VNS component, which is clearly a good choice when it comes to generate good compatible patterns. This clearly demonstrates the ability of the embedded VNS component in our hybrid approach to intelligently diversify and improve the quality of patterns within the pool of patterns, enabling the embedded MCNF to finding high quality solutions.

8.9. Cooperative Hybrid Approach vs. VNS

We wanted to test the effects and effectiveness of VNS (see Chapter 6) and our cooperative hybrid approach (as described in Chapter 7) based on the same run time limit t_{max}. The main difference between the VNS component and our cooperative hybrid approach is again the generation and usage of patterns. The hybrid approach can take a global view and consider all patterns held in the pool of patterns. Its memory is constantly updated as the embedded VNS component adds all patterns resulting from new best incumbent solution into the pool. The pure VNS component however is initiated with one fulfillment pattern per order. In every shaking step and the following local search step only a set of one pattern per order is considered simultaneously. The visibility is limited, all changes only take place on a rather local level.

In order to test the cooperative hybrid approach against the pure VNS variant the following set of experiments has been set up. Both algorithms were tested given a maximum total run time limit of t_{max} seconds. The VNS component is initiated with one pattern per order. Shaking and local search steps are executed iteratively until the total run time limit is reached. We initiated our cooperative hybrid approach with 15 fulfillment patterns per order (using the greedy procedure described in Section 5.1.1). The number of iterations (i) was set to 8. The percentage of run time devoted to the solution process of the embedded MCNF (p) was set to 30%, resulting in a remaining fraction of 70% designated for the execution of VNS.

We report the best (z_{min}) and average (z_{max}) solutions found for both the VNS component and our cooperative hybrid approach. The best results obtained are highlighted in bold. The improvement in percentage of the cooperative hybrid approach with respect to the VNS component is indicated by $\%z_{gap}$.

This again clearly demonstrates the superiority of the cooperative hybrid approach. The best solutions are always obtained by the cooperative hybrid version. Also on average the solutions obtained by the cooperative hybrid beat the VNS component. Please note that for medium and large instances the best solution obtained by VNS is outperformed by the average solution found by the hybrid approach. Not surprisingly both methods produce better results as the maximum total run time t_{max} increases. This clearly shows that the embedded VNS component used within the cooperative hybrid is highly useful to diversify the search effectively.

The solution values obtained for all classes are averaged over all individual instances within that class. The total run time devoted to every single run is denoted as t_{max}. The best and average solutions found are denoted as z_{min} and z_{avg} respectively. The best

Table 8.12.: Cooperative Hybrid vs. VNS

		VNS		CHyb		
class	t_{max}	z_{min}	z_{avg}	z_{min}	z_{avg}	$\%z_{gap}$
mini	150	809.04	825.77	**800.08**	814.78	−1.33
	300	808.64	824.58	**793.77**	806.28	−2.22
	600	808.64	823.41	**794.67**	808.35	−1.83
	1200	808.64	821.69	**793.08**	803.66	−2.19
	2400	807.95	819.80	**791.10**	800.61	−2.34
	4800	807.70	818.02	**792.98**	798.66	−2.37
small	150	1679.94	1796.89	**1597.10**	1674.84	−6.79
	300	1584.93	1683.56	**1575.05**	1627.38	−3.34
	600	1578.29	1644.61	**1558.61**	1605.30	−2.39
	1200	1573.76	1624.99	**1560.06**	1600.45	−1.51
	2400	1568.32	1611.83	**1535.54**	1570.12	−2.59
	4800	1568.32	1605.32	**1546.52**	1578.15	−1.69
medium	150	3341.22	3767.34	**2550.09**	2729.95	−27.54
	300	2875.96	3224.47	**2322.64**	2457.30	−23.79
	600	2475.64	2833.09	**2268.40**	2364.59	−16.54
	1200	2335.47	2477.99	**2207.48**	2289.56	−7.60
	2400	2282.88	2369.67	**2190.75**	2260.75	−4.60
	4800	2257.36	2345.65	**2164.66**	2245.08	−4.29
large	150	4035.64	4501.99	**2922.04**	3195.76	−29.01
	300	3528.55	3893.28	**2732.46**	2842.27	−27.00
	600	3182.03	3471.21	**2465.38**	2612.09	−24.75
	1200	2691.74	2876.55	**2399.34**	2500.48	−13.07
	2400	2556.84	2667.09	**2408.09**	2493.39	−6.51
	4800	2534.85	2619.46	**2375.71**	2461.54	−6.03

solution found - based on the same run time limit - in the direct comparison of VNS and the hybrid approach is highlighted in bold. The percental improvement of the solution found by our hybrid approach compared to the solution found by VNS is depicted in the last column and indicated by $\%z_{gap}$.

A more detailed overview of the results obtained for every single instance can be found in Table 8.13. The run time limit was set to 300 and 600 seconds respectively.

Table 8.13.: Cooperative Hybrid vs. VNS Approach (Details)

	$t_{max} = 300$ secs					$t_{max} = 600$ secs				
	VNS		CHyb			VNS		CHyb		
n	z_{min}	z_{avg}	z_{min}	z_{avg}	$\%z_{gap}$	z_{min}	z_{avg}	z_{min}	z_{avg}	$\%z_{gap}$
1	**331.90**	331.90	**331.90**	331.90	0.00	**331.90**	331.90	**331.90**	331.90	0.00
2	**762.00**	762.51	**762.00**	762.51	0.00	**762.00**	762.51	**762.00**	762.51	0.00
3	1090.90	1134.98	**1021.58**	1062.30	−6.40	1090.90	1132.15	**1026.07**	1073.17	−5.21
4	**1182.97**	1192.63	**1182.97**	1185.39	−0.61	**1182.97**	1190.92	**1182.97**	1185.39	−0.46
5	675.45	700.87	**670.42**	689.31	−1.65	675.45	699.56	**670.42**	688.79	−1.54
mini	808.64	824.58	**793.77**	806.28	−1.73	808.64	823.41	**794.67**	808.35	−1.44
6	1590.83	1673.09	**1581.68**	1639.95	−1.98	1589.95	1642.54	**1510.13**	1585.68	−3.46
7	1161.83	1228.30	**1118.00**	1140.30	−7.16	1161.83	1226.03	**1131.77**	1144.55	−6.65
8	**1534.13**	1600.29	1541.45	1585.43	−0.93	1534.13	1588.07	**1510.75**	1550.60	−2.36
9	1556.23	1689.96	**1547.87**	1615.70	−4.39	1547.97	1627.58	**1543.00**	1616.30	−0.69
10	**2081.63**	2226.19	2086.23	2155.51	−3.17	**2057.55**	2138.81	2097.42	2129.38	−0.44
small	1584.93	1683.56	**1575.05**	1627.38	−3.53	1578.29	1644.61	**1558.61**	1605.30	−2.72
11	3185.18	3502.68	**2632.18**	2748.89	−21.52	2628.63	2836.28	**2601.60**	2688.99	−5.19
12	2127.08	2436.70	**1953.87**	2071.51	−14.99	2056.60	2177.31	**1926.78**	2034.41	−6.56
13	2040.92	2191.52	**2025.88**	2059.70	−6.02	2040.92	2104.50	**2022.67**	2050.10	−2.58
14	3705.90	4256.65	**2784.83**	3025.02	−28.93	3201.90	3814.48	**2634.17**	2834.58	−25.69
15	3320.70	3734.78	**2216.43**	2381.39	−36.24	2450.13	3232.86	**2156.77**	2214.86	−31.49
medium	2875.96	3224.47	**2322.64**	2457.30	−21.54	2475.64	2833.09	**2268.40**	2364.59	−14.30
16	3367.17	3986.32	**2616.25**	2728.93	−31.54	3052.72	3353.09	**2277.22**	2440.06	−27.23
17	3224.45	3492.57	**2312.23**	2409.69	−31.01	2736.68	3100.87	**2215.73**	2323.32	−25.08
18	3483.98	3816.04	**2849.75**	2958.78	−22.46	3038.58	3442.29	**2523.17**	2705.24	−21.41
19	4206.72	4366.80	**3347.53**	3463.85	−20.68	3854.63	3978.70	**2939.25**	3094.95	−22.21
20	3360.42	3804.66	**2536.55**	2650.09	−30.35	3227.52	3481.08	**2371.52**	2496.88	−28.27
large	3528.55	3893.28	**2732.46**	2842.27	−27.21	3182.03	3471.21	**2465.38**	2612.09	−24.84
total	2199.52	2406.47	**1855.98**	1933.31	−13.50	2011.15	2193.08	**1771.77**	1847.58	−10.83

Based on a run time of 300 seconds the hybrid approach finds the best solution in 18 out of 20 instances. The average solutions obtained by the hybrid approach improve the average solution found by VNS by 1.73% (3.53%) for mini (small) instances and 21.54% (27.21%) for medium (large) instances. Over all instances the average solution found can be improved by 13.5%. For a run time of 600 seconds in 19 out of 20 instances the best solution found is obtained by using our hybrid approach. The average solution found by our hybrid approach over all instances can be improved by 10.83%. For small, medium

and large instance the improvement of the average solution found compared to the pure VNS variant is 2.72%, 14.3% and 24.84% respectively.

A even more comprehensive overview on the results obtained by VNS using different run time limits are illustrated in Section B.5.

8.10. Cooperative Hybrid Approach vs. Integrative Hybrid Approach

Finally we wanted to test the effectiveness of both hybrid approaches against each other. The cooperative approach (see Chapter 7) repeatedly solves problems based on MCNF formulations. It is basically a pattern selection mechanism. The fulfillment patterns at hand are generated using a greedy procedure. Moreover all patterns which are part of any new best incumbent solution found during the process of VNS are added. Hence the pool of pattern is currently updated and enriched allowing the MCNF to select good and compatible patterns. The integrative hybrid approach (as described in Chapter 4) however is based on a formulation for the VRP* formulation, which is iteratively solved. In each iteration, depending on the current Neighborhood $\mathcal{N}_\kappa$ certain decision variables are allowed to change with respect to the current incumbent solution. This systematic approach is inspired by LB and VNS. Table 8.14 presents the values obtained. The total runtime limit

Table 8.14.: Cooperative vs. Integrative Hybrid Approach

		IHyb		CHyb		
class	t_{max}	z_{min}	z_{avg}	z_{min}	z_{avg}	$\%z_{gap}$
mini	150	**798.88**	832.79	800.08	814.78	−2.16
	300	795.54	827.95	**793.77**	806.28	−2.62
	600	795.54	817.21	**794.67**	808.35	−1.08
	1200	794.53	807.60	**793.08**	803.66	−0.49
	2400	794.53	804.04	**791.10**	800.61	−0.43
	4800	**791.00**	798.12	792.98	798.66	0.07
small	150	1718.40	1794.06	**1597.10**	1674.84	−6.65
	300	1613.47	1701.95	**1575.05**	1627.38	−4.38
	600	1586.93	1639.48	**1558.61**	1605.30	−2.08
	1200	**1549.86**	1599.03	1560.06	1600.45	0.09
	2400	1554.60	1584.98	**1535.54**	1570.12	−0.94
	4800	**1527.83**	1559.12	1546.52	1578.15	1.22
medium	150	2851.23	3306.39	**2550.09**	2729.95	−17.43
	300	2660.16	2905.36	**2322.64**	2457.30	−15.42
	600	2391.57	2599.50	**2268.40**	2364.59	−9.04
	1200	2273.30	2370.47	**2207.48**	2289.56	−3.41
	2400	2218.30	2307.91	**2190.75**	2260.75	−2.04
	4800	2166.89	2229.17	**2164.66**	2245.08	0.71
large*	150	3355.97	3672.47	**2747.35**	3029.02	−17.52
	300	2881.34	3195.51	**2592.74**	2699.13	−15.53
	600	2570.67	2778.96	**2338.71**	2489.54	−10.41
	1200	2437.35	2538.57	**2273.89**	2375.65	−6.42
	2400	2370.15	2431.15	**2281.87**	2382.79	−1.99
	4800	2318.18	2372.52	**2278.62**	2369.05	−0.15

(* instances 19 and 20 omitted, as only few solutions available for integrative hybrid approach)

t_{max} was varied between 150 and 4800 seconds. For the cooperative hybrid approach the number of iterations was set to 8. Within every iteration 30% of the designated run time was dedicated to solving the MCNF, the remaining time is allocated to the VNS. The same amount of run time was designated for the execution of the integrative hybrid approach. The best results obtained per instance are highlighted in bold. z_{min} (z_{avg}) denotes the best (average) solution for all instances for both approaches.

