Freight Transportation
-Problems Related to Railways

Thesis Proposal

Submitted by:
Lukas Bach

Supervisor:	Associate Professor Sanne Wøhlk
Committee:	Professor Christian Larsen Associate Professor Jesper Larsen Associate Professor Sanne Wøhlk

Department of Economics and Business
School of Business and Social Sciences
Aarhus University
January 2012

Contents

Chapter 1

Introduction

The overall project is concerned with optimization of freight transportation, with particular focus on problems involving railways. This PhD project proposal is divided into 3 stages, first we present a lower bound method for the node, edge, and arc routing problem (NEARP) in chapter 3. Then in chapters 4 and 5 we present two proposal that together involves the overall planning process at a freight railway operator, which is applied to a case at DB Schencker Rail Scandinavia A/S (DBSRS). Finally we have a proposal regarding an intermodal planning problem from a freight forwarder perspective in chapter 6.

In chapter 3, which concern the paper *Lower and Upper Bounds for the Node, Edge, and Arc Routing Problem* we present an extended abstract that has been submitted for presentation at the conference Odysseus 2012. In the appendix the full version of the paper can be found, this is also the version already submitted to *Computers & Operations Research* in November 2011. This paper is joint work with Geir Hasle, SINTEF ICT, Dept. of Applied Mathematics, Norway and my supervisor Sanne Wøhlk.

Stage 2 of the project is divided into two proposals; I and II which is concerned with scheduling of engines and crew for the aforementioned freight railway operator. In proposal I, two scheduling problems will be handled in an integrated approach. This concerns a timetabling problem and a engine routing problem at DBSRS. Proposal I in chapter 4, is presented as a draft paper, that in some respects are well developed, but with respect to e.g. the branching and implementation still has to be further developed.

The second proposal in chapter 5 consider the crew planning at a freight railway operator, again with DBSRS as case. Furthermore, as we try to provide a degree of integration between the crew planning phase and the timetable and engine planning phase presented in proposal I. This is done in order to see how much there is to gain from combining these two parts of the planning process. In proposal II, we present the problem and give scenarios that could form the integration with proposal I. We do also as an outline provide some modeling of the crew planning problem. In general this proposal is at an earlier stage than proposal I.

Stage 3 is the intermodal problem presented in chapter 6. Proposal III consider planning seen from a freight forwarder perspective, where optimization methods are used to improve utiliza-

tion of railways in freight forwarding. This projects thus seek to increase the amount of intermodal transportation. We will consider point to point deliveries, while using a fixed IM-railway timetable as well as trucks in the planning of the transportation. The maturity level of proposal 3 is relatively low.

Finally in chapter 2 we present the PhD-plan for which courses has been (and will be attended), along with the teaching obligations covered, and with a project plan for the development of the proposals.

Chapter 2

Activities

During the first year I have been working on two projects, the NEARP project and the project described as proposal I. I have also conducted various teaching activities both as teaching assistant, coordinator for teaching assistants and corrected exams. Furthermore I will at the beginning of February 2012 have attended relevant PhD courses.

Table 2.1: Project plan

	S11	F11	S12	F12	S13	F13
NEARP	X					
Paper I		X	X			
Paper II			X	X		
Paper III					X	X
Teaching		X		X		
Courses	X	X				
Change of environment			X		X	

2.1 Change of Environment

From February 13th to August 7th 2012 a visit at CIRRELT at University of Montreal is planned. Here I will be under the supervision of Professor Michel Gendreau. A second change of environment is planned for spring 2013, destination is still unknown.

2.2 Courses

As a part of the PhD-program it is required that I take the equivalent of 30 ECTS units of courses. During this year I have taken the courses shown in 2.2, these courses represent an

ECTS load of 32 units. The courses span different areas of Operations Research, from practical courses on Reviewing of papers to courses on both exact and approximate methods.

Table 2.2: Courses taken, by February 2012

Term	Course / University	ECTS
Spring 2011	Reviewing papers on O.R. applications in Logistics/SCM/OM *School of Business and Social Sciences, AU*	5
Apr. 2011	Stochastic Modeling *NATCOR, Lancaster University*	3,5
Fall 2011	Branch & Bound and CPLEX Implementations *School of Business and Social Sciences, AU*	5
Aug. 2011	The Set Partitioning Optimization Model *Technical University of Denmark*	5
Sep. 2011	Combinatorial Optimization *NATCOR, University of Southampton*	3,5
Jan. 2012	Winter School on Optimization in Logistics and Transportation *CIRRELT, University of Montreal & CIO, University of Lisbon*	5
Feb. 2012	Metaheuristics with Applications in Logistics *School of Business and Social Sciences, AU*	5
Total		32

2.3 Teaching

The requirement for teaching activities are a total of 570 hours over the 3 year program. I will by mid February 2012 have covered about 500 hours which is about 88% of the teaching obligation.

Table 2.3: Performed teaching activities, by February 2012

Course	Code	Term	Description	Hours
Advanced Excel	29063	Jan. 2011	Eval take-home paper	76
Advanced Excel	29063	Fall 2011	Class teaching	62,5
Driftsøkonomiske planlægningsmodeller	29204	Fall 2011	Class teaching (4x55)	220
Driftsøkonomiske planlægningsmodeller	29204	Fall 2011	Coordinator of TAs	35
Simulation: Model. & Analysis		Fall 2011	Class teaching	30
Advanced Excel	29063	Jan. 2012	Eval take-home paper	40
Driftsøkonomiske planlægningsmodeller	29204	Jan. 2012	Eval written exam 4 hours	40
Total finished				503,5
Advanced Excel	29063	Jan. 2013	Eval take-home paper	40
Driftsøkonomiske planlægningsmodeller	29204	Jan. 2013	Eval written exam 4 hours	26,5
Total finished and remaining				570

Chapter 3

Paper I: Lower and Upper Bounds for the Node, Edge, and Arc Routing Problem

This paper is joint work with Geir Hasle, SINTEF ICT, Dept. of Applied Mathematics, Norway and my supervisor Sanne Wøhlk, CORAL, Dept. of Economics and Business, Aarhus University, Denmark. The paper was submitted to *Computers & Operations Research* in November 2011. The paper is also submitted for presentation at Odysseus 2012.

The following is an extended abstract of the paper.

3.1 Background

The Node, Edge, and Arc Routing Problem (NEARP) was defined by Prins and Bouchenoua in 2004 Prins and Bouchenoua (2004). The NEARP generalizes the classical CVRP, the CARP, and the General Routing Problem. It captures important aspects of real-life routing problems that were not adequately modeled in previous VRP variants. Hence, its definition and investigations contribute to the development of rich VRPs.

Examples of arc routing problems are street sweeping, gritting, and snow clearing. However, the arc routing model has been advocated in the literature for problems where the demand is located in nodes, for instance distribution of subscription newspapers to households and municipal pickup of waste. In real-life cases, there are often thousands of points to be serviced along a subset of all road links in the area. Such cases are often formulated as CARPs, typically with a large reduction of problem size. However, the reduction from points to arcs does not always provide an adequate model.

In their 2004 paper Prins and Bouchenoua (2004), Prins and Bouchenoua motivate the Node, Edge, and Arc Routing Problem (NEARP). They state that:

Despite the success of metaheuristics for the VRP and the CARP, it is clear that these two problems cannot formalize the requirements of many real-world scenarios.

There are good reasons to merge the arc / node routing problems and enable modeling of real-life situations, where they together form a better representation of reality. The introduction of the NEARP was a significant step towards the goal of rich VRP. Despite its importance, studies of the NEARP following its introduction are almost non-existent in the literature. Prins and Bouchenoua also created the CBMix benchmark and provided the first upper bounds for it using their memetic algorithm. Kokubugata and Kawashima Kokubugata et al. (2007) study problems from city logistics, including the VRP with Time Windows and the NEARP. They propose a Simulated Annealing metaheuristic for solving these problems. Computational results for the CBMix instances of Prins and Bouchenoua are presented, with several improvements. In Hasle et al. (2011) Hasle et al. report new best known results for the CBMix instances. To add to the NEARP literature, we wish to provide the first (to the best of our knowledge) lower bound for the NEARP.

Lower bounds have been developed for many VRP variants. Many of these are based on cutting planes. Also for the General Routing Problem, there is a tradition of obtaining lower bounds through algorithms involving cutting planes. In contrast to VRP the tradition for CARP are to develop combinatorial lower bounds. The majority of these are based on the construction of one or several matchings. The best such lower bound is the Multiple Cuts Node Duplication Lower Bound (MCNDLB), Wøhlk (2006).

3.2 Lower Bound for NEARP

The algorithm is a further development of the Multiple Cuts Node Duplication Lower Bound (MCNDLB) for the CARP.

For notational reasons, in the description of the algorithm we will assume that the graph is simple, i.e. that there is at most one required link between any pair of nodes. We stress that the algorithm can easily be extended to the non-simple case.

Starting with $U_1 = \{1\}$, we consider mutually disjoint cuts $(U_k\,,\, N \setminus U_k)$ such that $U_1 \subset U_2 \subset \ldots \subset U_k \subset U_{k+1}$. For each such cut, U_k, the graph induced by $N \setminus U_k$ will consist of one or more connected components, $G'_s = (N'_s, E'_s, A'_s), s = 1, \ldots, t$, as illustrated in figure 3.1. The number of vehicles needed to service the demand in G'_s and the demand of links connecting G'_s to U_k can be estimated by $p_s = \lceil (\sum_{i \in N'_s} q_i + \sum_{(i,j) \in E'_s \cup A'_s \cup \delta(N'_s)} q_{ij})/Q \rceil$.

Ideally, each vehicle would service the demand of an edge or arc when entering G'_s and when leaving G'_s. When this is not the case, we say that the vehicle is using an artificial link. Such links can be either links without demand or links with demand not currently being serviced. We estimate the number of artificial links (entering arcs, leaving arcs, and undirected edges) needed for all vehicles to both enter and leave G'_s. With this, we can estimate the cost of servicing demand in G'_s and demand of links connecting G'_s to U_k by constructing a node duplicated network and letting m_s be the cost of a minimum cost perfect matching in this network. We do

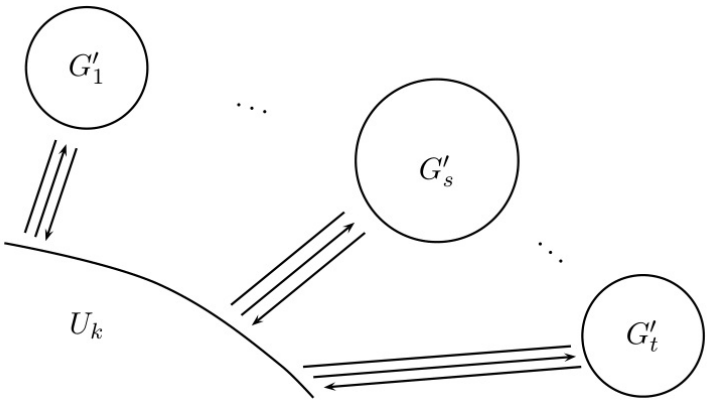


Figure 3.1: In each iteration, G is partitioned into U_k and a number of connected components, G'_s.

this for all the connected components and hence, $L = \sum_{s=1}^{t} m_s$ estimates the cost of servicing everything outside $G(U_k)$.

To estimate the cost of servicing demand in $G(U_k)$, we use the minimum cost $\overline{c}_s$ of a link between U and each component, G'_s and multiply this by the number of artificial links needed to connect the two: r_s. Iterating over all the mutually disjoint cuts and all the connected components of these, we can estimate the cost of servicing the demand in $G(U_k)$ as $L1 = \sum_{j=1}^{k-1} \sum_{s=1}^{t} \overline{c}_s r_s$.

For each of these cuts, $L + L1$ is a lower bound on the cost, and the algorithm selects the highest of these.

Note in the details of the algorithm that the calculations become more complex than outlined above due to the existence of both directed and undirected links. This is because the artificial links from U and each component are split into a directed and undirected part, enabling us to exploit the asymmetric cost structure the arcs create in the NEARP instances. This asymmetric cost structure is also used in the matching network, which also strengthens the bound.

3.3 Computational experiments

Only one set of benchmark instances exists for the NEARP: the CBMix instances Prins and Bouchenoua (2004). These instances are all based on graphs with a grid structure. To ensure more variation of the test platform for future algorithm developments and for testing the lower bound algorithm described above, we have developed two new sets of benchmark instances. The first set is called the BHW benchmark and is based on 20 classical CARP instances. Some edges have been transformed to directed arcs, and node demands have been added. The second set, the DI-NEARP benchmark, contains 24 instances. It is based on six real-life NEARP cases from the design of carrier routes for home delivery of subscription newspapers and other products in the Nordic countries.

For the existing CBMix instances, we have compared the best known upper bounds with our lower bound. Here the tests result in gaps between the LB and UB from 3.0 % to 39.9 %. For

the BHW and DI-NEARP instances the first upper bounds will be provided and compared to the lower bound.

Chapter 4

Proposal I: Freight Railway Operator Timetabling and Engine Routing

4.1 Introduction

When planning the timetable a railway operator has to apply for usage of railway infrastructure. In Europe the organization RailNetEurope(RNE) work to harmonize the access to infrastructure in their 38 member countries. The operators can, when designing their schedule apply for a train path, which is an origin and destination pair with a given departure time and transit times. These paths can be designed and applied for in multiple ways.

The paths are designed either by RNE, which is called a catalogue path, or the railway operator can design and apply for path themselves, which is called a tailor-made path. Per definition the catalogue paths comply with the regulations etc. for the paths to be feasible.

This paper focus on choosing among the RNE designed catalogue paths. Thus the problem is to chose the set of paths that reduces the operating costs of the railway operator. We apply this problem to a case arising at a railway operator, which manage the timetabling of block trains.

DB Schenker Rail Scandinavia A/S (DBSRS) is a railway operator who manage transports which originates from and/or has destination in a Scandinavian country. The core business of DBSRS is to provide engines and engine drivers to move customer's wagons between stations according to long term contracts. More specifically DBSRS is used by its mother companies to handle the timetabling and to allocate drivers and engines to the planned trips in the corridor 1 area from Maschen (Hamburg) in south to Hallsberg (in southern Sweden) in the north.