The results show that the cooperative approach is still better than the integrative hybridization. However some new best solutions were found by the integrative hybrid approach.

8.11. Cooperative Hybrid Approach vs. Simulated Annealing

A company in Austria developed and sold a tool designated for solving full truck load problems. Their approach is based on SA. SA is inspired by means of statistical mechanics and the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature (see Kirkpatrick et al., 1983). We compare the results obtained by our hybrid approach based on a total run time of 150 seconds, which is typically the amount of time the tool based on SA needs in order to terminate. All results are depicted in Table 8.15.

The results obtained by our hybrid approach represent average values over 5 runs. All results obtained by the approach based on SA are averaged over 25 runs. The number of cooling phases was varied between 1 and 5, with 5 runs executed at every stage. The best solution found is highlighted in bold. The best solution found per instance is always found by our hybrid approach, which also on average finds the best solutions. For small instances the average solution quality can be increased by 38.34% at comparable run times. For medium (large) instances the solution can be improved by 37.88% (38.09%) on average compared to a reduction in run time of 13.6% (14.02%)

Table 8.15.: Cooperative Hybrid vs. SA: Comparison of solution quality and run time

	SA			CHyb				
n	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	$\%z_{gap}$	$\%t_{gap}$
1	333.64	363.88	102.84	**331.90**	331.90	152.46	−8.79	48.25
2	778.58	834.95	60.84	**762.00**	762.51	152.72	−8.68	151.02
3	1180.89	1424.20	152.32	**1053.12**	1104.48	153.36	−22.45	0.68
4	1379.54	1586.97	134.48	**1182.97**	1186.61	153.92	−25.23	14.45
5	723.60	935.26	127.96	**670.42**	688.38	153.35	−26.40	19.84
mini	879.25	1029.05	115.69	**800.08**	814.78	153.16	−18.31	46.85
6	2337.61	2880.06	155.56	**1599.85**	1732.60	157.13	−39.84	1.01
7	1429.77	1672.89	137.64	**1139.53**	1157.88	155.37	−30.79	12.88
8	2077.83	2464.71	163.96	**1547.82**	1578.51	158.34	−35.96	−3.43
9	2409.88	2921.99	177.44	**1575.98**	1654.95	159.52	−43.36	−10.10
10	3223.41	3863.32	202.84	**2122.33**	2250.26	163.20	−41.75	−19.54
small	2295.70	2760.59	167.49	**1597.10**	1674.84	158.71	−38.34	−3.84
11	4057.40	4593.99	218.48	**2698.88**	2909.58	168.25	−36.67	−22.99
12	3110.75	3866.32	170.16	**2052.37**	2193.88	168.72	−43.26	−0.85
13	3231.78	3781.86	185.52	**2100.62**	2178.18	170.71	−42.40	−7.98
14	4215.76	5135.75	231.60	**3333.25**	3641.25	177.11	−29.10	−23.53
15	3670.80	4396.75	203.92	**2565.32**	2726.87	178.11	−37.98	−12.66
medium	3657.30	4354.93	201.94	**2550.09**	2729.95	172.58	−37.88	−13.60
16	3810.40	4673.39	215.08	**2673.67**	2984.90	183.61	−36.13	−14.63
17	4016.47	4510.84	207.92	**2444.65**	2692.13	177.17	−40.32	−14.79
18	4654.08	5587.77	211.20	**3123.72**	3410.02	185.99	−38.97	−11.94
19	5347.55	6027.76	245.16	**3499.70**	3734.63	190.76	−38.04	−22.19
20	4358.41	5008.30	204.68	**2868.47**	3157.13	191.27	−36.96	−6.55
large	4437.38	5161.61	216.81	**2922.04**	3195.76	185.76	−38.09	−14.02
all	2894.55	3413.93	177.45	**2012.79**	2155.83	168.42	−33.39	3.00

8.12. Summary

A comprehensive overview on the best results and bound obtained using the various approaches explained throughout this thesis can be found in Table 8.12. Three different methods for finding good lower bounds have been implemented. The first column of results corresponds to the best bounds obtained after letting run the formulation based on VRP* (as described in Chapter 3) using CPLEX as a black box for 1 million seconds. The second and third column headed by V1 and V2 correspond to the best bounds obtained when applying the iterative relaxation or solving the MIP including cuts respectively. At most 100 cuts or constraints were added on each step ($cut_{max} = 100$). A detailed description of the two approaches can be found in Section 3.2.3. The best bound obtained is highlighted in bold.

In terms of solution quality we compare the best (feasible) solutions found using integrative hybrid approach (as described in Chapter 4), solving the MCNF formulation (see Chapter 5) and the pure VNS approach (as described in Chapter 6). Furthermore we include the best results obtained using our cooperative hybrid approach (see Chapter 7) and the ones obtained using the tool based on SA, which is currently in use in a company in Austria. The integrative hybrid approach was run for a total run time limit of $t_{max} = 4800$ seconds. $t_{init} = 2400$ seconds were spent on finding a good starting solution. The MCNF was run (without any run time limit) given $p_{init} = 200$ (20) patterns generated brute force (using the compatible pattern generation) respectively. The very same instances were solved using the pure VNS approach for a total run time of $t_{max} = 4800$ seconds. The cooperative hybrid approach was solved given the same run time limit. The total run time limit was equally split among $i = 8$ iterations. Within every iteration $p = 30\%$ were spent for solving the embedded MCNF formulation. The best (average) solutions were obtained, after having executed 5 independent runs per instance. All best solutions are based on 5 independent runs. For the tool based on SA the following setting was chosen: the number of cooling phases was varied between 1 and 5, with 5 independent runs executed at every stage. The best results obtained are highlighted in bold.

The table clearly shows the superiority of both hybrid approaches. On average the integrative hybrid approach dominates the cooperative version. The average solutions found by the integrative hybrid version improve the ones found by the cooperative one by 2.13% based on a run time limit of 4800 seconds. For solving large scale problem instances however the cooperative version is much more reliable, as the integrative hybrid approach runs into memory problems for instances with more than 70 orders.

Regarding the pure VNS variant the cooperative hybrid approach on average finds

solutions which improve the ones found by VNS by 4.13% (based on the the same run time limit of 4800 seconds). For lower run times the difference is even higher. The tool based on SA has been outperformed. The average solutions found by our cooperative hybrid approach improve the ones found by SA by 46.77%.

For obtaining good lower bounds we executed our two Variants V1 and V2 for 4800 seconds respectively, using valid inequalities added on demand. On the other side CPLEX was ran for 500,000 seconds. The gap between our best solutions found and the lower bounds is remarkable. For mini and small instances the best solutions found the gap is as small as 2.71 and 8.86% respectively. For medium and large instances the gap only reaches 10.6 and 9.41%.

Table 8.16.: Summary of Results

	best bounds			best solutions z_{min}							average solutions z_{avg}					
n	C[1]	V1[2]	V2[3]	IHyb[4]	MCNF[5]	MCNF[6]	VNS[7]	CHyb[8]	SA[9]	gap	IHyb[4]	MCNF[5]	MCNF[6]	VNS[7]	CHyb[8]	SA[9]
1	331.02	**331.02**	**331.02**	**331.90**	333.13	333.13	**331.90**	**331.90**	333.64	0.27	**331.90**	345.99	333.41	**331.90**	**331.90**	363.88
2	**762.00**	754.82	754.63	**762.00**	769.53	771.48	**762.00**	**762.00**	778.58	0.00	**762.00**	782.50	819.04	762.51	**762.00**	834.95
3	889.87	903.51	**922.02**	**1007.73**	1243.77	1222.55	1089.68	1021.08	1180.89	8.51	**1034.46**	1352.98	1297.46	1112.31	1036.69	1424.20
4	1118.20	1128.06	**1170.34**	**1182.97**	1436.48	1327.47	**1182.97**	**1182.97**	1379.54	1.07	1185.49	1486.39	1403.63	1186.11	**1184.60**	1586.97
5	641.01	–	**642.07**	670.42	834.52	796.15	671.97	**666.93**	723.60	3.73	**676.76**	888.29	838.80	697.29	678.10	935.26
mini	748.42	**779.35**	764.01	**791.00**	923.49	890.16	807.70	792.98	879.25	2.71	**798.12**	971.23	938.47	818.02	798.66	1029.05
6	1118.20	1298.32	**1303.16**	**1483.15**	2297.13	2168.05	1568.72	1524.52	2337.61	12.14	**1500.39**	2346.93	2236.54	1585.63	1573.61	2880.06
7	1063.63	**1070.99**	–	1124.97	1243.03	1244.38	1148.02	**1120.32**	1429.77	4.40	1147.51	1320.50	1276.04	1189.58	**1127.76**	1672.89
8	1411.08	1416.50	**1422.29**	1532.42	1952.98	1922.65	1534.13	**1527.23**	2077.83	6.87	**1541.76**	1991.37	1947.91	1571.76	1544.42	2464.71
9	1414.15	1414.15	**1414.73**	**1526.90**	2009.13	2055.27	1535.02	1529.35	2409.88	7.35	1581.15	2240.99	2131.55	1583.53	**1553.17**	2921.99
10	1675.15	**1704.39**	**1704.39**	**1971.73**	2733.05	2480.60	2055.73	2031.20	3223.41	13.56	**2024.79**	2785.79	2519.20	2096.12	2091.77	3863.32
small	1336.44	1380.87	**1461.14**	**1527.83**	2047.06	1974.19	1568.32	1546.52	2295.70	8.86	**1559.12**	2137.12	2022.25	1605.32	1578.15	2760.59
11	2137.28	2137.28	**2157.27**	2453.25	3162.70	3136.22	2566.92	**2395.48**	4057.40	9.94	**2467.09**	3390.79	3173.99	2667.59	2532.27	4593.99
12	1813.15	1813.15	**1813.97**	1940.37	2355.42	2376.78	1921.03	**1884.97**	3110.75	3.77	1988.17	2397.36	2506.69	2007.63	**1971.01**	3866.32
13	1829.73	**1839.73**	**1839.73**	**1976.38**	2348.83	2437.12	2030.32	1993.88	3231.78	6.91	2015.84	2411.31	2478.77	2047.20	**2011.33**	3781.86
14	1892.10	1892.10	**1927.87**	2462.18	3527.02	3237.23	2592.85	**2457.80**	4215.76	21.56	2582.00	3639.64	3287.30	2784.32	**2580.00**	5135.75
15	1757.30	1749.35	**1785.26**	**2002.28**	2843.18	2739.30	2175.70	2091.17	3670.80	10.84	**2092.75**	2893.83	2834.30	2221.53	2130.80	4396.75
medium	1885.91	1886.32	**1904.82**	2166.89	2847.43	2785.33	2257.36	**2164.66**	3657.30	10.60	**2229.17**	2946.59	2856.21	2345.65	2245.08	4354.93
16	2017.20	2017.20	**2031.38**	**2273.98**	3055.58	2812.85	2441.40	2300.85	3810.40	10.67	**2336.68**	3111.70	2944.49	2496.31	2380.36	4673.39
17	2028.88	**2046.67**	2037.78	2215.23	2703.47	2652.65	2346.05	**2167.77**	4016.47	5.59	2262.07	2781.07	2767.51	2380.84	**2224.42**	4510.84
18	2080.05	2080.05	**2152.21**	2465.33	3014.60	3179.18	2533.83	**2367.25**	4654.08	9.08	2518.80	3078.77	3260.93	2562.18	**2502.38**	5587.77
19	2445.68	2445.68	**2450.63**	–	3695.02	3512.55	3073.52	**2853.68**	5347.55	14.12	–	3781.13	3598.98	3214.05	**2924.99**	6027.76
20	2016.68	2016.68	**2023.06**	2266.32	2782.55	2956.88	2279.45	**2189.02**	4358.41	7.58	**2266.32**	2941.22	2992.23	2443.95	2275.55	5008.30
large	2117.70	2121.26	**2139.01**	**2305.22**	3050.24	3022.82	2534.85	2375.71	4437.38	9.41	**2345.97**	3138.78	3112.83	2619.46	2461.54	5161.61
all	1522.12	1541.95	**1567.25**	**1697.74**	2217.06	2168.12	1792.06	1719.97	2817.41	7.90	**1733.09**	2298.43	2232.44	1847.12	1770.86	3326.55

[1] VRP* on CPLEX for 500,000 seconds
[2] Variant 1 (iterative relaxation): $t_{max} = 4800$ seconds, $cut_{max} = 100$
[3] Variant 2 (MIP including cuts): $t_{max} = 4800$ seconds, $cut_{max} = 100$
[4] Integrative Hybrid Approach: $t_{max} = 4800$ seconds, $t_{init} = 2400$ seconds
[5] MCNF using $p_{init} = 200$ patterns per order generated brute force
[6] MCNF using $p_{init} = 20$ compatible base patterns per order
[7] pure VNS: $t_{max} = 4800$ seconds
[8] Cooperative Hybrid Approach: $i = 8, p = 30\%, t_{max} = 4800$ seconds

9. Conclusion

In this thesis we presented some highly effective solution procedures for solving full truck routing problems for the delivery of ready-mixed concrete. Most of the approaches themselves are not restricted to this application exclusively. They can easily be extended in order to be applied for other problem classes. Typically the demand of a single order exceeds the capacity of any single truck available, hence every order needs to be split into several deliveries. As the fleet of trucks under consideration is supposed to be heterogeneous in terms of capacity the exact number of deliveries to be executed cannot be predetermined. All consecutive unloading operations corresponding to one single order cannot overlap and the gaps in between should be kept as small as possible. Additionally we consider multiple depots. Trucks are positioned at one (called their home) depot and need to return back there by the end of the day. Additionally we are taking into account special unloading equipment required by some constructors. If a special unloading equipment such as a pump or a belt is needed the first truck to arrive needs to be equipped accordingly, it needs to assist all later arriving trucks with their unloading operation and stays until the last delivery has been finished.

The first approach chosen is based on an extended version of the classical formulation for *vehicle routing problems* (VRP*). It has been extended in a sense that we allow (or even require) multiple visits to our customers. We consider the multiple depot case with a heterogeneous fleet of vehicles. Any single order is split into several deliveries and all consecutive deliveries need to be scheduled properly. The resulting MIP-formulation is highly accurate and theoretically would guarantee finding an optimal solution. In practice however the resulting run times are far too high and finding good feasible solutions - if any - cannot be guaranteed in a reasonable amount of time. For a detailed description of the model formulation see Chapter 3. Some results for small instances can be found in Section 8.2. For the best bounds obtained the reader is referred to Section 8.3. Only two of the very smallest instances consisting of 13 and 14 orders respectively could be solved to optimality. All remaining instances given a total run time limit of one million seconds could not be solved to optimality.

Alternatively we developed a model based on a *multi-commodity network flow formulation* (MCNF). This formulation works on a set of *patterns* used as input. A pattern represents an option concerning how an order could be satisfied. It uniquely specifies the exact timing of all unloading operations and the sequence of trucks to execute them. Every pattern itself is feasible from an order's point of view. Before initiating the MCNF formulation several patterns per order are generated and fed into the pool of patterns. The main task of MCNF is to select one pattern per order and making sure that everything turns out to be feasible from the trucks' point of view. Given a limited number of input patterns it only has a restricted view on the solution space. However the formulation itself can be solved to optimality, given the pool of patterns, in a reasonable amount of time. See Section Chapter 5 for a detailed description of the model formulation. Up to 10000 patterns have been randomly generated and get into the model formulation, but the results were not even able to come close to the ones obtained by our hybrid approaches. See Section 8.6. A more detailed overview on the results obtained can be found in Section B.3 and Section B.4 respectively.

Both previous approaches rely on a commercial solver such as CPLEX or XPRESS used as a black box. In order to go without a solver we developed a metaheuristic approach based on *variable neighborhood search* (VNS). VNS is a highly effective metaheuristic which is able to both explore and intensify the search within the solution space, starting from any initial solution. This method is able to score in terms of performance and is a rather simple but highly flexible approach. Finding a global optimum solution however is not guaranteed. So far we do not consider ascent moves. The formulation itself however can easily be adapted in order to incorporate such features as well. This approach based on VNS is a highly cost-efficient way to solve problems. It does not rely on any commercial solver such as XPRESS or CPLEX, but can be run on any personal computer. It guarantees finding a feasible solution. The run time can be set at discretion. The quality of solutions obtained is good. For a detailed description of the implemented algorithm see Chapter 6. Detailed results using various run time limits are depicted in Section B.5.

Hybrid approaches constitute a relatively new and highly ambitious field of study in the scientific community. By combining exact methods and heuristic or metaheuristic approaches we are able to combine their strengths and overcome their major drawbacks and disadvantages. All three approaches described before can be applied solely. However, when incorporated within an intelligent hybridization framework they are able to produce far better results.

As just "to MIP" the formulation based on VRPs was not the way to success we decided

to help and guide the optimization process by means of an *integrative hybrid* approach. A local branching (LB) scheme inspired by the concept of VNS and its resulting neighborhood structures has been developed in order to assist and guide the solution process of the MIP-optimization. Given any feasible solution, where most of the decision variables remain fixed, certain parts of the solution space can consciously be explored further by resetting bounds of previously fixed decision variables. This results in highly complex and specialized framework, where we - using our problem specific knowledge - are able to efficiently explore the solution space. Up to medium-sized problem instances can be solved while finding high quality results in a reasonable amount of time. This approach on average performs best for all mini, small and medium-sized instances. The best solutions for 10 out of 15 instances were found by our integrative hybrid approach. For large instances however with more than 65 orders this algorithm is no longer the preferred choice, as the required amount of memory increases and the problem instances no longer could be solved on our computers. For a detailed description of this hybrid framework the reader is referred to Chapter 4. A detailed overview on the results obtained can be found in Section 8.4 and Section B.2 respectively.

The second hybridization framework is *cooperative* in nature. Within this approach both MCNF and VNS communicate with each other. The best solution found by MCNF serves as an initial solution for VNS. All new best solutions found during the process of VNS will be fed back into the pool of patterns used by MCNF. This iterative procedure works highly effective. VNS is able to improve any given solution using the embedded shaking and local search operators and constantly updates and enriches MCNF's pool of patterns. MCNF on the other hand is able to take a global view over all patterns and picks compatible ones while scheduling exact timing of all truck movements. This approach has been proven to work highly efficient. Even large scale instances with up to 76 orders (resulting in up to 200 individual deliveries to be executed. The results obtained using this hybrid approach are of high quality and outperform the results obtained by each of its components when applied solely. This approach is able to solve even the largest problem instances at hand. For mini, small and medium instances the results obtained by this approach are comparable to the ones obtained by the integrative hybrid approach. For 9 out of 15 instances the best known solution could be obtained. Compared to the previous hybrid version this framework is even able to solve large instances and achieve high quality solutions. A comprehensive overview on this cooperative hybrid framework is given in Chapter 7. Detailed results can be found in Section B.6. A comparison of the performance of this approach compared to its components MCNF and VNS - if applied solely - are shown in Section 8.8 and Section 8.9. On average - over all instances

and based on the same total run time - the results obtained by the cooperative hybrid approach improve the ones obtained by MCNF (VNS) by 16.78% (9.22%). Compared to the tool based on SA we are able to improve their results - after a run time limit of 150 seconds by 33.39%. For a comparison of the two hybrid approaches the reader is referred to Section 8.10.

From a scientific point of view both the cooperative and hybrid approach are highly competitive and produce high-quality solutions. The latter mentioned cooperative hybrid approach works extremely well. However the framework is highly sophisticated. The former mentioned integrative hybrid approach is able to find some new best solutions while still taking the second place after the cooperative version. Especially when tackling to solve large instances we are not insusceptible from running into memory problems.

For a company planning to invest in a tool able to tackle problems like this, the author's recommendation looks as follows. Depending on the budget of a company VNS might be an appropriate choice for solving daily scheduling problems for the delivery of ready-mixed concrete. Good solutions can be obtained within a reasonable amount of time. For scheduling a company's operation on a larger scale it does make sense to decide in favor of a hybrid framework. However this includes the acquisition of a commercial solver able to solve large-scale MIP-formulations.

A. Abbreviations and Notation

A.1. Abbreviations

Abbreviation	Description
ACO	Ant Colony Optimization
CHyb	Cooperative Hybrid Approach
CPM	Critical Path Method
CVRP	Capacitated Vehicle Routing Problem
CVRPTW	Capacitated Vehicle Routing Problem with Time Windows
DSS	Decision Support System
GA	Genetic Algorithm
IHyb	Integrative Hybrid Approach
LP	Linear Programming
MCNF	Multi Commodity Network Flow
MDVRPTW	Multi Depot Vehicle Routing Problem with Time Windows
MIP	Mixed Integer Programming
RMC	Ready Mixed Concrete
SA	Simulated Annealing
SDMDHVRPTW	Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows (aka VRP*)
SDVRP	Split Delivery Vehicle Routing Problem
SDVRPTW	Split Delivery Vehicle Routing Problem with Time Windows
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRP*	see SDMDHVRPTW
VRPFL	Vehicle Routing Problem with Full Truckloads
VRPTW	Vehicle Routing Problem with Time Windows

A.2. Definition of Sets

Symbol	Description
P	set of all plants p
O	set of all orders o
O'	set of orders requiring special unloading equipment ($O' \subset O$)
D_o	set of deliveries for order o
T	set of all points in time t
K	set of all trucks k
C	set of all classes of trucks c
A_o	set of all patterns for order o

A.3. General Data

Symbol	Description
D_{min}	minimum number of deliveries per instance
D_{max}	maximum number of deliveries per instance
d^o_{min}	minimum number of deliveries for order o
D^o_{max}	maximum number of deliveries per order o
n	instance
n_o	number of orders per instance
z_{min}	best solution found
z_{avg}	average solution found
z_{gap}	percentage gap in average solutions found
b_{max}	best lower bound
t_{max}	run time limit in seconds
t_{init}	run time limit for finding a starting solution (IHyb)
i	number of iterations (cooperative hybrid approach)
p	percentage of time spent on execution of MCNF (CHyb)
cut_{max}	maximum number of cuts to be added per step
Q_o	ordered demand (in m^3) for order o
s_o	start of time window associated with order o
e_o	end of time window associated with order o
$oinstr_o$	unloading equipment required by order o

cap_k	capacity of truck k
cap_c	capacity of truck in class c
cap_{min}	capacity of truck with smallest capacity
cap_{max}	capacity of largest truck
$cap_{max}^{oinstr_o}$	capacity of the largest truck equipped with the instrumentation demanded by order o
$cap_{min}^{oinstr_o}$	capacity of the smallest truck equipped with the instrumentation demanded by order o
$tinstr_k$	type of unloading equipment of truck k
$tinstr_c$	type of unloading equipment for trucks within class c
p_k	home plant of truck k
p_c	home plant of trucks within class c
n^{p_c}	number of trucks of class c at plant p
UR_o	unloading rate at construction site associated with order o
U_o^k	time required to unload truck k at construction site of order o
U_o^c	time required to unload truck of class c at construction site of order o
L_p^c	time required to load truck of class c at plant p
$TT_{p,o}$	travel time from plant p to construction site of order o
$TT_{o,p}$	travel time from order o to plant p
$TTL_{p,o}^k$	travel time (from plant p to order o) + time for loading of truck k at p
$TTL_{p,o}^c$	travel time (from plant p to order o) + time for loading of truck of class c at p
TTL_{o_1,o_2}^k	travel time (from order o_1 to order o_2 via closest plant en route) plus time for loading truck k there
TTL_{o_1,o_2}^c	travel time (from order o_1 to order o_2 via closest plant en route) plus time for loading truck of class c there
d_{min}^o	minimum number of deliveries required for order o
$cumCap(pat_o^-)$	capacity of trucks what will be removed from pattern of order o
$cumCap(pat_o^+)$	capacity of trucks that will be inserted into pattern of order o