In section 4.2 we describe the problem and case in detail. Then in section 4.3 we present the modeling of the problem before we give a solution approach in section 4.4. Implementation details are provided in section 4.5.

4.2 Problem and case description

In this section we describe the problem in general and add to this the case specific details from DBSRS and the network they operate in. The overall goal of the problem is to minimize total costs for the railway operator when designing a timetable, the total costs can include both engine usage, track usage and other driving related costs. In the DBSRS case we have a given set of engines at hand, and the opportunity costs for these engines are not fixed. Thus it is difficult to associate the engines with a fixed charge. A review of railway routing and scheduling can be found in Cordeau et al. (1998), for a more recent review of railway optimization see Lusby et al. (2011).

The horizon for the timetable is weekly where it has to be repeated for a year. The timetable design is typically done more than half a year in advance. The contracts between DBSRS and their customers varies in length and scope, however, without loss generality the contracts all last for at least a year, which covers the timetabling period of a year.

The goal is subject to a set of constraints, first we can describe the network that the railway operator use. The general problem adheres to the RNE area, where catalogue paths are published. In this paper we only consider catalogue paths and thus do not design our own additional tailormade paths. To implement additional paths Kuo et al. (2010) suggest an iterative approach to design and implement tailormade paths in a timetabling problem. The network DBSRS operate in, covers a total of 15 stations that are connected in a tree structure such that there is only one path between any two stations.

The rail network is managed by a different infrastructure manager in each of the three countries wherein DBSRS operate. Within the the corridors time slots are allocated by RNE, corridors are stretches of railway that are coordinated by RNE. Time-slots are applied for through the local infrastructure manager (in Denmark "BaneDanmark") during spring the year before an operator actually use the tracks. Basically this requires contracts with clients to be made almost a year in advance.

The cost structure for using the tracks is also deviating. The first cost category is quite simple and is enforced per kilometer driven. In Denmark there is one constant charge for using a kilometer of rails, in Sweden there are two different charge levels, a base charge and a high charge. The high charge covers the network in the triangle between Stockholm, Gothenburg and Malmo. In Germany there is a more complex system that consist of different priority categories for the train driving where the cost varies depending on which priority chosen.

The cost in the model is not stationary and vary over time as the cost of using tracks in some areas are subject to a capacity charge. Capacity costs are designed in order to reduce traffic at certain times are enforced in Denmark and Sweden. In Denmark it is simple as the charge applies if the train is situated on any parts of the sections covered by the capacity charge within the time window from 7:00 to 18:59. The rule covers three sections of the network; Copenhagen airport Kastrup - Kalvebod, Hvidovre fjern - Hoeje Taastrup and Vojens - Vamdrup. In Sweden they have a similar system but here the time windows are from 7:00-9:00 and from 16:00-18:00 covering the passage of a number of sections in Stockholm, Gothenburg and Malmo. Here there are a total of five areas every time a train enters one of these areas a charge is made.

The demand in this model is strictly "block train" demand, such that any demand is for a full train driving from an origin to an arrival station. Hence it is not possible to aggregate demand. For each of these demands we have a fixed time-window wherein we have to start service, as transit times are quite stable, the start of service also implies an end of service time-window.

Currently the planning department at DBSRS has approximately 230 demands per week to cover. DBSRS handle two types of service, provided by separate departments within the organization; the Intermodal service and the Freight Logistics service. The two services are similar in the sense that they are both "block train" services w.r.t. DBSRS. For each demand there is a time-window, the width of the time-windows varies from customer to customer, but in general they are wider for the freight logistics (FL) customers than the intermodal (IM) customers. As DBSRS are a transit company the IM-demands are mainly originating from its two mother companies. In general the IM time-windows are about one to two hours wide whereas the FL time-windows are about 6 hours wide.

Each coverage of a demand is associated with some preparation time at the origin station and finalization time at the destination. This time is mainly used for shunting wagons but there can also be a safety margin that serves to prevent propagation of delays.

The model allows for multiple engine types, these can be different in the aspects of which demands they can serve, costs and safety margins and such. DBSRS operates two different types of engines, diesel and electric. The diesel engines cannot be operated in Germany and Sweden. This is because no environmental approval have been granted for this engine type in these countries. Secondly, the electric engines cannot be used various places. This is due to the fact that the track network is not completely electrified. Further there are 2 types of electric engines, where they distinct themselves in engine size such that one can pull heavier loads than the other. Diesel engines must be fueled at the end of each trip, a process which takes 15 minutes.

The transit times and time-slots in the DBSRS operational area on the main pathway of corridor 1 are identified using the information from RailNetEurope. This approach cannot be used to identify transit times and time-slots on legs outside the main pathway. The time-slots are not specified for legs outside the main pathway. Thus we design these by letting time-slots connect to the time-slots in the main pathway.

The model does not consider maintenance. In the DBSRS case, maintenance are handled approximately every 10,000 kilometers driven. Here the engines are taken out of service and replaced by a spare engine that continues the engine rotation.

4.3 Modeling

The problem can be described as a timetabling problem, where a set of heterogeneous engines have to serve a set of full-load demands that cannot be served simultaneously. Each demand is given a time-window during which the service of the demand has to start where a time-slot is available. The time-slots are laid out by the infrastructure manager(BaneDanmark), that set these in advance. There is approximately a time-slot every half hour and with time-windows between 1 and 6 hours there is a limited number of possible departure times to choose among.

The planning horizon for the schedule developed is one week, whereas the rotation of the engines can last for as long as necessary. In Cacchiani et al. (2008) perform timetabling in a similar corridor and Ziarati et al. (1997) performs engine assignment with a heterogeneous fleet.

The goal of the model is to produce a weekly timetable that minimizes the total costs, an overview of the model can be seen in figure 4.1. In the following section we will cover the handling and implications of the constraints.

Figure 4.1: Model overview

minimize	Total costs
subject to	Demand coverage
	Availability of engines
	Replication of weekly timetable
	Time-slot usage
	Time-slot compliance
	Time-window compliance
	Flow conservation
	Engine type compliance
	Waiting time after demands
	Transit time between demands

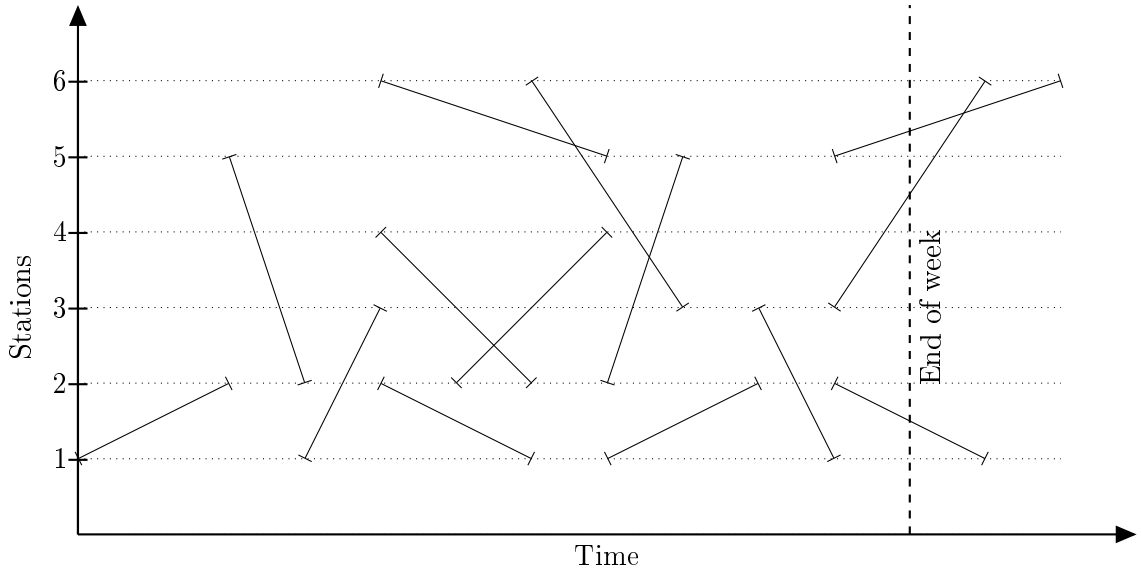
The demand coverage is straight forward, as we are interested in covering all the demands exactly once. Here we do not allow for multiple coverings of the demand. The engines are divided in multiple categories and for each category we have a fixed number of engines available. There can be a cost associated with the usage of the different engine types.

The replication of weekly timetables two things will be handled in the following, first we need to define how we handle the circular nature of the weekly plan. Secondly we have to decide how the engine rotations will be designed and the length hereof. A major problem with the weekly plan, is that there is no time-period during the week where the engines has to be grounded. This forces the planning period to be circular in the terms that an engine departing at the very end of the week will arrive at the beginning of the week, see figure 4.2.

Nahapetyan and Lawphongpanich (2007) solve a dynamic traffic assignment problem, where they also use the circular approach in opposition to starting and ending with an empty system. Caimi et al. (2011) use the Periodic Event Scheduling Model (PESP) on a Infrastructure Management level to generate feasible time-slot allocations. In PESP we have a time horizon over which events are repeated, this is often 1 hour. Lindner and Zimmermann (2005) use the PESP model, to create cost optimal train schedules for a railway operator. As our events in this case are periodic over a week, this would have to be set to 1 week. The PESP model consider the design of time-slots or at least the maximum time between different events such as two different trains departing the same station and the headway between them. This is in our case handled by the choice of time-slots and avoiding overlaps hereof.

For the replication of weekly cycles can be handled in several ways; one approach is to let

Figure 4.2: Weekly timetable - example



Note: *A line in the figure represent a demand in a timetable that is driven from the station at the beginning to the station at the end of the line.*

each engine follow a repetitive weekly plan such that each engine have the same routing every week. Another way is to keep the weekly timetable but plan with a longer finite horizon for the individual engines e.g. 4 weeks. The approach chosen here is to have a weekly routing that is not engine specific, in the sense that it is not necessarily operated by the same engine every week.

For the approach chosen to be viable we have to have a way to ensure the the different routings fit together. We would like to avoid a complex formulation where we have to balance the weekly routings with respect to both place and time. With respect to place, there are no depots thus we treat all stations as depots such that we have a problem with multiple depots. This is important because an engine has to be ready at the end of one week at a place, that fits with the starting place of the following week.

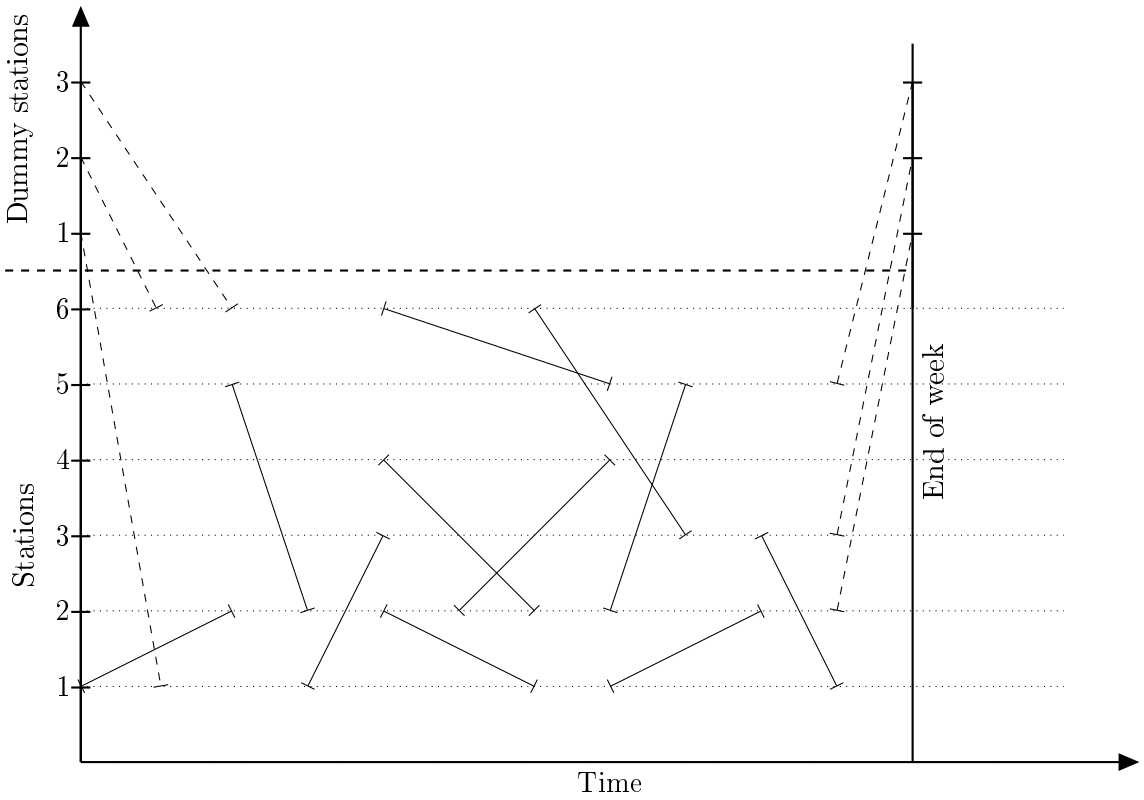
Because there is no time during the week where all engines or groups hereof are grounded. We have no natural point that we can designate as the beginning of a week. To remove the time factor, such that the start of the week can be kept fixed at the same time for all engines, we could chose a point during the week, where no - or the least - demands cross and then balance the weekly routings at this point.

If we consider removing the time factor, we have a problem with defining the beginning/end of the week. In figure 4.2 we see a placement of the end of week (eow). It can be done in such a way that demands starting in the last part of a week continue into the next week, which is essentially the beginning of the week. When balancing the number of weekly routings having a given station as origin station with the number having it as destination, we have to make sure that the origins time-wise lie after the destinations, otherwise we cannot guarantee that the engines are actually available.