A.4. Decision Variables for VRP*

Symbol	Description
$z_{o,d}$	binary, evaluates to 1 if delivery d for order o will be executed
$a_{o,d}$	start of delivery d for order o

$\underline{a_{o,d}}$	earliest possible start of delivery d for order o
$b_{o,d}$	end of delivery d for order o
$y^k_{o,d}$	binary, evaluates to 1 if truck k executes delivery d for order o
$late_o$	gap before first delivery for order o
$xp2o^k_{o,d}$	binary, evaluates to 1 if truck k serves delivery d of order o first
$xo2p^k_{o,d}$	binary, evaluates to 1 if delivery d of order o is the last one served by truck k
$xo2o^{o_2,d_2,k}_{o_1,d_1}$	binary, evaluates to 1 if truck k serves delivery d_2 of order o_2 immediately after having served delivery d_1 of order o_1

A.5. Decision Variables for MCNF (reduced version)

Symbol	Description
$wait^{c,t}_o$	number of trucks of class c waiting (idle) in time t at construction site of order o
$choose_{o,a}$	binary, evaluates to 1 if pattern a is chosen for order o
$stayHome_c$	number of trucks of class c that remain idle throughout the day
$moveP2O^{c,t}_o$	binary, number of trucks of class c that start being loaded at their home plant in t in order to go to construction site associated with order o immediately afterwards (first task per day)
$moveO2P^{c,t}_o$	number of trucks of class c that move back to their home plant in t after unloading at order o
$moveO2O^{c,t}_{o_1,o_2}$	binary, number of trucks of class c leaving order o_1 in time t and going to o_2 while being loaded at the closest plant en route

A.6. Decision Variables for MCNF (extensive version)

Symbol	Description
$waitAL^{c,t}_p$	number of trucks of class c waiting (full) at plant p in time t
$waitBL^{c,t}_p$	number of trucks of class c waiting (empty) at plant p in time t
$choose_{o,a}$	binary, evaluates to 1 if pattern a is chosen for order o
$stayHome_c$	number of trucks of class c that remain idle throughout the day

$moveP2O_{p,o}^{c,t}$	binary, number of trucks of class c that start being loaded at plant p in t in order to go to construction site associated with order o immediately afterwards
$moveO2P_{o,p}^{c,t}$	number of trucks of class c that move to plant p in t after unloading at order o
$load_{p}^{c,t}$	number of trucks of class c start being loaded at plant p in time t

A.7. Notation for Patterns

Symbol	**Description**
$start_{o,a}$	start of first unloading operation associated with pattern a of order o
$end_{o,a}$	end of last unloading operation associated with pattern a of order o
$first_{o,a}$	class of truck scheduled for first unloading operation by pattern a of order o
$delay_{o,a}$	gaps (before first delivery and between consecutive unloading operations) produced by pattern a of order o
$P_{o,a}^{c,t}$	binary indicator, equal to 1 if truck of class c is supposed to start unloading in t according to patter a of order o

B. Additional Results

B.1. Bounds for VRP*

Table B.1.: Best Bounds after $t_{max} = 150$ seconds

		Variant 1				Variant 2			
		cut_{max}							
class	n	100	500	1000	∞	100	500	1000	∞
mini	1	316.92	316.92	316.92	316.92	331.02	331.02	331.02	331.02
	2	754.43	754.43	754.43	754.43	754.43	754.43	754.43	754.43
	3	903.51	903.51	917.16	903.51	922.02	922.02	922.02	922.02
	4	1128.06	1128.06	1128.06	1128.06	1167.73	1167.73	1167.73	1167.73
	5	640.56	640.56	640.56	640.56	642.07	642.07	642.07	642.07
small	6	1289.92	1289.92	1289.92	1289.92	1289.92	1289.92	1289.92	1289.92
	7	1063.63	1063.63	1063.63	1063.63	1065.74	1065.74	1065.74	1065.74
	8	1411.08	1411.08	1411.08	1411.08	1416.04	1416.04	1416.04	1416.04
	9	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15
	10	1675.15	1675.15	1675.15	1675.15	1704.39	1704.39	1704.39	1704.39
medium	11	2137.28	2137.28	2137.28	2137.28	2137.28	2137.28	2137.28	2137.28
	12	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15
	13	1829.73	1829.73	1829.73	1829.73	1829.73	1829.73	1829.73	1829.73
	14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
large	16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table B.2.: Best Bounds after $t_{max} = 300$ seconds

		Variant 1				Variant 2			
		cut_{max}							
class	n	100	500	1000	∞	100	500	1000	∞
mini	1	331.02	316.92	316.92	316.92	331.02	331.02	331.02	331.02
	2	754.43	754.43	754.43	754.43	754.43	754.43	754.43	754.43
	3	903.51	903.51	923.69	903.51	922.02	922.02	922.02	922.02
	4	1128.06	1130.67	1128.06	1128.06	1169.11	1169.11	1169.11	1169.11
	5	642.15	640.64	641.19	640.56	642.07	642.07	642.07	642.07
small	6	1289.92	1289.92	1289.92	1289.92	1297.42	1297.42	1297.42	1297.42
	7	1063.63	1070.99	1070.99	1070.99	1078.35	1078.35	1078.35	1078.35
	8	1411.08	1411.08	1411.08	1411.08	1416.49	1416.49	1416.49	1416.49
	9	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15
	10	1675.15	1675.15	1675.15	1675.15	1704.39	1704.39	1704.39	1704.39
medium	11	2137.28	2137.28	2137.28	2137.28	2137.28	2137.28	2137.28	2137.28
	12	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15
	13	1829.73	1829.73	1829.73	1829.73	1831.16	1831.16	1831.16	1831.16
	14	1892.10	1892.10	1892.10	1892.10	1892.10	1892.10	1892.10	1892.10
	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
large	16	2017.20	2017.20	2017.20	2017.20	2017.20	2017.20	2017.20	2017.20
	17	2028.88	2028.88	2028.88	2028.88	2028.88	2028.88	2028.88	2028.88
	18	2080.05	2080.05	2080.05	2080.05	2080.05	2080.05	2080.05	2080.05
	19	0.00	2445.68	2445.68	2445.68	2445.68	2445.68	2445.68	2445.68
	20	2016.68	2016.68	2016.68	2016.68	2016.68	0.00	2016.68	2016.68

Table B.3.: Best Bounds after $t_{max} = 600$ seconds

		Variant 1				Variant 2			
		cut_{max}							
class	n	100	500	1000	∞	100	500	1000	∞
mini	1	331.02	316.92	316.92	316.92	331.02	331.02	331.02	331.02
	2	754.43	754.43	754.43	754.43	754.43	754.43	754.43	754.43
	3	903.51	903.51	923.69	903.51	922.02	922.02	922.02	922.02
	4	1128.06	1130.67	1128.06	1128.06	1170.34	1171.42	1171.42	1171.42
	5	642.15	640.64	641.19	640.56	642.07	642.07	642.07	642.07
small	6	1289.92	1289.92	1289.92	1289.92	1297.62	1297.62	1297.62	1297.62
	7	1070.99	1070.99	1070.99	1070.99	–	1079.65	1079.65	1079.65
	8	1411.08	1417.38	1429.72	1416.50	1422.28	1422.28	1422.28	1422.28
	9	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15	1414.15
	10	1675.15	1675.15	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39
medium	11	2137.28	2137.28	2137.28	–	2156.16	2156.16	2156.16	2156.16
	12	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15
	13	1829.73	1829.73	1829.73	1829.73	1831.16	1831.16	1831.16	1831.16
	14	1892.10	1892.10	1892.10	1892.10	1892.10	1892.10	1892.10	1892.10
	15	0.00	0.00	0.00	0.00	1749.35	0.00	1749.35	0.00
large	16	2017.20	2017.20	2017.20	2017.20	2024.91	2024.91	2024.91	2024.91
	17	2028.88	2028.88	2028.88	2028.88	2032.69	2032.69	2032.69	2032.69
	18	2080.05	2080.05	2080.05	2080.05	2080.05	2080.05	2080.05	2080.05
	19	2445.68	2445.68	2445.68	2445.68	2445.68	2445.68	2445.68	2445.68
	20	2016.68	2016.68	2016.68	2016.68	2016.68	2016.68	2016.68	2016.68

Table B.4.: Best Bounds after $t_{max} = 1200$ seconds

		Variant 1				Variant 2			
		cut_{max}							
class	n	100	500	1000	∞	100	500	1000	∞
mini	1	331.02	316.92	316.92	316.92	331.02	331.02	331.02	331.02
	2	754.43	754.43	754.43	754.43	754.43	754.43	754.43	754.43
	3	903.51	903.51	923.69	903.51	922.02	922.02	922.02	922.02
	4	1128.06	1130.67	1128.06	1128.06	1170.34	1171.42	1171.42	1171.42
	5	642.15	640.64	641.19	640.56	642.07	642.07	642.07	642.07
small	6	1289.92	1297.62	1297.62	1297.62	1303.14	1303.14	1303.14	1303.14
	7	1070.99	1070.99	1070.99	1070.99	–	–	–	–
	8	1416.50	1430.60	1429.72	1417.45	1422.28	1422.28	1422.28	1422.28
	9	1414.15	1414.15	1414.15	1414.15	1414.71	1414.71	1414.71	1414.71
	10	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39
medium	11	2137.28	2137.28	2137.28	–	2156.50	2156.50	2156.50	2156.50
	12	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15
	13	1829.73	1831.16	1831.27	1831.16	1839.73	1839.73	1839.73	1839.73
	14	1892.10	1892.10	1892.10	–	1920.86	1920.81	1920.86	1920.81
	15	1749.35	1749.35	1749.35	1749.35	1762.78	1762.78	1762.78	1762.78
large	16	2017.20	2017.20	2017.20	–	2025.04	2025.04	2025.04	2025.04
	17	2028.88	2028.88	2028.88	2028.88	2032.69	2032.69	2032.69	2032.69
	18	2080.05	2080.05	2080.05	2080.05	2093.61	2093.61	2093.61	2093.61
	19	2445.68	2445.68	2445.68	2445.68	2450.60	2450.60	2450.60	2450.60
	20	2016.68	2016.68	2016.68	–	2023.06	2023.06	2023.06	2023.06

Table B.5.: Best Bounds after $t_{max} = 2400$ seconds

		Variant 1				Variant 2			
		cut_{max}							
class	n	100	500	1000	∞	100	500	1000	∞
mini	1	331.02	331.02	316.92	316.92	331.02	331.02	331.02	331.02
	2	754.43	754.82	754.43	754.43	754.50	754.43	754.43	754.43
	3	903.51	903.51	923.69	903.51	922.02	922.02	922.02	922.02
	4	1128.06	1130.67	1128.06	1128.06	1170.34	1171.42	1171.42	1171.42
	5	–	640.64	641.19	–	642.07	642.07	642.07	642.07
small	6	1298.32	1297.89	1298.24	1297.62	1303.16	1303.16	1303.16	1303.16
	7	1070.99	1070.99	1070.99	1070.99	–	–	–	–
	8	1416.50	1430.60	1429.72	1430.79	1422.29	1422.29	1422.29	1422.29
	9	1414.15	1414.15	1414.15	1414.15	1414.73	1414.73	1414.73	1414.73
	10	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39	1704.39
medium	11	2137.28	2137.28	2137.28	–	2157.27	2157.27	2157.27	2157.27
	12	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15	1813.15
	13	1839.73	1831.16	1839.73	1839.73	1839.73	1839.73	1839.73	1839.73
	14	1892.10	1892.10	1892.10	–	1921.21	1921.21	1921.21	1921.21
	15	1749.35	1749.35	1749.35	1749.35	1762.92	1762.92	1762.92	1762.92
large	16	2017.20	2017.20	2017.20	–	2025.04	2025.04	2025.04	2025.04
	17	2028.88	2028.88	2032.69	2032.69	2032.69	2032.69	2032.69	2032.69
	18	2080.05	2080.05	2080.05	2080.05	2093.74	2093.74	2093.74	2093.74
	19	2445.68	2445.68	2445.68	–	2450.63	2450.63	2450.63	2450.63
	20	2016.68	2016.68	2016.68	–	2023.06	2023.06	2023.06	2023.06

B.2. Integrative Hybrid Approach

Table B.6.: Integrative Hybrid Approach ($t_{max} = 150$)

		t_{init}										
class	n	60	120	150	300	600	900	1200	1500	1800	2400	4800
mini	1	331.90	335.47									
	2	762.00	762.00									
	3	1171.75	1164.73									
	4	1197.58	1191.70									
	5	700.72	697.60									
small	6	1894.28	1839.27									
	7	1251.54	1216.38									
	8	1631.71	1591.79									
	9	1786.94	1718.54									
	10	2405.84	2288.94									
medium	11	4279.80	3581.56									
	12	2424.76	2206.67									
	13	2463.69	2275.26									
	14	4074.38	3811.67									
	15	3289.31	2841.12									
large	16	3821.47	3227.00									
	17	3004.15	2734.33									
	18	4191.80	3728.92									
	19	–	–									
	20	4287.11	3674.86									

Table B.7.: Integrative Hybrid Approach ($t_{max} = 300$)

		t_{init}										
class	n	60	120	150	300	600	900	1200	1500	1800	2400	4800
mini	1	331.90	331.90	335.47								
	2	762.00	762.00	762.00								
	3	1162.48	1157.38	1152.05								
	4	1197.58	1190.88	1190.88								
	5	698.42	697.60	695.08								
small	6	1881.50	1825.74	1798.62								
	7	1250.38	1179.90	1184.12								
	8	1598.80	1568.63	1559.35								
	9	1752.57	1696.68	1688.96								
	10	2329.71	2238.79	2217.05								
medium	11	4228.69	3544.17	3353.57								
	12	2358.29	2186.34	2162.54								
	13	2341.21	2221.57	2215.27								
	14	4062.45	3752.92	3527.57								
	15	3231.32	2821.78	2687.63								
large	16	3766.72	3178.24	2997.30								
	17	2969.44	2700.67	2649.78								
	18	4181.88	3707.61	3539.79								
	19	–	–	–								
	20	5021.12	3701.22	3391.86								

Table B.8.: Integrative Hybrid Approach ($t_{max} = 600$)

		t_{init}										
class	n	60	120	150	300	600	900	1200	1500	1800	2400	4800
mini	1	331.90	331.90	331.90	331.90							
	2	762.00	762.00	762.00	762.00							
	3	1150.49	1110.80	1116.53	1110.59							
	4	1197.58	1190.49	1185.88	1190.49							
	5	696.72	697.60	691.25	691.08							
small	6	1874.55	1816.44	1776.32	1694.00							
	7	1242.11	1174.52	1181.74	1176.67							
	8	1578.36	1561.36	1545.48	1557.68							
	9	1724.32	1689.64	1665.59	1640.87							
	10	2268.85	2185.84	2175.64	2128.17							
medium	11	4219.68	3486.47	3349.71	3023.36							
	12	2245.96	2135.07	2148.28	2106.55							
	13	2226.56	2156.43	2163.33	2101.89							
	14	4021.49	3736.13	3513.46	3358.04							
	15	3161.56	2746.86	2639.16	2407.65							
large	16	3701.50	3101.75	2934.02	2742.30							
	17	2899.63	2641.67	2614.42	2531.99							
	18	4164.14	3670.92	3516.67	3062.59							
	19	–	–	–	–							
	20	6033.30	4848.97	3645.90	3334.43							

Table B.9.: Integrative Hybrid Approach ($t_{max} = 1200$)

		t_{init}										
class	n	60	120	150	300	600	900	1200	1500	1800	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90					
	2	762.00	762.00	762.00	762.00	762.00	762.00					
	3	1138.82	1099.04	1098.11	1087.28	1071.49	1062.99					
	4	1197.19	1189.67	1185.88	1190.49	1190.49	1190.49					
	5	694.76	696.46	691.08	686.47	682.14	681.96					
small	6	1858.35	1814.44	1769.45	1690.41	1592.11	1562.65					
	7	1236.31	1170.91	1180.73	1173.96	1156.82	1155.25					
	8	1570.12	1553.92	1539.43	1548.48	1544.19	1545.92					
	9	1706.07	1668.13	1633.09	1615.83	1599.56	1604.68					
	10	2220.69	2161.41	2139.71	2112.97	2102.48	2107.90					
medium	11	4219.68	3466.86	3349.71	3018.54	2613.52	2585.93					
	12	2213.96	2038.98	2130.81	2103.72	2051.51	2032.19					
	13	2192.04	2131.11	2121.27	2081.78	2043.44	2042.24					
	14	3992.90	3731.96	3513.46	3348.90	2895.41	2839.46					
	15	3110.70	2732.53	2590.51	2373.21	2248.45	2205.14					
large	16	3841.70	3149.58	2921.82	2726.80	2509.83	2444.27					
	17	2871.08	2620.69	2608.53	2496.77	2370.24	2359.08					
	18	4106.38	3664.02	3493.58	3050.96	2735.65	2650.42					
	19	–	–	–	–	–	–					
	20	6033.30	4848.97	3645.90	3555.21	2640.32	2583.71					

Table B.10.: Integrative Hybrid Approach ($t_{max} = 2400$)

		t_{init}										
class	n	60	120	150	300	600	900	1200	1500	1800	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90	331.90	331.90	331.90		
	2	762.00	762.00	762.00	762.00	762.00	762.00	762.00	762.00	762.00		
	3	1116.66	1099.04	1069.56	1075.97	1061.66	1044.90	1059.56	1042.75	1042.08		
	4	1192.19	1189.67	1185.49	1190.49	1189.67	1189.67	1185.49	1185.49	1185.49		
	5	694.76	693.95	687.17	686.47	681.79	680.90	681.27	676.76	676.76		
small	6	1834.01	1811.34	1765.83	1681.60	1567.23	1559.16	1538.27	1516.62	1505.04		
	7	1218.94	1167.82	1172.47	1170.94	1156.73	1155.25	1156.73	1156.56	1153.59		
	8	1557.17	1553.66	1536.75	1546.95	1541.57	1545.64	1544.17	1545.30	1547.35		
	9	1690.88	1661.16	1601.71	1615.02	1587.44	1590.84	1587.99	1593.43	1597.85		
	10	2196.48	2145.52	2114.41	2108.64	2078.83	2072.46	2097.76	2059.34	2059.95		
medium	11	4219.68	3465.39	3349.71	2996.55	2608.71	2576.65	2551.84	2506.82	2494.88		
	12	2192.77	2022.67	2128.45	2098.12	2033.02	2013.45	2009.84	2019.08	2013.75		
	13	2171.09	2113.07	2106.51	2073.27	2038.41	2024.22	2021.76	2023.03	2015.38		
	14	3991.67	3725.19	3513.46	3347.06	2892.91	2838.29	2777.94	2646.94	2610.90		
	15	3109.62	2732.53	2585.18	2354.38	2246.30	2184.54	2178.16	2115.64	2121.93		
large	16	3802.59	3103.19	2889.88	2885.28	2464.54	2413.11	2384.89	2345.05	2353.02		
	17	2864.38	2563.43	2557.83	2475.99	2361.02	2329.51	2329.63	2293.47	2281.25		
	18	4073.55	3664.02	3462.90	3032.72	2716.06	2641.19	2578.93	2577.57	2560.79		
	19	–	–	–	–	–	–	–	–	–		
	20	6033.30	4848.97	3645.90	3555.21	2640.32	–	2382.03	–	–		

Table B.11.: Integrative Hybrid Approach ($t_{max} = 4800$)

		t_{init}										
class	n	60	120	150	300	600	900	1200	1500	1800	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90	331.90	331.90	331.90	331.90	331.90
	2	762.00	762.00	762.00	762.00	762.00	762.00	762.00	762.00	762.00	762.00	762.00
	3	1114.16	1083.58	1054.33	1072.11	1053.06	1030.71	1038.56	1041.61	1019.89	1034.46	1037.09
	4	1186.37	1189.67	1184.67	1189.67	1184.67	1189.67	1185.49	1184.67	1185.49	1185.49	1184.67
	5	692.75	693.07	686.99	685.60	681.09	680.26	681.27	676.76	676.06	676.76	675.06
small	6	1834.01	1806.18	1756.30	1681.60	1564.67	1549.50	1522.89	1507.94	1503.69	1500.39	1492.30
	7	1216.77	1162.34	1162.31	1170.94	1149.93	1155.22	1156.27	1156.15	1148.92	1147.51	1142.28
	8	1540.34	1550.75	1535.50	1546.95	1539.01	1540.56	1542.12	1536.02	1541.99	1541.76	1544.17
	9	1687.92	1654.67	1599.11	1607.64	1570.56	1568.53	1586.46	1585.24	1596.95	1581.15	1560.46
	10	2159.72	2121.08	2078.92	2108.20	2062.99	2071.07	2083.81	2056.36	2039.22	2024.79	2019.48
medium	11	4219.68	3452.58	3349.71	2996.55	2595.74	2576.65	2545.46	2495.11	2493.90	2467.09	2465.65
	12	2191.85	2014.19	2094.11	2098.12	2011.71	1994.07	2001.48	1999.29	2008.85	1988.17	1981.16
	13	2126.19	2072.01	2074.27	2059.25	2037.51	2015.02	2015.20	2021.88	2013.96	2015.84	2009.60
	14	3962.31	3725.19	3513.46	3347.06	2884.09	2832.11	2777.94	2646.09	2610.40	2582.00	2514.39
	15	3061.62	2732.53	2534.58	2315.58	2240.34	2179.56	2136.37	2089.72	2099.15	2092.75	2095.57
large	16	3946.87	3103.19	3008.82	2885.28	2508.52	2392.26	2384.89	2338.96	2307.73	2336.68	2294.50
	17	2767.97	2533.12	2536.84	2435.93	2342.26	2312.45	2299.36	2287.58	2268.45	2262.07	2227.11
	18	3801.78	3664.02	3414.58	3032.72	2677.79	2641.19	2588.16	2577.57	2551.34	2518.80	2477.30
	19	–	–	–	–	–	–	–	–	–	–	–
	20	6033.30	4848.97	3645.90	3555.21	2640.32	–	–	–	–	2266.32	2270.82

B.3. Brute Force Patterns for MCNF

Table B.12.: Best solutions found using MCNF generating p_{init} brute force patterns

		p_{init}										
class	n	50	100	200	300	400	500	600	700	800	900	1000
mini	1	367.30	352.77	333.13	333.13	333.13	333.13	333.13	333.13	333.13	333.13	333.13
	2	928.08	811.62	769.53	769.53	764.55	764.55	764.55	762.00	762.00	762.00	762.00
	3	1546.52	1450.65	1243.77	1239.68	1166.12	1159.48	1159.48	1159.48	1159.48	1159.48	1159.48
	4	1850.83	1619.82	1436.48	1359.07	1304.43	1281.73	1281.73	1281.73	1279.48	1270.95	1270.95
	5	1050.95	929.23	834.52	813.80	757.20	756.15	743.32	743.32	743.32	743.32	743.32
small	6	7084.05	2982.80	2297.13	2088.30	2010.80	1995.25	1995.25	1991.57	1976.62	1976.62	1962.55
	7	1596.80	1407.12	1243.03	1238.72	1225.35	1210.20	1206.65	1205.57	1205.57	1196.62	1196.62
	8	2551.23	2108.20	1952.98	1856.32	1802.72	1793.82	1793.82	1793.57	1783.25	1782.17	1768.00
	9	2746.90	2284.02	2009.13	1988.42	1974.98	1952.58	1931.62	1863.90	1863.90	1852.80	1852.18
	10	3405.92	2813.27	2733.05	2723.20	2692.08	2640.13	2608.93	2608.18	2566.40	2460.57	2534.43
medium	11	8381.37	3684.87	3162.70	3425.08							
	12	3022.97	2572.33	2355.42	2233.73							
	13	2964.65	2501.37	2348.83	2335.82							
	14	13126.60	4192.85	3527.02	3407.93							
	15	8408.03	3249.75	2843.18	2689.25							
large	16	8387.47	3401.10	3055.58								
	17	8348.70	3024.80	2703.47								
	18	8666.98	3318.97	3014.60								
	19	5006.97	3966.80	3695.02								
	20	3823.08	3110.68	2782.55								