As we would like to remove the time factor, we now show how we can make sure that the first station of any route, lies earlier in time than the last station for any route. To break the circularity of the weekly timetable we proposed to find the point during a week that interferes with the least demands. At this point we split all demands that crosses it in two parts, one from the origin till the eow and one from the start of week until arrival. We call these two parts, partial demands. These two partial demands are linked to what we call a dummy station, at the time it was split, this concept is shown in figure 4.3. Each dummy station is only used for one single demand using a specific time-slot.

The dummy station ensures that there will be one weekly routing, that starts with the first of the partial demands and one that ends with the last partial demand. Whether these two partial demands are on the same routing is not important. If they are on the same there will be one circular routing that will be driven by the same engine each week. If they are not on the same routing, they will still be driven by the engine, but now they force the two weekly routings involving them to be merged into a rotation of at least two weeks. At this point we do not know and do not care what the other origin and destination stations are, as they will be balance accordingly.

Figure 4.3: Weekly timetable with dummy stations - example



For all demands not crossing the eow we do not change anything, as it is certain that the first station in a routing is the origin and the last is the arrival and that these can never cross the eow. Thus when the flow in and out of ordinary and dummy stations are balanced, we ensure

that all routings are able to be part of an engine rotation, with a possible infinite time horizon.

After handling the replication of weekly timetable, we return to the remaining constraints. The time-slot usage governs that we can only plan one train per time-slot. Hence two demands that use the same or have partially overlapping routes, would have to be serviced at times such that no time-slot is not used twice.

Time-slot and time-window compliance are related in the way that the start of service of a demand have to be within the time-window. Within this time-window we have to select a time-slot, thus it is not enough just to be within the time-window. Therefore any time-window will have an associated set of time-slots, that can be used.

Engine type compliance, ensure that the right engine is used for a given demand. For all demands a subset of the engine types can be used to service it, this is mainly due to the previously described electrification issues. Thus it is important that we assign the right engine type to a demand to keep the timetable feasible.

Flow conservation and transit time between demands, compliment each other. Flow conservation ensure that the flow in and out of stations are equal and transit time between demands ensure that there are enough time to complete possible reposition trips between two jobs. It also concern the necessary time to prepare for service of the following demand. Waiting time after demands handles the safety margins that are planned after the completion of each demand, to avoid propagation of delays.

4.4 Solution approach

The problem of developing the weekly timetable is as previously described constrained by multiple constraints. Combining these into a joint formulation of the problem will result in a complex formulation. Instead a column generation approach is chosen, it allows us to split the problem in two and thus reduce the complexity. We use a Dantzig-Wolfe decomposition approach to split the problem in a master and sub-problem.

The model will be split in two such that all constraints concerning the individual engine will be dealt with in the sub-problem. In the master-problem we will consider replication of weekly timetables, time-slot usage, availability of engines and demand coverage, which is because these do apply to multiple engines at once. By solving the sub-problem we can generate engine routings that in the master-problem can be combined into a weekly timetable, w.r.t. the constraints in the master-problem. The splitting of the joint problem into sub and master-problems can be seen in figure 4.4. When considering the sub-problem it can be seen that there are numerous possible combinations hereof. This would be a problem if we where to use a complete enumeration approach where all columns for the master-problem where generated at the beginning. Instead we use a delayed column generation approach, where we generate the columns that would enter the basis i.e. the column with the most favorable reduced cost each time we solve this restricted master problem to optimality. The restricted master problem is the master-problem concerning a limited number of the columns that would be generated for the full master problem.

The integer properties of the problem are handled in the sub-problem where we only get integer

Figure 4.4: Model overview - split into master and sub-problem

minimize	total costs	
subject to	Demand coverage	$master - problem$
	Availability of engines	
	Replication of weekly timetable	
	Time-slot usage	
	Time-slot compliance	$sub - problem$
	Time-window compliance	
	Engine type compliance	
	Flow conservation	
	Transit time between demands	
	Waiting time after demands	

solutions, but in the restricted master problem we solve a LP-relaxation of the problem. Then at each time we have solved the LP-relaxation to optimality we branch the problem and remove possible infeasible columns before we re-optimize the LP-relaxation of the restricted master-problem. This process continues until we arrive at an integer solution to the restricted master-problem that satisfy the optimality conditions. When solving the LP-relaxation of the restricted master-problem from now on Z_{RMP}, we would like to be able to terminate at an earlier stage in each root node. Thus when branching we can apply a lower bound on the optimal integer solution in each node. Because we know that the best Z_{IP} for any node can be bound by the lower bound Z_{lb} found in equation 4.1. Among other Dell'Amico et al. (2006) show that Z_{IP} can be bounded by the Z_{RMP} and column with the lowest reduced cost. This implies that we can possibly terminate the LP-relaxation of the RMP at an earlier stage and then branch again, which would possible speed up the solution process.

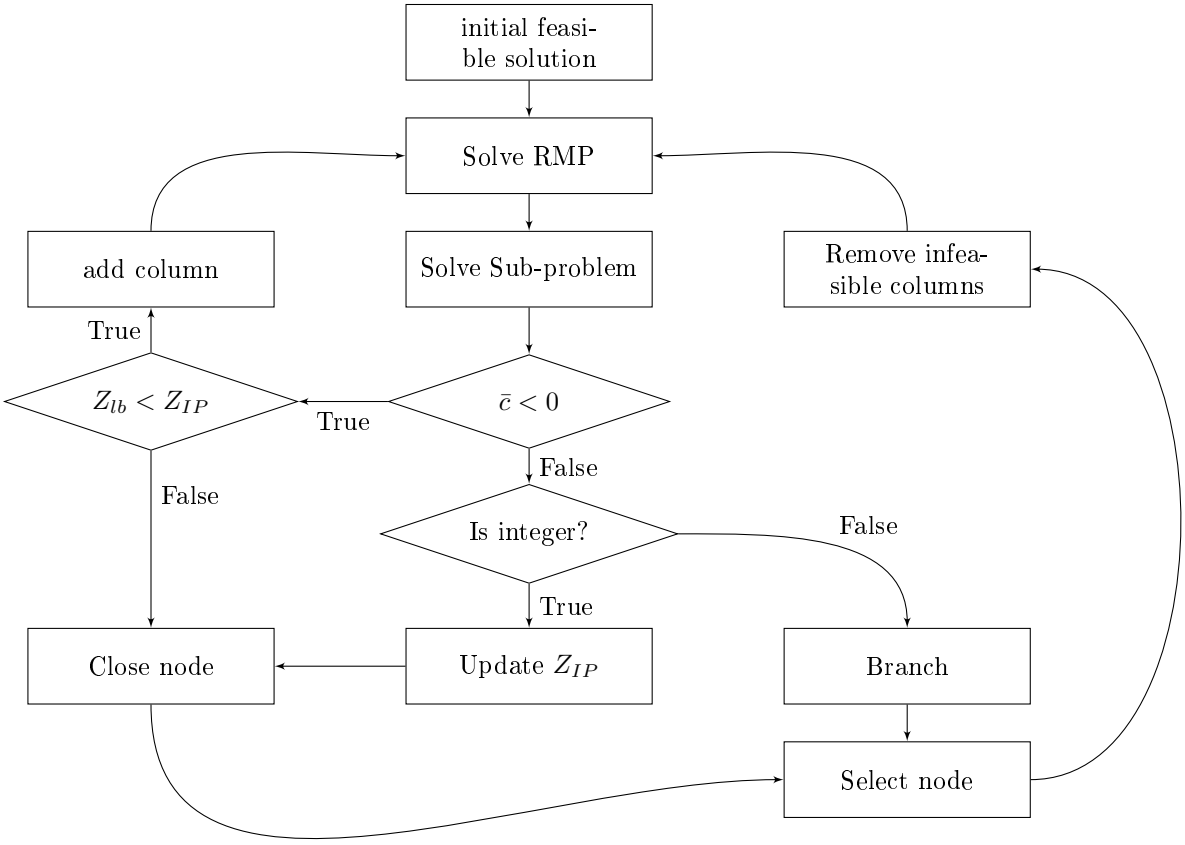
$$Z_{lb} = Z_{RMP} + \bar{c} \tag{4.1}$$

In figure 4.5 we show the overall column generation algorithm. The algorithm is initialized by a set of columns allowing for a feasible solution. Normally this can be handled by a identity matrix, but in this case it is not enough. Because we have a constraint on the engines available, an identity matrix for the demands would not necessarily provide a feasible solution as it would probably violate the engine availability constraints, as it implies that there will be one engine per demand. Instead we will have an initialization phase where we get to a initial feasible starting point, this will be described later in this section.

After solving the RMP, we solve the sub-problem to get the column with the lowest reduced cost. Two conditions has to be satisfied to check whether we let the column enter the RMP and resolve it. First we check if the reduced cost is negative, if not then the RMP is optimal and we have resolved the node. Second if the reduced cost is negative we use our lower bound to check if we can achieve a solution that is better than the best integer solution already obtained. If we can not close the node in this way, we add the column to the RMP and resolve it.

If the node is solved to optimality, we check if we have an integer solution. If the solution is

Figure 4.5: Column generation algorithm



integer we update the best integer solution and close the node, otherwise we branch on the node. In both cases we will the return to selecting a new node solve. Before solving the node, we remove columns that are infeasible in the selected node of the branching tree. The algorithm terminates when there are no more nodes to select.

Figure 4.5 gave an overview of the algorithm. In the following sections, we will elaborate on the details in the algorithm and will as well formulate the master-problem, restricted master-problem, sub-problem and the branching techniques.

4.4.1 Master-problem

In this section we will first show the formulation of the full master-problem then we will deduce the restricted master-problem and the linear relaxation hereof. The full master-problem is formulated with the parameters given in table 4.1 and equations 4.2 through 4.7. As columns in the master-problem we use routings generated in the sub-problem, thus selecting a column represent using the corresponding routing for an engine. The only variable in the model x_r represent choosing

this column and equals 1 if a given column is used and 0 otherwise. The set R is the set of columns r under consideration, hence this set will be adjusted as we add and remove columns in the restricted master-problem.

The objective function is given in equation 4.2, where c_r is the total cost for the column that corresponds to a given routing for a specific engine type. Thus we try to minimize total costs, which are composed by a fixed engine usage cost and driving costs, both are engine type specific.

Table 4.1: Parameters

Parameter	Description
N	Set of all Stations including dummy stations, indexed by n
D	Set of all Demands, d
S	Set of all time-slots, s
E	Set of Engine types, e
w_e	Engine availability for engine type e
c_r	Cost of routing r
α_r^d	Demand d is covered on route r, 1 = yes, 0 = no
β_r^n	Station n is used on route r, as origin station = 1, as destination = -1, as both = 0 or neither = 0
γ_r^s	Time-slot s is used by route r, 1 = yes, 0 = no
δ_r^e	Route r driven by engine type e, 1 = yes, 0 = no

$$\text{minimize } Z = \sum c_r x_r \tag{4.2}$$

$$\text{s.t.,} \sum_r \alpha_r^d x_r = 1, \forall d \in D \tag{4.3}$$

$$\sum_r \beta_r^n x_r = 0, \forall n \in N \tag{4.4}$$

$$\sum_r \gamma_r^s x_r \leq 1, \forall s \in S \tag{4.5}$$

$$\sum_r \delta_r^e x_r \leq w_e, \forall e \in E \tag{4.6}$$

$$x_r \in \{0; 1\}, \forall r \in R \tag{4.7}$$

In (4.3) we make sure that all demands are serviced by the engines. To ensure that there are the necessary balance between the starting and ending stations of the engines we have the balance constraint (4.4). This constraint is also balancing the dummy stations such that an engine ending at a dummy station force an engine to start at the same dummy station. In (4.5) we ensure that each time-slot is used at most once. Engine availability is ensured by (4.6), where we constrain the number of each engine type used. Finally the binary constraint (4.6) force each routing to be used at most once.

To generate columns in the sub-problem we need to find columns with negative reduced costs. To do this we have to calculate the reduced costs, for this we use the duals for the master problem

to alter the costs in the sub-problem such that when minimizing the sub-problem it will result in the the column with the lowest reduced cost. We define the duals for each constraint set in the master-problem such that $\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ represent the vectors of dual variables defined by constraints 4.3, 4.4, 4.5 and 4.6 respectively, e.g. $\boldsymbol{\pi} = [\pi_d], \forall d \in D$.

4.4.2 Sub-problem

In the sub problem we handle one weekly routing of an engine through the network. The problem is to chose a set of demands to form a route. To solve the routing we use Dynamic programming. As cost for the jobs we use the original costs adjusted with the duals from the master-problem. The total cost in the sub-problem is then the reduced cost for the corresponding column in the master-problem.

The sub-problem will be solved using dynamic programming, where we are able to solve this constrained shortest path problem (Bertsekas, 2005). A drawback of the dynamic programming approach to the shortest path problem is that there is no guarantee that we do not visit the same demand twice. This is called a cycle, a 1-cycle where we visit the same demand twice immediately after each other are avoided in the standard algorithm. A 2-cycle is where we visit an other demand in-between servicing the same demand twice, a k-cycle denominates such a cycle with a length k.

Since we have time-windows wherein the service of a demand has to be started, we can be sure that the highest k is limited. The shorter the time-window the smaller the worst case k-cycle length becomes. When avoiding k-cycles the runtime of the algorithm can deteriorate. Houck et al. (1980) shows how to avoid 2-cycles without deteriorating the asymptotic worst case running time.

In section 4.4.2.1 we present a standard dynamic programming algorithm, in section 4.4.2.2 we extend it to handle 2-cycles. For solving the sub-problem we will first use the standard algorithm and if cycles occur we will apply the 2-cycle elimination, in this way we should be able to speed up the runtime. Because the existence of such is limited in the case study, cases where 2-cycle elimination is not enough, a branching can be applied on the duration of the time-windows.