Table B.13.: Average solutions found using MCNF generating p_{init} brute force patterns

		p_{init}										
class	n	50	100	200	300	400	500	600	700	800	900	1000
mini	1	461.18	383.50	345.99	337.93	336.98	333.59	333.31	333.13	333.13	333.13	333.13
	2	1005.59	853.17	782.50	774.16	770.35	768.36	767.36	766.85	766.85	764.04	764.04
	3	1594.44	1495.36	1352.98	1335.16	1269.97	1252.57	1227.86	1224.45	1219.07	1218.12	1216.82
	4	1875.05	1659.10	1486.39	1399.29	1363.27	1342.67	1320.12	1314.11	1304.49	1298.50	1287.11
	5	1155.83	988.98	888.29	856.06	798.12	788.04	780.15	777.57	773.24	771.87	770.09
small	6	7295.48	6151.30	2346.93	2192.48	2163.12	2148.50	2110.46	2065.95	2033.24	2023.92	2006.49
	7	1707.88	1451.65	1320.50	1263.08	1237.91	1229.68	1221.10	1217.78	1215.98	1206.43	1204.72
	8	3470.82	2178.06	1991.37	1903.67	1867.28	1854.65	1844.89	1834.08	1824.05	1820.43	1796.22
	9	3820.16	2427.55	2240.99	2154.32	2081.14	2051.82	2012.20	1944.49	1939.76	1923.78	1918.62
	10	5803.57	3047.74	2785.79	2747.37	2717.86	2683.79	2653.55	2633.11	2619.85	2559.54	2559.33
medium	11	9401.47	3873.28	3390.79	3425.08							
	12	3166.76	2645.08	2397.36	2233.73							
	13	3049.38	2643.80	2411.31	2335.82							
	14	14246.68	7514.34	3639.64	3407.93							
	15	11793.87	5901.09	2893.83	2689.25							
large	16	9464.01	3520.92	3111.70								
	17	9476.18	3128.63	2781.07								
	18	9887.88	3535.87	3078.77								
	19	5092.20	4133.45	3781.13								
	20	4711.64	3226.02	2941.22								

Table B.14.: Average runtimes using MCNF generating p_{init} brute force patterns

		p_{init}										
class	n	50	100	200	300	400	500	600	700	800	900	1000
mini	1	1.3	1.4	1.8	2.1	2.2	2.5	2.7	3.1	3.3	3.7	3.8
	2	1.6	1.7	2.2	2.5	2.7	2.9	3.1	3.4	3.8	4.0	4.5
	3	2.9	4.8	6.8	11.4	10.6	14.9	16.2	19.4	23.6	28.4	35.0
	4	3.6	4.8	6.1	6.1	6.7	6.8	11.3	12.5	12.0	15.0	18.1
	5	2.9	3.5	4.5	4.9	5.2	6.2	6.6	8.1	8.5	9.0	11.1
small	6	7.8	74.2	483.0	1921.2	7200.3	19185.0	25341.8	32752.4	47531.2	86220.8	64091.1
	7	5.2	7.2	9.1	13.3	11.6	17.1	21.2	23.1	28.9	28.0	26.9
	8	8.4	12.4	25.4	24.3	38.0	86.2	104.5	101.6	103.5	288.4	115.2
	9	10.5	23.1	119.7	155.9	180.9	284.1	462.1	165.9	212.6	453.5	810.5
	10	11.0	30.2	260.7	1055.7	4810.0	7633.6	35475.3	41389.9	92973.0	69090.8	103187.0
medium	11	29.9	371.9	5964.1	36608.8							
	12	21.2	33.0	106.1	28.1							
	13	17.2	33.3	38.8	53.3							
	14	81.3	3359.3	41303.4	200330.0							
	15	44.8	854.1	11267.7	28066.3							
large	16	130.6	1828.8	16006.2								
	17	37.7	259.2	2597.4								
	18	152.2	557.4	2516.2								
	19	107.4	5741.0	99823.4								
	20	71.4	352.4	1515.5								

B.4. Intelligent Base Patterns for MCNF

Table B.15.: Best and Average Solutions found using compatible base patterns for MCNF

		z_{avg}				z_{min}			
		p_{init}							
class	n	5	10	15	20	5	10	15	20
mini	1	338.78	334.02	334.02	333.13	353.16	334.71	334.19	333.41
	2	945.82	827.65	794.23	771.48	1016.94	890.65	843.19	819.04
	3	1448.75	1363.10	1283.45	1222.55	1543.49	1409.89	1332.08	1297.46
	4	1617.27	1482.32	1420.20	1327.47	1817.33	1556.76	1493.29	1403.63
	5	1076.58	906.35	856.77	796.15	1114.29	961.52	893.26	838.80
small	6	2425.33	2342.55	2191.92	2168.05	2832.13	2541.29	2327.57	2236.54
	7	1530.80	1388.32	1301.95	1244.38	1575.77	1404.75	1329.10	1276.04
	8	2281.30	1996.72	1923.43	1922.65	2438.96	2118.73	2025.37	1947.91
	9	2734.27	2108.25	2085.53	2055.27	2962.82	2373.83	2190.05	2131.55
	10	2968.23	2602.97	2553.97	2480.60	3221.10	2762.44	2619.65	2519.20
medium	11	3835.87	3513.17	3260.20	3136.22	4222.74	3542.27	3292.25	3173.99
	12	3249.38	2723.52	2427.20	2376.78	3363.17	2854.08	2588.08	2506.69
	13	3108.75	2684.90	2473.07	2437.12	3274.88	2760.34	2555.52	2478.77
	14	5011.77	3677.72	3303.03	3237.23	5346.76	3901.00	3446.81	3287.30
	15	4268.37	3244.62	2920.75	2739.30	4762.58	3367.29	2973.71	2834.30
large	16	3794.52	3128.00	2881.88	2812.85	4465.41	3406.65	3083.81	2944.49
	17	4191.22	3216.30	2867.75	2652.65	4577.36	3330.29	2942.87	2767.51
	18	4611.90	3659.85	3253.72	3179.18	5006.35	3796.61	3396.58	3260.93
	19	4969.63	3982.92	3624.58	3512.55	5166.71	4068.27	3771.53	3598.98
	20	4551.03	3332.37	3123.72	2956.88	5043.90	3429.80	3137.03	2992.23

Table B.16.: Average total run times and time for generating p_{init} compatible base patterns

		t_{avg}				t_{avg}^{init}			
		p_{init}							
class	n	5	10	15	20	5	10	15	20
mini	1	2.48	2.61	2.71	3.13	0.01	0.02	0.03	0.06
	2	2.60	2.85	2.98	3.09	0.02	0.03	0.05	0.06
	3	3.15	5.35	6.25	7.88	0.03	0.07	0.12	0.16
	4	3.90	4.97	5.77	6.12	0.03	0.06	0.10	0.14
	5	3.21	3.87	4.21	4.80	0.02	0.05	0.08	0.13
small	6	8.00	17.38	41.70	118.29	0.09	0.18	0.28	0.40
	7	5.16	8.03	10.69	11.89	0.05	0.10	0.17	0.23
	8	7.83	12.61	19.34	22.06	0.07	0.15	0.24	0.33
	9	11.06	30.14	50.59	104.31	0.09	0.19	0.31	0.42
	10	12.44	21.97	49.15	59.97	0.12	0.24	0.38	0.52
medium	11	24.92	111.26	382.20	1320.30	0.20	0.42	0.64	0.88
	12	17.45	45.51	81.03	279.25	0.17	0.34	0.52	0.71
	13	22.55	56.07	58.84	135.90	0.19	0.39	0.60	0.83
	14	37.27	482.57	2393.55	8064.79	0.27	0.56	0.85	1.13
	15	42.79	201.28	510.24	2003.82	0.28	0.57	0.88	1.21
large	16	37.65	241.08	600.34	1802.65	0.34	0.72	1.11	1.52
	17	59.09	239.34	624.55	1043.74	0.28	0.53	0.82	1.11
	18	56.94	347.84	1644.19	3508.40	0.34	0.72	1.08	1.48
	19	83.06	778.10	5792.82	24195.52	0.34	0.70	1.06	1.45
	20	46.39	256.72	813.22	1715.96	0.43	0.89	1.35	1.80

B.5. VNS

Table B.17.: Best Solutions (z_{min}) found using VNS after t_{max} seconds

		t_{max}					
class	n	150	300	600	1200	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90
	2	762.00	762.00	762.00	762.00	762.00	762.00
	3	1090.90	1090.90	1090.90	1090.90	1090.90	1089.68
	4	1184.95	1182.97	1182.97	1182.97	1182.97	1182.97
	5	675.45	675.45	675.45	675.45	671.97	671.97
small	6	1618.38	1590.83	1589.95	1568.72	1568.72	1568.72
	7	1161.83	1161.83	1161.83	1161.83	1148.02	1148.02
	8	1534.83	1534.13	1534.13	1534.13	1534.13	1534.13
	9	1680.62	1556.23	1547.97	1546.58	1535.02	1535.02
	10	2404.03	2081.63	2057.55	2057.55	2055.73	2055.73
medium	11	3629.38	3185.18	2628.63	2610.77	2590.03	2566.92
	12	2585.60	2127.08	2056.60	1991.57	1921.03	1921.03
	13	2785.35	2040.92	2040.92	2039.75	2036.05	2030.32
	14	4010.82	3705.90	3201.90	2738.12	2691.32	2592.85
	15	3694.93	3320.70	2450.13	2297.12	2175.95	2175.70
large	16	3861.93	3367.17	3052.72	2499.27	2448.42	2441.40
	17	3706.55	3224.45	2736.68	2374.28	2373.20	2346.05
	18	3946.97	3483.98	3038.58	2586.58	2533.83	2533.83
	19	4494.88	4206.72	3854.63	3331.78	3112.65	3073.52
	20	4167.85	3360.42	3227.52	2666.80	2316.08	2279.45

Table B.18.: Average Solutions (z_{avg}) found using VNS after t_{max} seconds

		t_{max}					
class	n	150	300	600	1200	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90
	2	763.02	762.51	762.51	762.51	762.51	762.51
	3	1139.23	1134.98	1132.15	1127.53	1121.20	1112.31
	4	1193.84	1192.63	1190.92	1188.52	1186.11	1186.11
	5	700.87	700.87	699.56	697.99	697.29	697.29
small	6	1769.89	1673.09	1642.54	1603.94	1600.27	1585.63
	7	1230.36	1228.30	1226.03	1211.12	1189.80	1189.58
	8	1606.13	1600.29	1588.07	1587.66	1575.62	1571.76
	9	1738.37	1689.96	1627.58	1601.64	1583.90	1583.53
	10	2639.71	2226.19	2138.81	2120.57	2109.56	2096.12
medium	11	4012.65	3502.68	2836.28	2748.02	2679.89	2667.59
	12	2869.20	2436.70	2177.31	2055.58	2016.84	2007.63
	13	2962.88	2191.52	2104.50	2077.61	2050.92	2047.20
	14	4814.34	4256.65	3814.48	3082.71	2860.66	2784.32
	15	4177.63	3734.78	3232.86	2426.04	2240.03	2221.53
large	16	4634.89	3986.32	3353.09	2706.43	2519.82	2496.31
	17	4105.21	3492.57	3100.87	2474.92	2418.08	2380.84
	18	4456.51	3816.04	3442.29	2731.30	2628.57	2562.18
	19	4938.90	4366.80	3978.70	3483.62	3257.61	3214.05
	20	4374.43	3804.66	3481.08	2986.49	2511.37	2443.95

B.6. Cooperative Hybrid Approach

Table B.19.: Best Solutions (z_{min}) found using Cooperative Hybrid after t_{max} seconds

		t_{max}					
class	n	150	300	600	1200	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90
	2	762.00	762.00	762.00	762.00	762.00	762.00
	3	1053.12	1021.58	1026.07	1021.58	1011.70	1021.08
	4	1182.97	1182.97	1182.97	1182.97	1182.97	1182.97
	5	670.42	670.42	670.42	666.93	666.93	666.93
small	6	1599.85	1581.68	1510.13	1529.15	1510.10	1524.52
	7	1139.53	1118.00	1131.77	1118.00	1127.08	1120.32
	8	1547.82	1541.45	1510.75	1532.22	1508.13	1527.23
	9	1575.98	1547.87	1543.00	1545.35	1532.32	1529.35
	10	2122.33	2086.23	2097.42	2075.57	2000.05	2031.20
medium	11	2698.88	2632.18	2601.60	2512.98	2440.63	2395.48
	12	2052.37	1953.87	1926.78	1951.82	1939.23	1884.97
	13	2100.62	2025.88	2022.67	2006.85	2007.53	1993.88
	14	3333.25	2784.83	2634.17	2432.48	2438.58	2457.80
	15	2565.32	2216.43	2156.77	2133.28	2127.77	2091.17
large	16	2673.67	2616.25	2277.22	2249.10	2323.70	2300.85
	17	2444.65	2312.23	2215.73	2166.58	2163.87	2167.77
	18	3123.72	2849.75	2523.17	2405.98	2358.03	2367.25
	19	3499.70	3347.53	2939.25	2890.43	2917.38	2853.68
	20	2868.47	2536.55	2371.52	2284.60	2277.48	2189.02

Table B.20.: Average Solutions (z_{avg}) found using Cooperative Hybrid after t_{max} seconds

		t_{max}					
class	n	150	300	600	1200	2400	4800
mini	1	331.90	331.90	331.90	331.90	331.90	331.90
	2	762.51	762.51	762.51	762.00	762.00	762.00
	3	1104.48	1062.30	1073.17	1059.42	1040.01	1036.69
	4	1186.61	1185.39	1185.39	1185.39	1185.39	1184.60
	5	688.38	689.31	688.79	679.56	683.76	678.10
small	6	1732.60	1639.95	1585.68	1573.40	1555.26	1573.61
	7	1157.88	1140.30	1144.55	1135.49	1138.48	1127.76
	8	1578.51	1585.43	1550.60	1565.74	1548.76	1544.42
	9	1654.95	1615.70	1616.30	1588.14	1544.49	1553.17
	10	2250.26	2155.51	2129.38	2139.45	2063.62	2091.77
medium	11	2909.58	2748.89	2688.99	2614.89	2512.02	2532.27
	12	2193.88	2071.51	2034.41	1997.36	2009.64	1971.01
	13	2178.18	2059.70	2050.10	2039.02	2045.22	2011.33
	14	3641.25	3025.02	2834.58	2588.24	2539.27	2580.00
	15	2726.87	2381.39	2214.86	2208.29	2197.62	2130.80
large	16	2984.90	2728.93	2440.06	2379.48	2379.70	2380.36
	17	2692.13	2409.69	2323.32	2223.85	2235.77	2224.42
	18	3410.02	2958.78	2705.24	2523.61	2532.91	2502.38
	19	3734.63	3463.85	3094.95	3005.84	2960.84	2924.99
	20	3157.13	2650.09	2496.88	2369.63	2357.72	2275.55

C. Acknowledgment

We would like to thank Hans Karl Huber and Daniela Feichter from Rienz Beton for providing us access to real-world data. This project was supported, in part, by FWF grant P20342-NI3. Academic Licenses of XPRESS-MP (v. 2006B) by Dash Optimization as well as CPLEX (v. 10.1) were used for the test runs.

The author would like to thank Martin W.P Savelsbergh and Juan José Salazar-González for their valuable collaboration and useful insight into MCNF and VRP* respectively.

Bibliography

Ahuja, R.K. and Magnanti, T.L. and Orlin, J.B. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

Archetti, C. and Speranza, M.G. An Overview on the Split Delivery Vehicle Routing Problem. In *Operations Research Proceedings 2006*, pages 123–127, 2007.

Archetti, C. and Mansini, R. and Speranza, M.G. Complexity and Reducibility of the Skip Delivery Problem. *Transportation Science*, 39(2):182–187, 2005.

Archetti, C. and Savelsbergh, M.W.P. and Speranza, M.G. Worst-Case Analysis for Split Delivery Vehicle Routing Problems. *Transportation Science*, 40(2):226–234, 2006.

Archetti, C. and Hertz, A. and Speranza, M.G. Metaheuristics for the Team Orienteering Problem. *Journal of Heuristics*, 13(1):49–76, 2007.

Arunapuram, S. and Mathur, M. and Solow, D. Vehicle Routing and Scheduling with Full Truckloads. *Transportation Science*, 37(2):170–182, 2003.

Ball, M.O. and Golden, B.L. and Assad, A.A. and Bodin, L.D. Planning for Truck Fleet Size in the Presence of a Common-Carrier Option. *Decision Sciences*, 14(1), 1983.

Belenguer, J.M. and Martinez, M.C. and Mota, E. A Lower Bound for the Split Delivery Vehicle Routing Problem. *Operations Research*, 48(5):801–810, 2000.

Bellmore, M. and Nemhauser, G.L. The Traveling Salesman Problem: A Survey. *Operations Research*, 16(3):538–558, 1968.

Blazewicz, J. and Pesch, E. and Sterna, M. and Werner, F. Metaheuristic Approaches for the Two-Machine Flow-Shop Problem with Weighted Late Work Criterion and Common Due Date. *Computers and Operations Research*, 35(2):574–599, 2008.

Blum, C. Beam-ACO: Hybridizing Ant Colony Optimization with Beam Search: An Application to Open Shop Scheduling. *Computers & Operations Research*, 32:1565–1591, 2005.

Bodin, L. and Golden, B. and Assad, A. and Ball, M. Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers & Operations Research*, 10(2):63–211, 1983.

Bräysy, O. A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows. *INFORMS Journal of Computing*, 15(4):347 – 368, 2003.

Bräysy, O. and Gendreau, M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39:104 – 118, 2005a.

Bräysy, O. and Gendreau, M. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39:119 – 139, 2005b.

Burke, E.K. and Curtois, T. and Post, G. and Qu, R. and Veltman, B. A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem. *European Journal of Operational Research*, 2007. to appear.

Chao, I.M. and Golden, B.L. and Wasil, E.A. A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves upon Best Known Solutions. *American Journal of Mathematical and Management Sciences*, 13:371–406, 1993.

Cordeau, J.F. and Gendreau, M. and Laporte, G. A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. *Networks*, 30:105–119, 1997.

Cordeau, J.F. and Laporte, G. and Mercier, A. A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

Desrosiers, J. and Laporte, G. and Sauve, M. and Soumis, F. and Taillefer, S. Vehicle Routing with Full Loads. *Computers & Operations Research*, 15:219–226, 1988.

Dror, M. and Trudeau, P. Savings by Split Delivery Routing. *Transportation Science*, 23: 141–145, 1989.

Dror, M. and Trudeau, P. Split Delivery Routing. *Naval Research Logistics*, 37:383–402, 1990.

Dror, M. and Laporte, G. and Trudeau, P. Vehicle Routing with Split Deliveries. *Discrete Applied Mathematics*, 50:239–254, 1994.

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

ERMCO. *Ready mixed concrete (A natural choice)*. European Ready Mixed Concrete Organization, 2000.

ERMCO. *European ready-mixed Concrete Industry Statistics*. European Ready Mixed Concrete Organization, 2004.

ERMCO. *European ready-mixed Concrete Industry Statistics*. European Ready Mixed Concrete Organization, 2005.

Feillet, D. and Dejax, P. and Gendreau, M. and Gueguen, C. Vehicle Routing with Time Windows and Split Deliveries. Working paper, 2002.

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

Gendreau, M. and Laporte, G. and Potvin, J.Y. , *Local Search in Combinatorial Optimization*. John Wiley & Sons, New York (NY), USA, 1997.

Handbook of Metaheuristics. International Series in Operations Research & Management Science. Kluwer Academic Publishers, Norwell (MA), USA, 2003.

Glover, F.W. and Klingman, D. and Phillips, N.V. *Network Models in Optimization and their Application in Practice*. John Wiley & Sons, New York (NY), USA, 2003.

Gomory, R.E. Outline of an Algorithm for Integer Solutions to Linear Programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

Gomory, R.E. Solving Linear Programming Problems in Integers. In *Combinatorial Analysis*, Proceedings of Symposia in Appied Mathematics, 1960.

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

Hansen, P. and Mladenović, N. and Urošević, D. Variable Neighborhood Search and Local Branching. *Computers & Operations Research*, 33(10):3034–3045, 2006.

Hansen, P. and Oğuz, C. and Mladenović, N. Variable Neighborhood Search for Minimum Cost Berth Allocation. *European Journal of Operational Research*, 2007. to appear.

Hemmelmayr, V.C. and Doerner, K.F. and Hartl, R.F. A Variable Neighborhood Search Heuristic for Periodic Routing Problems. *European Journal of Operational Research*, 2007. to appear.

Ho, S.C. and Haugland, D. A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows and Split Deliveries. *Computers & Operations Research*, 31(12): 1947–1964, 2004.

Hoffman, K. and Durbin, M. The Dance of the Thirty Ton Trucks. *Operations Research*, 2007. to appear.

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

Kindervater, G.A.P. and Savelsbergh, M.W.P. , *Local Search in Combinatorial Optimization*. John Wiley & Sons, Chichester, USA, 1997.

Kirkpatrick, S. and Gelatt, C.D. and Vecchi, M.P. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

Kytöjoki, J. and Nuortio, T. and Bräysy, O. and Gendreau, M. An Efficient Variable Neighborhood Search Heuristic for Very Large Scale Vehicle Routing Problems. *Computers and Operations Research*, 34(9):2743–2757, 2007.

Marchand, H. and Martin, A. and Weismanteil, R. and Wolsey, L. Cutting Planes in Integer and Mixed Integer Programming. *Discrete Applied Mathematics*, 123:397–446, 2002.

Matsatsinis, N.F. 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.

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

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

Naso, D. and Surico, M. and Turchiano, B. and Kaymak, U. Genetic Algorithms for Supply-Chain Scheduling: A Case Study in the Distribution of Ready-Mixed Concrete. *European Journal of Operational Research*, 177(3):2069–2099, 2007.

Pitakaso, R. and Almeder, C and Doerner, K.F. and Hartl, R.F. Combining Population-Based and Exact Methods for Mmulti-Level Capacitated Lot-Sizing Problems. *International Journal of Production Research*, 44(22):4755–4771, 2006.

Pitakaso, R. and Almeder, C. and Doerner, K.F. and Hartl, R.F. A MAX-MIN Ant System for Unconstrained Multi-Level Lot-Sizing problems. *Computers & Operations Research*, 34:2533–2552, 2007.

Polacek, M. and Hartl, R.F. and Doerner, K.F. and Reimann, M. A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of Heuristics*, 10(6):613–627, 2004.

Polacek, M. and Doerner, K.F. and Hartl, R.F. and Kiechle, G. and Reimann, M. Scheduling Periodic Customer Visits for a Traveling Salesperson. *European Journal of Operational Research*, 179(3):823–837, 2007a.

Polacek, M. and Doerner, K.F. and Hartl, R.F. and Maniezzo, V. A Variable Neighborhood Search for the Capacitated Arc Routing Problem with Intermediate Facilities. *Journal of Heuristics*, 2007b. to appear.

Prandstetter, M. and Raidl, G.R. An Integer Linear Programming Approach and a Hybrid Variable Neighborhood Search for the Car Sequencing Problem. *European Journal of Operational Research*, 2007. to appear.

Ribeiro, C.C. and Aloise, D. and Noronha, T.F. and Rocha, C. and Urrutia, S. An Efficient Implementation of a VNS/ILS Heuristic for a Real-Life Car Sequencing Problem. *European Journal of Operational Research*, 2007. to appear.