For the sub-problem we have defined parameters as shown in table 4.2. The objective in the sub-problem is to minimize total cost. If we denote $y_{tij} \in \{0; 1\}$ where $y_{tij} = 1$ when demand j is serviced after demand i at time t, and $y_{tij} = 0$ otherwise. We also define $z_e \in \{0; 1\}$ as $z_e = 1$ when engine type e is used and $z_e = 0$ otherwise. Then we can define the objective function as in equation 4.8.

$$c_r = \sum_{e \in E} \upsilon_e z_e + \sum_{t \in T} \sum_{j \in D} \left(\psi_{tj} \sum_{i \in D} y_{tij} + \sum_{i \in D} \phi_{tij} y_{tij} \right) \tag{4.8}$$

As the purpose with the sub-problem main is to get the reduced cost for a column, we have to associate the dual values with the original costs. To do this we modify the original cost with the dual costs, this modification for e.g. υ_e is defined as $\hat{\upsilon}_e$, the same applies to the other costs. To arrive at the modified cost we set $\hat{\upsilon}_e = \upsilon_e - \tau_e$ to be the altered cost of υ_e, thus when we

Table 4.2: Parameters

Parameter	Description
T	Set of all time periods t in the model, T covers one week.
Δ	Resolution of time-periods in minutes, $\Delta = 1$ gives $\|T\| = 10080$
g_{tij}	Transition time from demand $i \in D$ to $j \in D$ at time t
b_{tj}	Whether the time-window for the given demand $j \in D$ allows for start of service at time t
o_j	Origin station for demand $j \in D$
a_j	Arrival station for demand $j \in D$
ϕ_{tij}	Transit cost from demand $i \in D$ to $j \in D$ at time t
υ_e	Cost of using engine type e
ψ_{tj}	Cost of demand $j \in D$ at time t

minimize using the altered costs in the algorithm we get the column with the lowest reduced cost.

To set $\hat{\psi}_{tj}$ we have to tie σ_s to the time of t where the demand $j \in D$ is used. Hence we have to define which time-slots are used by ψ_{tj}, this can be done by defining $S_{dt} \subset S$ where S_{dt} contains the time-slots s which are used by job d at time t, if it is not possible to start the job at the given time $S_{dt} = \emptyset$. Now we can set $\hat{\psi}_{tj} = \psi_{tj} - \pi_j - \sum_{s \in S_{jt}} \sigma_s$.

Finally we have to handle the starting / ending stations and the duals hereof $\boldsymbol{\rho}$. As we can see in table 4.1, ρ_n represent the cost to use a station n as origin station, $-\rho_n$ as destination, $\rho_n - \rho_n$ as both or 0 as neither. Hence $\boldsymbol{\rho}$ can have different interpretations depending on it being negative or positive. This could be counterintuitive, but it makes sense in the way that for the master-problem it does the same difference for the balance whether we add a station as a origin station or if we remove the same station as arrival station. As removing the origin station will in fact free the corresponding arrival station and thus we do in fact add a arrival station by removing a origin station. This argument also hold for the opposite situation, and thus we the negative interpretation of $\boldsymbol{\rho}$ is the cost of using a station as arrival as this in turn force us to add the corresponding origin station. In this way the cost of using a station will be 0 if we use the same station as origin and arrival station.

Initially there are no cost for $\boldsymbol{\rho}$ as they do not carry a cost and do not interfere with any existing cost parameters. But we need $\boldsymbol{\rho}$ when calculating the reduced cost, thus they are introduced to the sub-problem.

The new modified objective function can now be formulated as shown in equation 4.9. These functions serves to find Hence we find the this first time period by

$$\begin{aligned} \bar{c_r} = & \sum_{e \in E} \hat{\upsilon}_e z_e + \sum_{t \in T} \sum_{j \in D} \left(\hat{\psi}_{tj} \sum_{i \in D} y_{tij} + \sum_{i \in D} \hat{\phi}_{tij} y_{tij} \right) \\ & + \sum_{j | y_{(\arg\min_{t | \sum_{j \in D} y_{t0j} = 1} \{t\}) 0j} = 1} \rho_{o_j} - \sum_{j | y_{(\arg\max_{t | \sum_{i \in D} y_{ti0} = 1} \{t\}) i0} = 1} \rho_{a_j} \end{aligned} \tag{4.9}$$

In addition to using the modified costs, we have introduced the costs for using a station as origin station and arrival station. For the origin station, the first term with $\boldsymbol{\rho}$, we find the origin station o_j by finding the first demand of the week. Then we take the appropriate cost ρ_n for using this station n as origin station. We find the first time-period t where we start service for any demand by $\arg\min_{t|\sum_{j\in D} y_{t0j}=1}\{t\}$. We do this by finding the first time-period t where we have $y_{t0j} = 1$ for any j, we use $i = 0$ as we have use 0 to represent the source and sink station. For the last time-period we reverse it so we have $j = 0$ and maximize t instead to find the last time-period with a demand start in order to find the last arrival station using a_j for the last demand j.

4.4.2.1 Dynamic programming

In this section we will present the standard dynamic programming algorithm for the sub-problem. This algorithm will be used first in any case, if it results in cycles we will initiate the 2-cycle elimination extension, that will be presented in section 4.4.2.2.

There are different engine types and these are only allowed to handle a subset of the demands due to different requirements. Therefore we define the subset of demands $D_e \subset D$ which is the set of demands that can be handled by a single engine type. Hence it is necessary to repeat the algorithm once for each engine type $e \in E$ and find the one with the lowest reduced cost. The dynamic programming algorithm can now be defined as follows in algorithm 1, with the parameters and variables shown in table 4.2 and 4.3.

Table 4.3: Variables

Variable	Description
f_{tj}	Lowest Cost to reach job $j \in D$ at time t
p_{tj}	Predecessor for f_{tj} in job j at time t

In order to solve the sub-problem, algorithm 1 starts with setting the cost for being in demand j at time 0. For most demands the time-window for time 0 is closed, thus we set the cost for demand j at time-period 0, equal to the cost of demand j at the time-period where the demand can be serviced, which is in the first time-period t where $b_{tj} = 1$.

To ensure that ρ_{o_j}, the cost of starting at demand j, is only counted once we also add it at the initialization. In this way we can handle the costs of origin stations ρ_{o_j} at first and avoid including them in later stages of the algorithm. We then deduct ρ_{a_j} to ensure that the cost of this station possibly being the final arrival station is considered, if we only service 1 demand it will be the case. Finally we add the cost of using the engine being considered v_e.

Now we iterate over all time-periods, note that the first $t = 1$, for each of these we iterate over all demands D. If a time-window is not open for a given demand, we set the cost of fulfilling this demand in the specific period equal to the last period, the same applies to the predecessor.

If a time-window is open we set the lowest cost of reaching the demand in the time-period f_{tj}, equal to the cheapest of either the previous period or moving from another demand to this. In the latter case we use the lowest cost to reach the demand when we move from at time t minus

Algorithm 1 Dynamic program for sub-problem

initialize $f_{0j} = \hat{\psi}_{(\arg\min_{t|b_{tj}=1}\{t\})j} + \rho_{o_j} - \rho_{a_j} + \hat{v_e}$, $p_{0j} = j$, $\forall j \in D_e$.
for $t = 1$ **to** $|T|$ **do**
  for $j \in D_e$ **do**
    if $b_{tj} \neq 1$ **then**
      $f_{tj} = f_{(t-1)j}$, $p_{tj} = p_{(t-1)j}$
    else
      $f_{tj} = \min\{\min_{i \neq j}\{f_{(t-g_{ij})i} + \hat{\phi_{tij}} + \hat{\psi_{tj}} + \rho_{a_i} - \rho_{a_j}\}, f_{(t-1)j}\}$
      if $f_{tj} = f_{(t-1)j}$ **then**
        $p_{tj} = p_{(t-1)j}$
      else
        $p_{tj} = \arg\min_{i \neq j}\{f_{(t-g_{ij})i} + \hat{\phi_{tij}} + \hat{\psi_{tj}} + \rho_{a_i} - \rho_{a_j}\}$
      end if
    end if
  end for
end for

the travel time between the two jobs g_{tij}, then we add the cost of this transit ϕ_{tij} and the cost of the demand we are moving to ψ_{tj}. To make sure that the cost for the previous final arrival station is removed, we add ρ_{a_i} and deduct the cost for the new final arrival station ρ_{a_j}. Then we set the predecessor to either the same as the previous if we did not move from another demand, or we set it to the demand we moved from if that was the case.

Upon completion algorithm 1 returns two matrices from which we can get the column with the lowest reduced costs. This is obtained by finding the last job with $\arg\min_j\{f_{|T|j}\}$ and then backtrack using the predecessor variable $p_{|T|j}$.

Algorithm 1 does not consider the risk of cycles between the same demands, as we are not interested in covering the same demand more than once i.e. at different times during the time-window. The algorithm currently avoids servicing a demand in a 1-cycle, as it cannot follow a demand with the same demand. To avoid case where we visit another demand in between a repetition of the same demand, we introduce the 2-cycle elimination.

4.4.2.2 Dynamic programming with 2-cycle elimination

Algorithm 2 allows us to avoid 2-cycles, it uses a new set of matrices that also record an alternative path through the demands, thus we introduce the new variables in table 4.4 that replace the previous variables in table 4.3. In addition to the variables f_{tj} and p_{tj} we add the second lowest cost f^2_{tj} of reaching a demand at a given time and the predecessor hereof p^2_{tj}.

This allows the dynamic programming to replace the lowest cost with the second lowest in the case where the lowest cost would form a 2-cycle. In this way we still consider all options that does not form a 2-cycle, whereas if we had just ignored f_{tj}, we would not have considered that the second best option could have involved this demand. This consideration of the alternative option for every demand is introduced by the function seen in equation (4.11) which determines

whether the predecessor of the previous move is allowed v.r.t. the 2-cycle elimination and then choose the appropriate option.

Table 4.4: Variables - Extended to handle 2-cycles

Variable	Description
f_{tj}	Lowest Cost to reach job $j \in D$ at time t
f^2_{tj}	Second lowest cost to reach job j at time t
p_{tj}	Predecessor for f_{tj} in job j at time t
p^2_{tj}	Predecessor for f^2_{tj} in job j at time t

$$f(tij) = \begin{cases} f_{(t-g_{tij})i} & \text{if } j \neq p_{(t-g_{tij})i} \\ f^2_{(t-g_{tij})i} & \text{otherwise} \end{cases} \tag{4.10}$$

$$\tag{4.11}$$

Algorithm 2 Dynamic program for sub-problem, with 2-cycle elimination

initialize $f_{0j} = \hat{\psi}_{(\arg\min_{t|b_{tj}=1}\{t\})j} + \rho_{o_j} - \rho_{a_j} + \hat{v}_e$, $p_{0j} = j$, $f^2_{0j} = \infty$, $p^2_{0j} = 0$, $\forall j \in D_e$.
for $t = 1$ **to** $|T|$ **do**
  for $j \in D_e$ **do**
    if $b_{tj} \neq 1$ **then**
      $f_{tj} = f_{(t-1)j}$, $p_{tj} = p_{(t-1)j}$, $f^2_{tj} = f^2_{(t-1)j}$, $p^2_{tj} = p^2_{(t-1)j}$
    else
      $f_{tj} = \min\{\min_{i \neq j}\{f(tij) + \hat{\phi}_{ij} + \hat{\psi}_{tj} + \rho_{a_i} - \rho_{a_j}\}, f_{(t-1)j}\}$
      if $f_{tj} = f_{(t-1)j}$ **then**
        $p_{tj} = p_{(t-1)j}$
      else
        $p_{tj} = \arg\min_{i \neq j}\{f(tij) + \hat{\phi}_{ij} + \hat{\psi}_{tj} + \rho_{a_i} - \rho_{a_j}\}$
      end if
      $f^2_{tj} = \min_{i \neq j, p_{tj}}\{f(tij) + \hat{\phi}_{ij} + \hat{\psi}_{tj} + \rho_{a_i} - \rho_{a_j}\}$
      $p^2_{tj} = \arg\min_{i \neq j, p_{tj}}\{f(tij) + \hat{\phi}_{ij} + \hat{\psi}_{tj} + \rho_{a_i} - \rho_{a_j}\}$
      if $p_{tj} \neq p_{(t-1)j}$ **and** $f_{(t-1)j} \leq f^2_{tj}$ **then**
        $f^2_{tj} = f_{(t-1)j}$, $p^2_{tj} = p_{(t-1)j}$
      else if $p_{tj} = p_{(t-1)j}$ **and** $f^2_{(t-1)j} \leq f^2_{tj}$ **then**
        $f^2_{tj} = f^2_{(t-1)j}$, $p^2_{tj} = p^2_{(t-1)j}$
      end if
    end if
  end for
end for

The initialization follows the previous version with the addition of $f^2_{0j} = \infty$, $p^2_{0j} = 0$, this is to ensure that the alternative is not used before it is an alternative to the lowest cost solution

for the demand at a given time. In line 5 we keep everything unchanged if there is no open time-window for the given job. Then we set the lowest cost for reaching the demand on line 7 the addition here is the usage of the function (4.11) $f(tij)$. When finding the lowest cost for this demand it checks whether the predecessor of the previous demand that we are moving from, is the same as the current demand, if it is the alternative predecessor is used. Lines 8-12 are unchanged and determines the primary predecessor.

From line 13 to 19 we determine the alternative lowest cost and its predecessor. First they are both set to the case where we do not consider staying from the demand we are currently in, notice that we do not consider the primary predecessor. This is to ensure that we do not pick the predecessor that resulted in the lowest cost. Earlier we copied the previous predecessors when staying in the same job or the time-window was closed. In this way we can ensure that the alternative lowest cost is a real alternative and it just moved to the current demand at a later time-period from the same predecessor.

After setting the initial values for arriving from other demands, we consider two scenarios that could alter these values; if the primary predecessor changes i.e. that we updated f_{tj} moving from a new demand and $f_{(t-1)j}$ is lower than the alternative we already found, line 13. The second scenario considered is; the primary predecessor remains unchanged and $f^2_{(t-1)j}$ is lower than the new alternative set on line 13. In the case of scenario 1 we set the new alternative solution equal to the previous primary solution, the same applies to the predecessor, for scenario 2 the previous alternative solution and predecessor are kept unchanged in the new time-period.

Upon completion we find the solution as in algorithm 1 with the exception that, we when back-tracking has to consider whether an alternative predecessor was used, this can be done by reversing the 2-cycle condition.