Schmid, V. and Doerner, K.F. and Hartl, R.F. and Savelsbergh, M.W.P. and Stoecher, W. Hybridization of Exact Methods and Metaheuristics for Solving Full Truckload Problems for Ready-Mixed Concrete. Presentation at *Matheuristics 2006, 1st Workshop on Mathematical Contributions to Metaheuristics*, 2006a.

Schmid, V. and Doerner, K.F. and Hartl, R.F. and Savelsbergh, M.W.P. and Stoecher, W. Hybridization of Exact Methods and Metaheuristics for Solving Full Truckload Problems for Ready-Mixed Concrete. Presentation at *IWDL 2006, International Workshop on Distribution Logistics*, 2006b.

Schmid, V. and Doerner, K.F. and Hartl, R.F. and Salazar-González, J.J. Local Branching guided by Variable Neighborhood Search for Ready-Mixed Concrete Delivery Problems. 2007a. working paper.

Schmid, V. and Doerner, K.F. and Hartl, R.F. and Savelsbergh, M.W.P. and Stoecher, W. An Effective Heuristic for Ready Mixed Concrete Delivery. Presentation at *TRISTAN VI, Sixth Triennial Symposium on Transportation Analysis*, 2007b.

Schmid, V. and Doerner, K.F. and Hartl, R.F. and Savelsbergh, M.W.P. and Stoecher, W. Hybridization of Exact and Metaheuristic Approaches: a Cooperative Local Search and Network Flow Heuristic. Presentation at *MIC 2007, Seventh International Metaheuristics Conference*, 2007c.

Schmid, V. and Doerner, K.F. and Hartl, R.F. and Savelsbergh, M.W.P. and Stoecher, W. A Hybrid Solution Approach for Ready-Mixed Concrete Delivery. 2007d. submitted.

Schrijver, A. On Cutting Planes. *Annals of Discrete Mathematics*, 9:291–296, 1980.

Tommelein, I.D. and Li, A. 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.

The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia (PA), USA, 2001.

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. As the ordered quantity of concrete typically exceeds the capacity of a single vehicle several deliveries need to be scheduled to fulfill an order. The deliveries cannot overlap and the time between consecutive deliveries has to be small.

This thesis presents a broad range of different ways on how to solve the problem stated above. Various solution methods based on exact, heuristic, meta-heuristic and hybrid approaches have been developed.

Exact methods based on a formulation the so called VRP* (a Split Delivery Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows) have been implemented. The resulting problem formulation can be solved to optimality for very small instances. For real-world-sized instances however, even with a steady increase in computational power, just to "to MIP" is not the way to success. Hence an algorithm, which controls the solution process of the embedded MIP-formulation, has been developed in order to tackler larger problem instances. This *integrative hybrid* approach is based on Local Branching (LB) which itself is guided by means of Variable Neighborhood Search (VNS). Attention has also been paid to the development of valid inequalities and cuts in order to improve the quality of lower bounds.

Another approach has been developed, which is based on a multi-commodity network flow model (MCNF) formulation. Rather than having a comprehensive view on the problem only subparts are considered and solved to optimality. So called *patterns* (options on how orders could be satisfied) are generated heuristically and serve as an input for the MCNF. Given on a set of input pattern it is possible to solve the problem to optimality.

Moreover the entire problem can be tackled by just using VNS on its own. The best results where obtained when combining the two approaches based on MCNF and VNS. Both methods used solely are capable of solving such problems. However, only the *co-operative hybrid* approach enables us to combine the strengths of both techniques and to

compensate for their major drawbacks. Iteratively solutions obtained by MCNF serve as input for VNS which is going to (locally) optimize it. The resulting solution (in terms of pattern) is fed back into the MCNF problem, which is going to be optimized again.

It can be shown that both hybrid approaches and the embedded combination of two methods are by far more efficient then the use of any approach solely. Additionally we compare our algorithm to a software available in Austria, which is based on Simulated Annealing (SA). Our hybrid algorithms outperform results obtained by the commercial product.

Abstract in German

Beton erzeugende Unternehmen sehen sich täglich vor die Aufgabe gestellt, für die Belieferung der Baustellen eine möglichst effiziente Tourenplanung - unter Berücksichtigung ihrer heterogenen Fahrzeugflotte - zu erstellen. Da der Betonbedarf einer Baustelle die Kapazität eines einzelnen Fahrzeuges übersteigt, muss in der Regel jede Baustelle mehrmals hintereinander mit Beton beliefert werden. Das Planungsproblem ergibt sich nun insbesondere daraus, dass sich aufeinander folgenden Lieferungen nicht überschneiden dürfen, da nicht mehrere Fahrzeuge gleichzeitig entladen werden können. Eventuell entstehenden Lücken zwischen aufeinanderfolgenden Lieferungen jedoch sollten möglichst kurz gehalten werden.

Im Rahmen dieser Dissertation werden mehrere Methoden besprochen, mit Hilfe derer eingangs erwähntes Tourenplanungsproblem gelöst werden kann. Die angewendeten Konzepte basieren auf exakten Verfahren, Heuristiken, Metaheuristiken, sowie hybriden Ansätzen.

Ein exaktes Modell, beruhend auf einer Erweiterung des klassischen *Vehicle Routing Problems* (VRP, Tourenplanungsproblem) wurde entwickelt. Allerdings lässt sich die daraus resultierende Formulierung nur für äußerst kleine Instanzen exakt lösen. In der Praxis hingegen, ist dieser Ansatz aufgrund der zu langen Rechenzeiten und des enormen Rechenaufwandes nicht sinnvoll anwendbar. Daher wurde ein von *Local Branching* (LB) inspiriertes Verfahren konzipiert. Dieser *integrativ hybride* Ansatz wendet zusätzlich Nachbarschaftstrukturen, wie sie auch bei *Variable Neighborhood Search* (VNS) angewendet werden, kombiniert an. Darüber hinaus wurden *valid inequalities* für eine Verbesserung der unteren Schranken herangezogen.

Ein weiterer Ansatz beruht auf einer Formulierung für *multi-commodity network flow* Problemen (MCNF). Anstatt einer globalen Sicht auf das Problem an sich, werden in diesem Zusammenhang nur ausgewählte Subbereiche näher betrachtet. So genannte *Muster* werden für alle Bestellungen generiert. Jedes Muster legt eindeutig fest, wie und wann ein bestimmter Auftrag abgewickelt werden könnte. Neben der Auswahl der Fahrzeuge wird auch der zeitliche Ablauf der Lieferungen aus Sicht der Baustelle bestimmt. All diese Muster dienen als Input für die MCNF Formulierung, dessen Hauptaufgabe die Selektion

genau eines Musters pro Auftrag darstellt. Gleichzeitig muss sichergestellt werden, dass alle daraus resultierenden Touren aus Sicht der Fahrzeuge zeitlich tatsächlich durchführbar sind.

Darüber hinaus kann die Problemstellung auch allein mithilfe von VNS gelöst werden. Beide Verfahren sind zwar eigenständig in der Lage das Problem zu lösen, die besten Ergebnisse wurden jedoch durch eine Kombination dieser Ansätze erzielt. Nur durch eine gezielte *kooperative* Verzahnung der beiden Ansätze ist in der Lage die Stärken der beiden zusammenzulegen und etwaige Nachteile zu verringern. Lösungen die vom MCNF-Ansatz erzielt werden, dienen als Input für den VNS Ansatz. Dieser wiederum füttert sämtliche neuen besten Lösungen (im Sinne der oben erwähnten Muster) zurück an das MCNF, welches daraufhin einen erneuten Auswahlprozess initiiert.

Es wurde gezeigt, dass sich mit beiden hybriden Ansätzen ausgezeichnete Ergebnisse erzielen lassen. Der verzahnte Ansatz ist um einiges effizienter als die jeweils einzeln angewandten Verfahren. Weiters werden die erzielten Ergebnisse mit einer Softwarelösung, welches basierend auf *Simulated Annealing* (SA) entwickelt wurde, verglichen. Die oben erwähnten Verfahren sind in der Lage diese Ergebnisse in hohem Maße zu übertreffen.

verenaschmid

Personal Data

date of birth: August 13, 1981 in Feldkirch, Austria

citizenship: Austria

Education

2004-2008 **Master Program in Business Informatics**.
University of Vienna, Vienna, Austria

2003–2007 **Doctoral Program in International Business Administration**.
University of Vienna, Vienna, Austria

Aug–Dec 2002 **Exchange Program**.
University of Illinois at Urbana-Champaign, Urbana-Champaign, USA

2000–2004 **Bachelor Program in Business Informatics**.
Vienna University of Technology, Vienna, Austria

1999–2003 **Master Program in International Business Administration**.
University of Vienna, Vienna, Austria

PhD Thesis

title: *Trucks in Movement*

supervisor: Richard F. Hartl

description: Hybridization of Exact Approaches and Variable Neighborhood Search for the Delivery of Ready-Mixed Concrete

Master Thesis

title: *Der Einsatz von Materialflussanalyse bei der logistischen Beurteilung einer Ersatzinvestition am Beispiel eines Infrastrukturzulieferunternehmens*

supervisor: Manfred Gronalt

Experience

Vocational

2003–2007 **Research Assistant**, *University of Vienna*, Vienna.
Department of Business Administration - Production and Operations Management

Mar–Jun 2003 **Tutor**, *Vienna University of Technology*, Vienna.
Institute of Computer Languages - Theory and Logic Group

Talks & Presentations

October 2007 **Annual Meeting of Austrian Society of Operations Research (OEGOR)**.
A Hybrid Solution Approach for Ready-Mixed Concrete Delivery
V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh and W. Stoecher

June 2007 **MIC 2007 (Montreal, Canada)**
Metaheuristics International Conference.
Hybridization of exact and metaheuristic approaches: a cooperative VNS and network flow heuristic
V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh and W. Stoecher

Fleischmanngasse 4/26 • 1040 Wien, Austria
✉ *verena.schmid@univie.ac.at*

June 2007 **Tristan IV (Phuket, Thailand)**
Sixth Triennial Symposium on Transportation Analysis.
An effective heuristic for ready mixed concrete delivery
V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh and W. Stoecher

October 2006 **IWDL 2006 (Brescia, Italy)**
International Workshop on Distribution Logistics.
Hybridization of exact methods and metaheuristics for solving full truck load problems for ready-mixed concrete
V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh, and W. Stoecher

August 2006 **Matheuristics 2006 (Bertinoro, Italy)**
1st Workshop on Mathematical Contributions to Metaheuristics.
Hybridization of exact methods and metaheuristics for solving full truck load problems for ready-mixed concrete
V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh and W. Stoecher

November 2005 **SimForumSteyr'05 (Linz, Austria)**
Produktions- und Logistiksysteme am optimalen Betriebspunkt.
Simulation des Schiffsverkehrs auf der Donau durch Prioritäten- und Reihenfolgeplanung - Vorbild für die Fertigungslogistik?
V. Schmid and R. Fromwald

Papers

submitted **A Hybrid Solution Approach for Ready-Mixed Concrete Delivery**.
V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh and W. Stoecher

working paper **Local Branching guided by Variable Neighborhood Search for Ready-Mixed Concrete Delivery Problems**.
V. Schmid, K.F. Doerner, R.F. Hartl and J.J. Salazar Gonzáles

Teaching

summer term **Linear Programming using XPress-MP**, University of Vienna.
Linear Programming and Modeling, Sensitivity Analysis, Duality, Advanced Modeling Techniques
Applications: Transportation, -shipment, Ressource Allocation, Location, TSP, VRP, etc.

winter term **Simulation using AnyLogic**, University of Vienna.
Queueing Theory, Discrete Event Simulation
Applications: Service, Manufacturing Models, Shop Floors, Health Care

Languages

german	mother tongue	
english	advanced, fluent in written and spoken	*toefl score 263*
spanish	fluent	*diploma básico de la lengua española*

Computer skills

programming	C++, Java, Visual Basic
optimization	Xpress, Gams, Cplex
simulation	AnyLogic, Arena

Interests

research	Variable Neighborhood Search, Linear Programming, Hybridization
traveling	South East Asia, South America
hobbies	Boxing, Skiing, Cycling

Fleischmanngasse 4/26 • 1040 Wien, Austria
✉ *verena.schmid@univie.ac.at*