4.4.3 Branching

To get integer solutions with our solution approach presented in figure 4.5 we need to develop a branching technique. We wish to impose a branching approach where we focus on matching constraints. Such an approach is presented by Ryan and Foster (1981), where they identify constraints that should be served together, Vance et al. (1994) use a similar branching approach. This could e.g. be that we say that an engine serving one demand should also serve an other specific demand. Also Anbil et al. (1992) suggest a "follow on", immediate predecessor, such that we do not only say that two demands should be served on the same route but should be served immediately after each other.

4.5 Implementation

In this section we present implementation related issues tied to the DBSRS case to which we have applied the model presented in section 4.4. It will be coed i c++, using the ILOG CPLEX callable library.

4.5.1 Heuristics

In the sub-problem and the dynamic programming algorithm presented in section 4.4.2.1 and 4.4.2.2, we solve the sub-problem in a time resolution of 1 minute $\Delta = 1$ such that we have 10080 time-periods per week. In the dynamic programming we solve the sub-problem in a matrix that has n rows one for each time-period and m columns one for each demand. We should be able to reduce the runtime of the algorithm by introducing a heuristic approach to the sub-problem, where we round the time-periods to an other resolution e.g. $\Delta = 2$, $\Delta = 5$ or $\Delta = 10$. As there are normally one time-slot per half hour it would normally not result in two time-windows being merged.

When rounding the time-periods we do a round up to ensure that we do not create infeasible columns but we in worst case disregard some options and create less than optimal columns. The rounding of time-periods are done according to equation 4.12.

$$\hat{t} = \left\lceil \frac{t}{\Delta} \right\rceil \tag{4.12}$$

The new time period $\hat{t}$ is equal to the original time-period t divided by the resolution of the new time-periods Δ. When rouding up time-periods in the interval $\{1; 5\} \rightarrow 1$. We apply the same rounding to the duration of any transit times such that they will be rounded up as well. When translating the solution back to the original problem, we find if something was planned in period $\hat{t} = 5$ and $\Delta = 5$, that a time-slot must be found in a time-period $t \in \{\Delta(\hat{t} - 1) + 1; \Delta\hat{t}\}$. We are then guaranteed that the columns when translated back to the original problem that we have a feasible solution to the sub-problem, but we are not guaranteed optimality. Hence we have to solve the sub-problem with the exact approach at least once to conclude that we cannot get a column with negative reduced costs.

4.6 Additional issues

In this section we present issues that still needs to be addressed in the final version of this paper.

(i) Cycle elimination, is it enough in the case to avoid 2-cycles or should we implement, if not could it be enough with a simple method, as it is used seldom or should we design something more sophisticated.

(ii) Catalogue paths vs. tailormade paths; is it enough just to consider the catalogue paths or do we need to implement a method to consider extra paths as in Kuo et al. (2010).

(iii) Uncertainty in time-slots actually allocated; as stated we are not certain that we will get the exact time-slot we apply for in the end. Is it a valid assumption that this can be disregarded.

(iv) Branching methods are not completely developed and we should consider alternative methods.

Chapter 5

Proposal II: Crew Planning Problem and an overall integration of the planning process at a freight railway operator

5.1 Background

In this paper we seek to solve the Crew Planning Problem(CPP) as described by Caprara et al. (2007), for a case at DB Schenker Rail Scandinavia (DBSRS), in a joint context where we seek to integrate its two sub-problems; the Crew Scheduling Problem and Crew Rostering Problem. Furthermore we investigate whether we can improve the overall planning at the railway operator, by considering the CPP at earlier stages of the planning process.

The planning process at railway operators are described with different levels of detail by, among others, Lusby et al. (2011) and Caprara et al. (2007). In general the planning process can be described in 5 levels as follows:

- Line Planning
- Timetabling
- Train Routing / Rolling Stock Circulation
- Crew Scheduling
- Crew Rostering

The line planning problem is the design of the lines to be operated, a line is a origin and destination station with a set of intermediate stations. The frequency of the line, is how often the line is operated. Hence this problem is related to the network and mainly demand driven. In the timetabling problem we design a timetable for the desired lines and fix departure and arrival times, also we allocate time-slots in the network to secure a feasible timetable. Before we

in the Train Routing Problem assign engines and rolling stock to the lines in accordance with the timetable. We ensure that this allocation gives a feasible plan and often the objective would also be to minimize cost.

The purpose of the Crew Scheduling Problem is to design duties based on trips given by the timetable, a set of work regulations govern the creation of these duties. In the Crew Rostering Problem we assign these duties to rosters, for which a set of work regulations also applies.

In *Proposal I* (**?**) we joined the Timetabling and Train Routing problems and applied it to a case at DBSRS, in this proposal we refer to that problem as problem A. Now we wish to investigate to what extend it is possible and advantageous to consider the CPP when developing the timetable and train rosters, as we do in problem A. Because problem A gives the set of trips to be handled by the CPP, there is a clear connection which is not considered if the problems are handled completely independent.

In this proposal, we will in section 5.2 give a problem description, in 5.3 an outline of the overall approach, in 5.3.1 we cover the CPP and an exact solution approach hereto, in 5.4 an approach to the integration of problem A, from proposal I, and the CPP, finally we present further issues that has to be addressed in 5.5.

5.2 Problem description

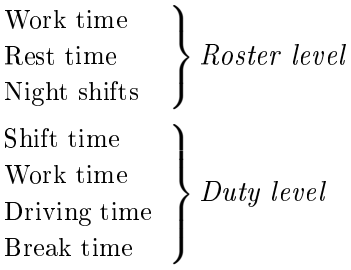
The CPP problem arises at a railway operator. In this proposal we adapt the general problem to a case from DBSRS. A particular difference is that we consider crew employed in different countries, and thus with different sets of working regulations. The crew considered work in Sweden, Denmark, or Germany. The main constraints for the working regulations will be explained in this section. Other work can be found considering regulations related to these countries e.g. see Rezanova and Ryan (2010) for a Danish and Jütte et al. (2011) for a German setting.

Before describing the problem we give a few definitions of key concepts; a *trip* is the shortest segment of a route, driven by a train, that can be covered by a single crew member, this can also be referred to as a task. A *shift* is a collection of trips and other tasks, such as breaks, that last for one work engagement. A *duty* on the other hand can be a shift or multiple shifts if the shifts do not return to the home depot at the end of a shift. A *roster* is a combination of one or more shifts that extents for the crew planing horizon, working regulations not handled in building the duties are handled here. In figure 5.1 we give an overview of what regulations that are handled in the different level.

In figure 5.1 we show the different states the drivers can be in and where they apply to the duty generation and rostering process. Working time is the time during a shift where we are engaged with a task. For working time there are shift specific requirements, as there is a maximum work time per shift. The work time is also limited on a weekly and monthly basis.

Shift time is regulated by a maximum that are greater than or equal to the working time. Here we also have a minimum on the shift time, as there is a minimum time the employees can expect to work when called to work. Shift time is also in some countries regulated by a monthly maximum.

Figure 5.1: Work regulations



The driving time is regulated by a maximum during the shift, this again depends on whether it is a day or night shift. Also there is a maximum on continues driving time. The last state the driver can be in during a shift is break time, here there are minimum regulations on the length of the breaks. The length of breaks are dependent on the amount of driving in the shift, the length of the shift and has to be placed within a certain time in the shift.

A shift can be deemed to be a night shift, of which the amount of are constrained. Night shifts are defined by different means in the different countries, but mainly it is defined by the shift involving a time of day which is considered to be night time.

An engine driver has a base station where he or she must start and end a shift if layovers are to be avoided. Layovers are associated with specific costs depending on nationality of the driver and the location where the layover takes place. In practice drivers can stay away from the home-base a number of times during a month at an additional cost.

In the duties there are some tasks that have to be carried out. At the beginning of a duty an engine driver has 15 minutes to do some preparation. Likewise a driver has 10 minutes to finalize a duty, meaning that the duty ends earliest 10 minutes after the driver has arrived at his or her base station. In addition to the preparation and finalization time, it takes some time for an engine driver to go from the room where they start/end the duties and to/from the tracks where the engine is, this is dependent on the station being used. This time also applies when having a break.

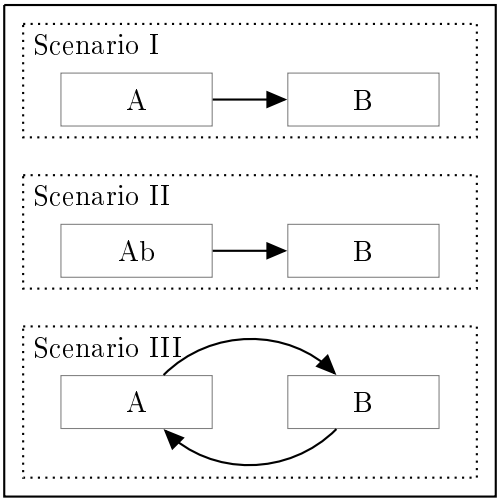
The objective is to minimize the overall cost of using; tracks, engines and crew. Pertaining to the crew mainly described in this proposal, we consider minimizing costs through the design of the optimal rosters which adheres to the case specific constraints.

5.3 Possible overall solution approaches

In this section we present two alternative approaches to integrate the planning process described in section 5.1. In figure 5.2 and this section we illustrate and present these overall solutions approaches.

If we consider figure 5.2 we see that three blocks are presented; A, B and Ab. A is the solution

Figure 5.2: Solution approaches



Note: *A: Engine Planning, B: Crew Planning, Ab: Engine Planning with input from B*

approach to the joint Engine and Timetabling problem presented in proposal I. B is the solution approach for the Crew Planning Problem, that will be presented in section 5.3.1. Ab is a version of A, where we try to consider the cost that will be incurred by using crew in problem B.

Scenario I is a classic sequential approach, where the Timetabling and Crew Scheduling are performed completely disjoint. Thus we are not particularly constrained in our solution approach and modeling of problem B. Scenario II has the same sequential approach but here we adjust problem A with cost information from problem B. Which is done such that we to some extend consider problem B when solving problem A. Again there are no special requirements to the method applied to problem B.

Scenario III separates itself from the other scenarios in the way that it contains a stronger link between problem A, which is kept unchanged, and problem B. Here the idea is to solve A, then B and afterwards start an iterative approach. When iterating between A and B, we pass dual variables that corresponds to adjusting the timetable. These dual variables are then used to adjust the prices in problem A. To get dual variables that are meaningful we need to solve problem B with an exact method, thus Scenario III entails that we apply an exact solution method to the Crew Planning Problem.

In section 5.3.1 we present the Crew Planning Problem and solution methods pertaining hereto and in section 5.4, we describe these scenarios in greater depth.

5.3.1 Problem B: Crew Planning Problem

The motivation for the Crew Planning Problem, which is sometimes referred to as the Crew Management Problem (Caprara et al., 1997; Ernst et al., 2004), is that solving the Crew Scheduling Problem and Crew Rostering Problem sequentially does not adequately describe the full prob-

lem. Here there is a degree of sub-optimization when we generate the duties before we generate the rosters (Caprara et al., 2007). The motivation for integrating these two problems in the CPP are presented in among others (Caprara et al., 2001; Ernst et al., 2001).

The integration of the two problems can be to different degrees, thus in the following we present first the Crew Scheduling Problem (CSP) and then the Crew Rostering Problem (CRP). Before we suggest how to integrate and solve them.

In Crew Scheduling Problem we generate duties, also referred to as Crew Pairings (Alfieri et al., 2007) and sometimes shifts, these duties are a set of tasks to be performed by a crew member stationed at a given depot. Hence we first define a set of depots Q, the tasks to be covered are trips which is the shortest part of a demand route that can be covered by a single crew member. Therefore a demand that is driven at a given time breaks down to a set of trips that together covers the demand, all trips are described in the set P.

A duty consist of both trips and other tasks that has to be performed, such as breaks, dead-heading, preparation, finalization, walking to and from trains. The total length of the duties can vary, usually they are restricted to a single day, but can be of longer length if it is necessary to stay overnight at another depot. In general for train services duty lengths are shorter than for instance the airline sector, as we often have more dense networks and it is therefore easier to return the crew members to their home depot at the end of the shifts.

Depending on the degree of integration and other factors the objective of the CSP can be quite different, often it is to get the lowest cost of the duties or to reduce the crew necessary to perform the duties. Where we have to cover all trips while adhering to all working regulations governing the crew.

The duties generated in the CSP are then passed to the Crew Rostering Problem (Caprara and Toth, 1998) where we combine the duties to general crew rosters, these rosters are again specific for a crew depot but not necessarily on a single crew level.

In the CRP we now consider a longer planning horizon than in the CSP. Here we also have some constraints that govern the forming of rosters, but there are in general fewer. The duties now have features given by the trips and other tasks the duty contain. We use this information to ensure that we comply with the regulations for the rosters.

When presenting the CSP and CRP it is obvious that there has to be some link between the two problems. If we could formulate and solve it as a joint CPP it would be favorable as we would then design the duties to form the optimal rosters, and not form the duties to reduce cost, staff or average work-time assigned. In the following section 5.3.1.1 we present exact solution methods for the CSP and CRP. Whereas we have not yet given a joint formulation.

5.3.1.1 Exact approach

In this section we give the sequential formulation of the CPP. The CSP is often formulated as a Set Partitioning Problem (SPP) or variation hereof, the Set Covering Problem (SCP) is also widely used in the literature.

We formulate the CSP as a SPP with extra constraints in equation (5.1-5.4) with the parameters given in table 5.1. Where y_r is a binary variable indicating whether a duty r is used or not.

Table 5.1: Parameters

Parameter	Description
R	Set of all duties (columns) under consideration, indexed by r
P	Set of all trips, indexed by p
Q	Set of all Depots, indexed by q
c_r	Cost of duty r
α_r^p	Trip p is covered by duty r, 1 = yes, 0 = no
β_r^q	Depot d is home depot for duty r, 1 = yes, 0 = no
w_q	Maximum work that can be allocated to depot q

$$\text{minimize } Z = \sum_{r \in R} c_r y_r \tag{5.1}$$

$$\text{s.t.}, \sum_{r \in R} \alpha_r^p y_r = 1, \forall p \in P \tag{5.2}$$

$$\sum_{r \in R} \beta_r^q y_r \leq w_q, \forall q \in Q \tag{5.3}$$

$$y_r \in \{0; 1\}, \forall r \in R \tag{5.4}$$

The CSP is sometimes formulated such that we generate all relevant duties in advance in a phase called duty generation (Caprara et al., 2007). Alternatively we can use a delayed column generation method where we have equation (5.1-5.4) as a Restricted Master Problem and generate duties ad-hoc in a Sub-problem. In this way we do not have to generate all duties in advance, but we do instead have to pass dual variables to the sub-problem and solve it efficiently.

For the CRP we can make a similar SPP formulation such that we allocate the duties from the CSP to individual crew-members over a longer time horizon, could be 4 weeks. Due to the duties being tied to the crew bases in the CSP, we can decompose the CRP on a crew base level. This gives the SPP defined by equation (5.5-5.7), where x_k^u is equal 1 if a given roster k is covered by crew member u and 0 otherwise. The set U is the set of all crew-members and K is all roster under consideration, γ_k^r is whether a roster k covers duty r.

$$\text{minimize } Z = \sum_{k \in K} c_k x_k^u \tag{5.5}$$

$$\text{s.t.}, \sum_{u \in U} \sum_{k \in K} \gamma_k^r x_k^u = 1, \forall r \in R \tag{5.6}$$

$$x_k^u \in \{0; 1\}, \forall k \in K \tag{5.7}$$

Again we need a way to generate columns, either by predefining them or through delayed column generation.

A joint formulation or integrated approach for the CPP is necessary in order to have an overall iterative approach as described by scenario III in section 5.3.1. The joint approach can be formulated as in Ernst et al. (2001), but here it is still necessary to solve the CRP and CSP somewhat sequentially.

Though we have not formulated an exact joint approach for the CPP, we illustrate the integration with problem A, which we do in section 5.4, by using the CSP. The CPP (problem B) should be solved in a joint approach, whether it be a joint IP-formulation or an iterative or otherwise integrated approach.

5.4 Overall Integration Approach

In this section we present the three approaches to overall integration (see figure 5.1) between problem A and B. As both problem A and B are quite time consuming to solve, we have to consider the runtime of a joint approach. As described in section 5.2 the overall problem is executed well in advance of when we have to apply for railway tracks, and almost a year before we have to use the plan. Hence we can allow the joint runtime to be quite long e.g. up to a week, as there are no requirements to the max runtime we are not particularly constrained here.

The motivation for integration of the two problems is to reduce the total costs for the joint solution. Hence we seek to see whether an increase in cost or different solutions at the same cost for problem A, can influence problem B in such a way that the total costs are reduced more than we increase the cost of problem A. Thus we define C as the total costs for the integrated problem, C_A as the total costs for problem A and C_B for problem B. Hence we have the relationship between the costs shown in equation (5.8).

$$C = C_A + C_B \tag{5.8}$$

In section 5.3.1 we had set P for all trips and set D for all demands. We define $\widetilde{P}$ to be a set of trips that are fixed to a given time-period, we call these trips time-fixed-trips. We set $\widetilde{D}$ to be a set of demands fixed w.r.t. time, these are called time-fixed-demands. The sets are indexed by $\tilde{p}$ and $\tilde{d}$ respectively. To tie the time-fixed-trips in problem B to the time-fixed-demands from problem A, we let $\Lambda_{\tilde{d}}, \forall \tilde{d} \in \widetilde{D}$ be the set of time-fixed-trips corresponding to the time-fixed-demand $\tilde{d}$.

5.4.1 Scenario I

As scenario I is the most simple, and the most commonly used in the literature, we designate it as our benchmark scenario. The integration method here is straight forward as we first solve problem A, we define this solution as S_A, and then apply our method for problem B and finally get our solution for B, which we define as S_B. Combined they define the joint solution S. The joint solution is therefore defined by $S = S_A \cup S_B$.

In this scenario the information passed from A to B is purely the timetable, where the demands are split into trips, which is the shortest segment of work that can be allocated to the crew at a specific time-period. Based on this we solve B and get the total cost C for scenario I.

5.4.2 Scenario II

Scenario II focus on adjusting the cost of the overall demands in problem A. Instead of just focusing on the overall demands, we could also try to adjust the time-fixed-demands $\tilde{d}$. This adjustment in the cost for problem A will be the link to problem B. Hence the adjustment should somehow reflect the attractiveness or the cost implications a $\tilde{d}$ has on the crew plan. This could be that certain $\tilde{d}$ implies using corresponding number of $\tilde{p}$, that would possibly cause the following costs in problem B such as:

(i) The time-fixed-trip is prone to cause an additional night shift.

(ii) Overlap or come to close time-wise to other trips: Could say that we maximally want a certain amount of trips placed within a period of the week and impose a cost penalty on exceeding this.

(iii) Lack of or to short time to connect to the next departure when servicing a destination that has infrequent connections.

(iv) Others?

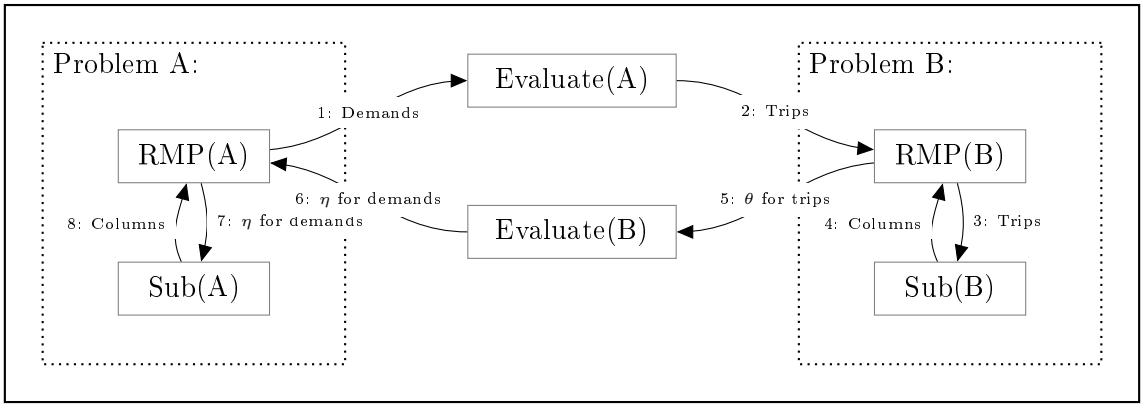
Commonly for anything to be included here, is that we should be able to calculate the cost implication before solving problem A, and adjust the costs for A in such a way that we can obtain a solution S_A and pass it to problem B, and solve it as described previously in scenario I.

5.4.3 Scenario III

This scenario is the most complicated as we strive to optimize both A and B to minimize C. The idea evolves around a joint solution S obtained as in scenario I. Then we try to pass information from S_B back to A in order to re-optimize A and obtain a new S_A to provide input for B. In figure 5.3 we illustrate these iterations.

Figure 5.3 shows the flow of information in the integration for scenario III. Starting with problem A; we send information regarding when demands are driven to $Evaluate(A)$. In $Evaluate(A)$ we check if we have met a stopping criterion. Then we split the demands into trips (remember, trips are the shortest segment of a demand, that can be assigned to crew, derived from a demand and as the demands here are fixed in time, so are the trips) and pass these to $RMP(B)$, the restricted master problem for B. Now we remove columns that have become infeasible due to change in the trips. We pass the new trips and duals as we normally do along to the sub-problem for B, $Sub(B)$. In $Sub(B)$ we then create sufficiently new columns and pass them to $RMP(B)$ that is re-solved until we meet the optimality condition, which is when no columns with negative reduced costs can be generated.

Figure 5.3: Scenario III



Note: *A: Engine Planning, B: Crew Planning.*
Assuming B is solved using delayed column generation as explained in section 5.3.1.1, and A is solved by the algorithm developed in proposal I

The duals for all trips are passed to $Evaluate(B)$ that combine the duals for trips to duals for demands and again evaluates the stopping criterion. As we need to pass cost information regarding all trips, and not just the ones chosen i S_A, we have to find a way to get duals for all possible trips. The duals for all possible trips are found in $RMP(B)$, the procedure herefore will be shown in section 5.4.3.1. We re-optimize $RMP(A)$ after adjusting the existing columns using the duals for demands, and iterations with the sub-problem as usual. We now need to explain how we get the duals for the trips not used in $RMP(B)$ and elaborate on the stopping criterion, this will be done in sections 5.4.3.1 and 5.4.3.2 respectively.

5.4.3.1 Duals from problem B

From the CSP formulated in equation (5.1-5.4) we extract constraint (5.2), or the corresponding constraint in a joint formulation of the CPP, and reformulate it to constraint (5.9) and add constraint (5.10).

Set $\widetilde{D}$ equal all time-fixed-demands possible in problem A and set $\widetilde{P}$ equal all possible time-fixed-trips corresponding hereto. Now let Ω be all time-fixed-trips corresponding to the time-fixed-demands not chosen in S_A. This implies that there is a set $\{\widetilde{P} \setminus \Omega\} \subset \widetilde{P}$, which are the time-fixed-trips chosen in S_A, which we currently service.

There are more than one time-fixed-trip in $\widetilde{P}$ that cover a trip $p \in P$. Thus let $\widetilde{P}_p \subset \widetilde{P}, \forall p \in P$, be the set of time-fixed-trips that cover a trip p. All the sets $\widetilde{P}_p$ are disjoint subsets of $\widetilde{P}$.

$$\sum_{r \in R} \sum_{\tilde{p} \in \widetilde{P}_p} \alpha_r^{\tilde{p}} y_r = 1, \forall p \in P \tag{5.9}$$

$$\sum_{r \in R} \alpha_r^{\tilde{p}} y_r = 0, \forall \tilde{p} \in \Omega \tag{5.10}$$

We extract the duals to $\boldsymbol{\theta} = [\theta_{\tilde{p}}], \forall \tilde{p} \in \widetilde{P}$. From (5.9) we get the duals for the time-fixed-trips not serviced; $\theta_{\tilde{p}}, \forall \tilde{p} \in \Omega$. The duals for the time-fixed-trips serviced; $\theta_{\tilde{p}}, \forall \tilde{p} \in \{\widetilde{P} \setminus \Omega\}$, is set to 0. As we have to cover the corresponding trip which is implied by equation (5.9), and (5.10) implies that all $\tilde{p}$ not being serviced are equal 0, we have that all $\tilde{p} \in \{\widetilde{P} \setminus \Omega\}$ should equal 1 in a feasible solution.

Let $\boldsymbol{\eta} = [\eta_{\tilde{d}}], \forall \tilde{d} \in \widetilde{D}$ be the dual vector for time-fixed-demands. Now combine the duals for the time-fixed-trips to the duals for the corresponding time-fixed-demands by equation (5.11).

$$\eta_{\tilde{d}} = \sum_{\tilde{p} \in \Lambda_{\tilde{d}}} \theta_{\tilde{p}}, \forall \tilde{d} \in \widetilde{D} \tag{5.11}$$

Finally we pass the vector $\boldsymbol{\eta}$ to problem A, where we adjust the cost for performing the time-fixed-demand $\tilde{d}$ accordingly.

5.4.3.2 Stopping criterion

When imposing a stopping criterion on scenario III, we have to consider that we, as previously described, can allow for a relatively long runtime. Let C_i be the overall costs C for an iteration i. An obvious stopping criterion is then if $S_{A_i} = S_{A_{i-1}}$,$S_{B_i} = S_{B_{i-1}}$, which shows no change in the solution to either A or B. Further stopping criteria can be considered:

(i) Total time limit

(ii) $C_i = C_{i-1}$

(iii) $\triangle_i \leq$ minimum change, $\triangle_i = C_i - C_{i-1}$

(iv) $S_A = S_{A_{i-1}}$

(v) Others?

Criteria (i) is necessary to introduce in order to risk any sort of endless loop. Stopping on criteria (ii) should in most cases be sufficient, unless the underlying solution changed and further iterations could lead to a lower C. Criteria (iii) is a general case of (ii), where $\triangle = 0$, here we stop if the increments in solutions become so small that no further iterations are deemed worthwhile. The last criteria (iv) is a simplification of the basic as it would just force termination before we recheck problem B as two identical S_A will lead to the same S_B in this case. Regarding (v), the stopping criteria should be evaluated during numerical experimentation.

5.5 Additional issues

Here we present some issues that have to be further addressed in proposal II:

(i) In 5.3.1.1 we outline an exact approach for the CSP and CRP, but not an integrated approach. It is the aim with this proposal to give an integrated approach and such we should develop a joint approach to CPP?

(ii) Will scenario III render an optimal solution if we were to allow for an unlimited runtime, and if so can we prove it?

(iii) Is it realistic to execute an overall model, can we expect to get improved results with the overall integration of the planning process?

The list is non-exhaustive and other ideas and modifications are appreciated.

Chapter 6

Proposal III: Intermodal freight transportation planning with time windows

6.1 Background

With the increased focus on sustainability and the general wish to reduce co^2 emissions there is a focus on moving heavy duty trucks away from the roads and onto railways (Cacchiani et al., 2010). A way to do this is to increase the attractiveness of the railways as they are sometimes to slow compared to road transport.

Freight forwarding companies have incoming demand that gives a time and place to pick up a container along with destination and deadline for delivery. The shipper have the opportunity to deliver directly with a vehicle or deliver by combining the vehicle transportation with an intermodal train service that performs a part of the transport.

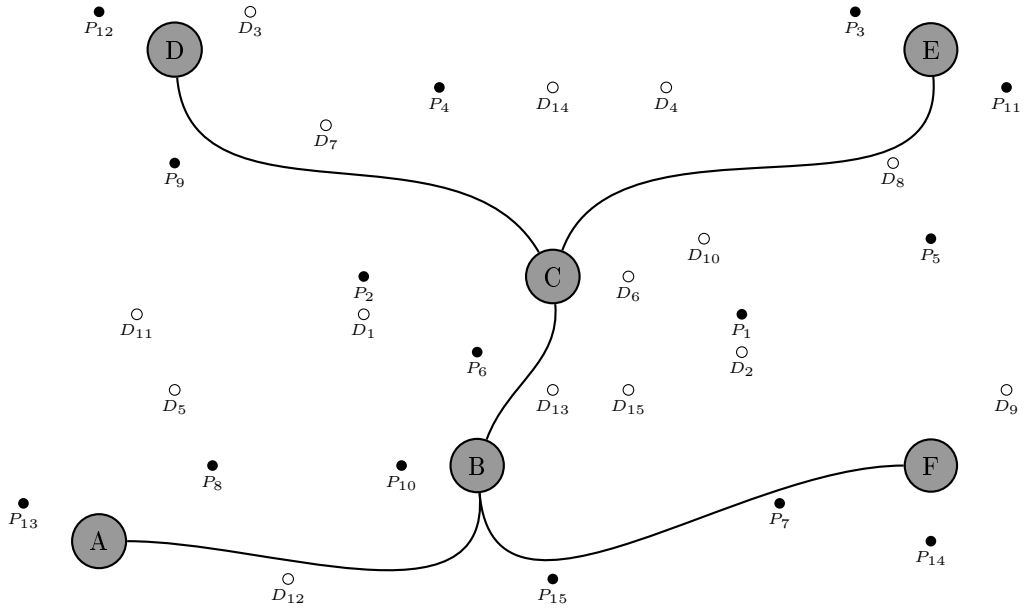
We consider a set of full-truck-load demands with pick up and delivery points and time windows for both. With a set of trucks we can serve these demands, which would be a one-to-one Pickup and Delivery Problem with Time Windows (PDPTW) as described by Cordeau et al. (2008). I propose to generalize the problem considered in that paper by including the option of using intermodal transportation options. This is generalized such that a demand can be picked up and delivered at an intermediate point, intermodal-station (IM-station), where it can be transported to another IM-station in the network according to a fixed timetable, where it is again picked up by a truck that handles the final transportation to the delivery point.

The objective is to reduce the total costs subject to capacity w.r.t. the number of vehicles and space on trains, time-window compliance and that we serve all demand. There are costs for using trains, driving vehicles, on/off loading and penalties for delays.

The horizon of the problem can either be short or long term. In the short term we could consider a real-time problem where we try to insert extra demand into an already existing solution. We

though wish to consider a long term or static horizon where we start from scratch and create a full plan for a given demand set.

Figure 6.1: Outline of stations and demands



Note: *Circles are stations in a railway network connected by tracks (lines), the stations are index A-F. Small black circles are the pick up points referred to as P_i and the small white circles are delivery point referred to as D_i, where $i = \{1; 15\}$ is the demand number.*

An example of a problem with intermediate IM-stations are shown in figure 6.1. Here we have a set of IM-stations node $A - F$ with are connected by railway. For each demand $i \in \{1; 15\}$ we have a pickup node P_i, a delivery node D_i and a time-window $[a_i; b_i]$, where a_i is the earliest pickup and b_i the latest delivery.

We wish to solve this problem using variable neighbourhood search, see Hansen et al. (2010) for a recent survey.

6.2 Additional issues

We here give a non-exhaustive list of issues that should be considered in connection with this proposal III:

(i) We would like to find out whether to include a depot structure, or if it is more realistic to assume that the vehicles do not return to a specific depot but instead cycles between demands.

(ii) Do there exist benchmark problems in the literature?

(iii) How to set the cost for delays, if we allow delays to occur?

Chapter 7

References

Abbink, E., L. Albino, T. Dollevoet, D. Huisman, J. Roussado, and R. Saldanha (2011). Solving large scale crew scheduling problems in practice. *Public Transport 3*(2), 149–164.

Alfieri, A., L. Kroon, and S. van de Velde (2007). Personnel scheduling in a complex logistic system: a railway application case. *Journal of Intelligent Manufacturing 18*(2), 223–232.

Anbil, R., R. Tanga, and E. L. Johnson (1992). A global approach to crew-pairing optimization. *IBM Syst.J. 31*(1), 71–78.

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control* (3 ed.), Volume 1. Belmont, Massachusetts: Athena Scientific.

Cacchiani, V., A. Caprara, and P. Toth (2008). A column generation approach to train timetabling on a corridor. *4OR: A Quarterly Journal of Operations Research 6*(2), 125–142.

Cacchiani, V., A. Caprara, and P. Toth (2010). Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological 44*(2), 215 – 231.

Caimi, G., M. Fuchsberger, M. Laumanns, and K. Schüpbach (2011). Periodic railway timetabling with event flexibility. *Networks 57*(1), 3–18.

Caprara, A., M. Fischetti, P. Toth, D. Vigo, and P. Guida (1997). Algorithms for railway crew management. *Mathematical Programming 79*(1), 125–141.

Caprara, A., L. Kroon, M. Monaci, M. Peeters, and P. Toth (2007). Passenger railway optimization. In C. Barnhart and G. Laporte (Eds.), *Transportation*, Volume 14 of *Handbooks in Operations Research and Management Science*, pp. 129–187. Amsterdam: Elsevier.

Caprara, A., M. Monaci, and P. Toth (2001). A global method for crew planning in railway applications. In S. Voß and J. Daduna (Eds.), *Computer-Aided Scheduling of Public Transport*, Volume 505 of *Lecture Notes in Economics and Mathematical Systems*, pp. 17–37. Springer.

Caprara, A. and P. Toth (1998). Modeling and solving the crew rostering problem. *Operations research 46*(6), 820–830.

Cordeau, J.-F., G. Laporte, and S. Ropke (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil (Eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges*, Volume 43, pp. 327–357. Springer US.

Cordeau, J.-F., P. Toth, and D. Vigo (1998). A survey of optimization models for train routing and scheduling. *Transportation Science 32*(4), 380–404.

Dell'Amico, M., G. Righini, and M. Salani (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science 40*(2), 235–247.

Ernst, A. T., H. Jiang, M. Krishnamoorthy, H. Nott, and D. Sier (2001). An integrated optimization model for train crew management. *Annals of Operations Research 108*(1), 211–224.

Ernst, A. T., H. Jiang, M. Krishnamoorthy, and D. Sier (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research 153*(1), 3–27.

Hansen, P., N. Mladenovic, and J. M. Pérez (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research 175*(1), 367–407.

Hasle, G., O. Kloster, and M. Smedsrud (2011). Experiments on the node, edge, and arc routing problem. Technical report, SINTEF.

Houck, D., J.-C. Picard, M. Queyranne, and R. Vemuganti (1980). The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *OPSEARCH 17*, 93–109.

Jütte, S., M. Albers, U. W. Thonemann, and K. Haase (2011). Optimizing railway crew scheduling at DB Schenker. *Interfaces 41*(2), 109–122.

Kokubugata, H., A. Moriyama, and H. Kawashima (2007). A practical solution using simulated annealing for general routing problems with nodes, edges, and arcs. In *Proceedings of the 2007 international conference on Engineering stochastic local search algorithms: designing, implementing and analyzing effective heuristics*, SLS'07, Berlin, Heidelberg, pp. 136–149. Springer-Verlag.

Kuo, A., E. Miller-Hooks, and H. S. Mahmassani (2010). Freight train scheduling with elastic demand. *Transportation Research Part E: Logistics and Transportation Review 46*(6), 1057 – 1070.

Lindner, T. and U. Zimmermann (2005). Cost optimal periodic train scheduling. *Mathematical Methods of Operations Research 62*(2), 281–295.

Lusby, R., J. Larsen, M. Ehrgott, and D. Ryan (2011). Railway track allocation: models and methods. *OR Spectrum 33*(4), 843–883.

Nahapetyan, A. and S. Lawphongpanich (2007). Discrete-time dynamic traffic assignment models with periodic planning horizon: system optimum. *Journal of Global Optimization 38*(1), 41–60.

Prins, C. and S. Bouchenoua (2004). A memetic algorithm solving the vrp, the carp and general routing problems with nodes, edges and arcs. In W. Hart, N. Krasnogor, and J. Smith (Eds.), *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, pp. 65–85. Springer.

Rezanova, N. J. and D. M. Ryan (2010). The train driver recovery problem - a set partitioning based model and solution method. *Computers & Operations Research 37*(5), 845–856.

Ryan, D. M. and B. A. Foster (1981). An integer programming approach to scheduling. In A. Wren (Ed.), *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, pp. 269 – 280. North-Holland.

Vance, P. H., C. Barnhart, E. L. Johnson, and G. L. Nemhauser (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications 3*, 111–130.

Wøhlk, S. (2006). New lower bound for the capacitated arc routing problem. *Computers and Operations Research 33*(12), 3458–3472.

Ziarati, K., F. Soumis, J. Desrosiers, S. Gélinas, and A. Saintonge (1997). Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research 97*(2), 281–292.

Zwaneveld, P. J., L. G. Kroon, H. E. Romeijn, M. Solomon, S. DauzÃ¨re-PÃ©rÃ¨s, S. P. M. Van Hoesel, and H. W. Ambergen (1996). Routing trains through railway stations: Model formulation and algorithms. *Transportation Science 30*(3), 181–194.

Zwaneveld, P. J., L. G. Kroon, and S. P. M. van Hoesel (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operational Research 128*(1), 14–33.

Lower and Upper Bounds for the Node, Edge, and Arc Routing Problem

Lukas Bach
Centre for OR Applications in Logistics
Dept. of Economics and Business
Aarhus University, Denmark
luba@asb.dk

Geir Hasle
SINTEF ICT
Dept. of Applied Mathematics
Oslo, Norway
Geir.Hasle@sintef.no

Sanne Wøhlk
Centre for OR Applications in Logistics
Dept. of Economics and Business
Aarhus University, Denmark
sanw@asb.dk

November 15, 2011

Abstract

The Node, Edge, and Arc Routing Problem (NEARP) was defined by Prins and Bouchenoua in 2004. They also proposed a memetic algorithm procedure and defined a set of test instances: the so-called CBMix benchmark. The NEARP generalizes the classical CVRP, the CARP, and the General Routing Problem. It captures important aspects of real-life routing problems that were not adequately modeled in previous VRP variants. Hence, its definition and investigation contribute to the development of rich VRPs. In this paper we present the first lower bound for the NEARP. It is a further development of lower bounds for the CARP. We also define two novel sets of test instances to complement the CBMix benchmark. The first is based on well-known CARP instances; the second consists of real life cases of newspaper delivery routing. We provide numerical results in the form of lower and best known upper bounds for all instances of all three benchmarks. For two of the instances, the gap is closed.

Keywords: VRP; Node Edge Arc Routing; Bounds; Benchmarks

1 Introduction

The Vehicle Routing Problem (VRP) [26, 18] is central to transportation management. VRP research is regarded as one of the great successes of OR, partly due to the fact that a tool industry has emerged and results have been disseminated and exploited in industry. The VRP, construed in a wide sense, is a family of problems. Since the first definition of the classical, capacitated VRP in 1959 [14], many generalizations have been studied in a systematic fashion. Typically, exact and approximative solution methods have been proposed and investigated for each VRP variant. Efficient procedures for generating good lower bounds are important, both to speed up exact methods, and as a benchmark for approximative methods such as metaheuristics.

There has been a tremendous increase in our ability to find exact and approximate solutions to VRP variants over the past half century. A few years ago, the best exact methods could consistently solve instances of the classical Capacitated VRP (CVRP) up to some 70 customers. Today, the number is around 100, see for instance [7]. Approximative methods such as heuristic column generation, matheuristics, and metaheuristics seem to provide high quality solutions in realistic times even for large size instances of complex VRP variants. As problems are regarded as being solved for practical purposes, researchers turn to new extensions and larger-size instances. This trend is enhanced by market pull from the tool industry and their end users. The somewhat imprecise term "rich VRP" has recently been introduced to denote variants that are close to capturing all essential aspects of some set of real-life routing problems. Generalizations of models in the literature are defined, exact and approximative methods are proposed and investigated, and lower bounds are developed.

In contrast with the CVRP where demand for service is located in the nodes of the graph, arc routing problems have been proposed to model the situation where demand is located on edges or arcs in a transportation network [15]. Of particular industrial relevance is the Capacitated Arc Routing Problem (CARP) defined by Golden and Wong in 1981 [19] and its generalizations, as the CARP has multiple vehicles with capacity.

Up until 2004, there was a dichotomy in the VRP literature between arc routing problems and node routing problems. Some cases are naturally modeled as arc routing problems because the demand is fundamentally defined on arcs or edges in a transportation network. Prime examples are street

sweeping, gritting, and snow clearing. However, the arc routing model has been advocated in the literature for problems where the demand is located in nodes, for instance distribution of subscription newspapers to households and municipal pickup of waste, particularly in urban areas. In real-life cases, there are often thousands or tens of thousands of points to be serviced along a subset of all road links in the area. Such cases are often formulated as CARPs, typically with a drastic reduction of problem size.

In their 2004 paper [25], Prins and Bouchenoua motivate and define the Node, Edge, and Arc Routing Problem (NEARP). They state that:

> Despite the success of metaheuristics for the VRP and the CARP, it is clear that these two problems cannot formalize the requirements of many real-world scenarios.

Their example is urban waste collection, where most demand may adequately be modeled on street segments, but there may also be demand located in points, for instance at supermarkets. Hence, they motivate a generalization of both the classical CVRP and the CARP. To this end, they define the NEARP, which can also be viewed as a capacitated extension of the General Routing Problem [24]. A memetic algorithm for the NEARP is proposed and investigated empirically on standard CVRP and CARP instances from the literature. The authors also create a NEARP benchmark consisting of 23 grid-based test cases, the so-called CBMix-instances, and provide experimental results for their proposed method.

We would like to enhance the motivation for the NEARP and further emphasize its high importance to practice. The arc routing model for node based demand cases such as subscription newspaper delivery is based on an underlying idea of abstraction. The assumption that all point based demands can be aggregated into edges or arcs may be crude in practice and lead to significant errors and plans that are unnecessarily costly even in urban cases. In industry, a route planning task may cover areas that have a mixture of urban, suburban, and rural parts where many demand points will be far apart and aggregation would impose unnecessary constraints on visit sequences. A more sophisticated practical approach to contain the comutational complexity resulting from a large number of demand points in industrial routing is aggregation of demand based on the underlying transportation network topology. Such aggregation procedures must also take capacity, time, and travel restrictions into consideration to avoid aggregation that would lead to impractical or low quality plans. In general, such procedures will produce a NEARP instance with a combination of demands on arcs, edges, and nodes.

There are clearly good reasons to remove the arc/node routing dichotomy and enable modeling of the continuum of node and arc routing problems that is needed for representational adequacy of real-life situations. The introduction of the NEARP was a significant step towards the goal of rich VRP. Despite its importance, studies of the NEARP following its introduction are almost non-existent in the literature. Kokubugata and Kawashima [22] study problems from city logistics, including the VRP with Time Windows and the NEARP. They propose a Simulated Annealing metaheuristic for solving these problems. Computational results for the CBMix instances of Prins and Bouchenoua are presented, with several improvements. In [21], Hasle et al. describe results from experiments on NEARP test instances using their rich VRP solver Spider [20, 2], also reporting new best known results.

For the CARP, there has been an academic tradition for developing combinatorial lower bounds. The majority of these are based on the construction of one or several matchings. The best such lower bound is the Multiple Cuts Node Duplication Lower Bound (MCNDLB), [27], with the extensions added in [4]. Two good lower bounds based on other strategies are the Hierarchical Relaxations Lower Bound, [5], and a Cutting Plane Algorithm, [8]. See [4] for an overview of CARP lower bounds and [28] for a recent survey on CARP in general.

Lower bounds have been developed for many VRP variants. Many of these are based on cutting planes. See [17] and [23] for state-of-the-art lower bounds for the VRP. Also for the General Routing Problem, there is a tradition of obtaining lower bounds through algorithms involving cutting planes. See [11], [12], and [13] for some of the best lower bound algorithms for this problem.

The main contribution of this paper is to provide the first (to the best of our knowledge) lower bound for the NEARP. This bound is inspired by the MCNDLB for CARP and its extensions. We also define two new sets of test instances that complement the grid based CBMix instances of Prins and Bouchenoua. The first set is called the BHW benchmark. It is based on 20 well-known CARP instances from the literature. The second is called the DI-NEARP benchmark, consisting of 24 instances inspired from real cases of newspaper delivery routing. For all test instances, we provide numerical results in the form of lower and best known upper bound.

The remainder of this paper is organized as follows. In Section 2, we formally state the Node, Edge, and Arc Routing Problem and in Section 3, we

describe our lower bound algorithm for the problem and argue its correctness. In Section 4, we present two new benchmarks for the NEARP, and in Section 5 we give computational results. Finally, in Section 6, we offer concluding remarks.

2 The Node, Edge, and Arc Routing Problem

The Node, Edge, and Arc Routing Problem (NEARP) is defined on a connected graph $G = (N, E, A)$, where N is the set of nodes, E is the set of undirected edges, and A is the set of directed arcs. Note that G is not necessarily simple, i.e. there may be more that one link connecting any two nodes. Let c_{ij} denote the non-negative traversal cost for $(i, j) \in E \cup A$, also known as deadheading cost. There traversal cost is zero for nodes. Let $N_R \subseteq N$ be the set of required nodes, and let q_i denote the demand and p_i the processing cost of node $i \in N_R$. Similarly, let E_R and A_R be the set of required edges and arcs, respectively, and let q_{ij} and p_{ij} denote the demand and processing cost of $(i, j) \in E_R \cup A_R$. The processing cost is the total cost that accrues when the required edge or arc is serviced.

A fleet of identical vehicles each with capacity Q is initially located in a special depot node, denoted node 1. It is assumed that the size of the fleet is unbounded.

The goal is to identify a number of tours for the vehicles such that 1) every node $i \in N_R$, every edge $(i, j) \in E_R$, and every arc $(i, j) \in A_R$ is serviced by exactly one vehicle, 2) the sum of demands serviced by each vehicle does not exceed Q, and 3) the total cost of the tours is minimized.

Please note that the total processing cost for all feasible solutions to a given NEARP is constant. Hence, we do not need to consider processing costs in our lower bound procedure. Also, the convention for reporting results on the CBMix benchmark is such that the constant sum of processing costs has been subtracted.

In the following we use $SPL(i, j)$ to denote the cost of a shortest path in G from i to j. Let $N' \subset N$ be a subset of the nodes. We define $\delta^-(N') = \{(i, j) \in E \cup A | i \in N \setminus N' \text{ and } j \in N'\}$ to be the set of links entering N' and $\delta^+(N') = \{(i, j) \in E \cup A | i \in N' \text{ and } j \in N \setminus N'\}$ to be the set of links leaving N'. Note that due to the existence of undirected edges, $\delta^-(N')$ and $\delta^+(N')$ are not necessarily disjoint. Finally, we define $\delta(N') = \delta^-(N') \cup \delta^+(N')$ to

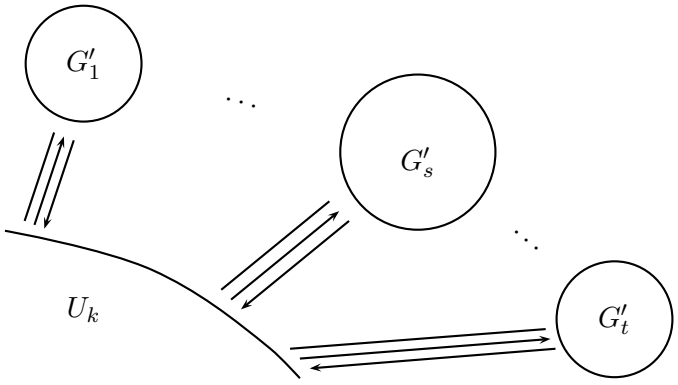


Figure 1: In each iteration, G is partitioned into U_k and a number of connected components, G'_s.

be the set of links connecting N' to the remaining graph. Finally, for any set of nodes, U, we use $G(U)$ to denote the graph induced by U.

3 Lower Bound for NEARP

The algorithm is a further development of the Multiple Cuts Node Duplication Lower Bound (MCNDLB) for the CARP, [27]. We give first an intuitive description of the structure of the algorithm and subsequently a formal description.

For notational reasons, in the description of the algorithm we will assume that the graph is simple, i.e. that there is at most one required link between any pair of nodes. We stress that the algorithm can easily be extended to the non-simple case.

Starting with $U_1 = \{1\}$, we consider mutually disjoint cuts $(U_k\,,\, N \setminus U_k)$ such that $U_1 \subset U_2 \subset \ldots \subset U_k \subset U_{k+1}$. For each such cut, U_k, the graph induced by $N \setminus U_k$ will consist of one or more connected components, $G'_s = (N'_s, E'_s, A'_s), s = 1, \ldots, t$, as illustrated in Figure 1. The number of vehicles needed to service the demand in G'_s and the demand of links connecting G'_s to U_k can be estimated by $p_s = \lceil (\sum_{i \in N'_s} q_i + \sum_{(i,j) \in E'_s \cup A'_s \cup \delta(N'_s)} q_{ij})/Q \rceil$.

Ideally, each vehicle would service the demand of an edge or arc when entering G'_s and when leaving G'_s. When this is not the case, we say that

the vehicle is using an artificial link. Such links can be either links without demand or links with demand not currently being serviced. We estimate the number of artificial links (entering arcs, leaving arcs, and undirected edges) needed for all vehicles to both enter and leave G'_s. With this, we can estimate the cost of servicing demand in G'_s and demand of links connecting G'_s to U_k by constructing a node duplicated network and letting m_s be the cost of a minimum cost perfect matching in this network. We do this for all the connected components and hence, $L = \sum_{s=1}^{t} m_s$ estimates the cost of servicing everything outside $G(U_k)$.

To estimate the cost of servicing demand in $G(U_k)$, we use the minimum cost $\overline{c}_s^u$ of a link between U and each component, G'_s and multiply this by the number of artificial links needed to connect the two: r_s^u. Iterating over all the mutually disjoint cuts and all the connected components of these, we can estimate the cost of servicing the demand in $G(U_k)$ as $L1 = \sum_{j=1}^{k-1} \sum_{s=1}^{t} \overline{c}_s^u r_s^u$.

For each of these cuts, $L+L1$ is a lower bound on the cost, and the algorithm selects the highest of these.

Note in the details of the algorithm that the calculations become more complex than outlined above due to the existence of both directed and undirected links.

3.1 The Lower Bound Algorithm

The MCNDLB for NEARP is outlined as follows.

1. Set $U = \{1\},\ L1 = 0,\ L = 0,\ L2 = 0.$
2. Let $N' = N \setminus U$ and G' be the graph induced by N'.
   Find the connected components of G'.
   Suppose that G' has t components: $G'_s = (N'_s, E'_s, A'_s)$, $1 \leq s \leq t$.
   - 2.1. For $s = 1$ to t do:
     - 2.1.1. Number of vehicles needed to service the demand of nodes, edges, and arcs in G'_s and $\delta(N'_s)$.
       $p_s = \lceil (\sum_{i \in N'_s} q_i + \sum_{(i,j) \in E'_s \cup A'_s \cup \delta(N'_s)} q_{ij})/Q \rceil$
     - 2.1.2. Number of required edges and arcs in cutset.
       $\psi_s^u = |\{(i,j) \in \delta(N'_s) \cap E_R\}|$

$$\psi_s^- = |\{(i,j) \in \delta^-(N'_s) \cap A_R\}|$$

$$\psi_s^+ = |\{(i,j) \in \delta^+(N'_s) \cap A_R\}|$$

2.1.3. Number of artificial edges and arcs needed in cutset.

$$r_s^- = \max\{0, p_s - (\psi_s^- + \psi_s^u)\}$$

$$r_s^+ = \max\{0, p_s - (\psi_s^+ + \psi_s^u)\}$$

$$r_s^u = \max\{0, 2p_s - (\psi_s^u + \psi_s^- + \psi_s^+ + r_s^- + r_s^+)\}$$

2.1.4. Minimum cost of edges and arcs in cutset.

$$\overline{c}_s^- = \min_{(i,j) \in \delta^-(N'_s)} c_{ij}$$

$$\overline{c}_s^+ = \min_{(i,j) \in \delta^+(N'_s)} c_{ij}$$

$$\overline{c}_s^u = \min_{(i,j) \in \delta(N'_s)} c_{ij}$$

2.1.5. Construct the Matching Network, G_s^M. (Further explained in Section 3.2.)

2.1.6. Set m_s = cost of a minimum cost perfect matching in G_s^M.

2.2. $L = \sum_{s=1}^{t} m_s$.

2.3. $L2 = \max\{L2\,,\, L + L1 + \sum_{(i,j) \in E_R \cup A_R} c_{ij}\}$.

2.4. $L1 = L1 + \sum_{s=1}^{t} (r_s^u \cdot \overline{c}_s^u + r_s^+ \cdot \overline{c}_s^+ + r_s^- \cdot \overline{c}_s^-)$.

3. Set $U' = \{i \in N : i$ is adjacent to a vertex in $U\}$.
   Set $U = U \cup U'$.

4. If $U \neq N$: go to Step 2, otherwise go to Step 5.

5. Set $MCNDLB = L2$. Stop.

To strengthen the quality of the bound, for each of the nodes of U' in step 3, the node is added to U and steps 2 through 2.3 is repeated, hereafter the node is once again removed from U. Step 2.4 is performed only after all nodes of U' has been examined. This can strengthen the quality of the bound in that part of the matching. This procedure has been proved valid for similar lower bound procedures in [4, 10]. When testing the addition of nodes from U' to U, the number of nodes added jointly as well as their combination influences the quality of the bound. Unfortunately, the best number and best combination cannot easily be predicted beforehand. We have chosen to add the nodes individually.

3.2 Construction of the Matching Network

For the construction of the matching network $G_s^M = (N_s^M, E_s^M)$ in Step 2.1.5, we let the node set N_s^M consist of three disjoint sets, S, T, and X, where S will consist of copies of nodes from N_s', T will consist of copies of nodes in U, and X can be considered to be extra copies of nodes in U and will be added later.

Consider the s'th component, represented by the graph $G_s' = (N_s', E_s', A_s')$. For every node i in N_s' set $m^-(i) = \min_{u \in U} SPL(u, i)$ and similarly $m^+(i) = \min_{u \in U} SPL(i, u)$, i.e., $m^-(i)$ and $m^+(i)$ is the length of a shortest path from any node in U to i and from i to any node in U, respectively.

Furthermore, consider the degree information of nodes $i \in N$: Let $D^-(R, i) = |\{(j, i) \in A_R\}|$ be the number of required arcs entering node i, $D^+(R, i) = |\{(i, j) \in A_R\}|$ the number of required arcs leaving node i, let $D^u(R, i) = |\{(i, j) \in E_R\}|$ be the number of required edges incident to node i and let $D(R, i) = D^-(R, i) + D^+(R, i) + D^u(R, i)$ be the total number of required edges and arcs incident to node i.

For each node i in N_s', we add $D(R, i)$ nodes to S and call these nodes the family of i, denoted by $\chi(i)$. We say that the nodes in $\chi(i)$ are copies of i. Given a node j in $\chi(i)$, we refer to i as the origin of j, denoted by $\omega(j)$. We partition S into three disjoint subsets γ^-, γ^+ and γ^u. For each node i in N_s', we consider the family $\chi(i)$ consisting of $D(R, i) = D^-(R, i) + D^+(R, i) + D^u(R, i)$ nodes. Of these, we associate $D^-(R, i)$ nodes with γ^-, $D^+(R, i)$ with γ^+, and $D^u(R, i)$ with γ^u.

Now, consider the nodes in $N_s' \cap N_R$ for which $D(R, i) = 0$, i.e. required nodes without incident required arcs or edges. Note that these nodes were not considered above. For each such node, i, we add two nodes to S and call these the family of i, denoted by $\chi(i)$. We add one of the nodes to γ^- and the other to γ^+. As above, we call these nodes copies of i and for a node j in $\chi(i)$, we say that i is the origin of j, denoted by $\omega(j)$.

The set T consists of $2p_s$ nodes which can be considered to be copies of nodes in U. Because we know the minimum number of artificial edges needed in U, we can partition T into four disjoint subsets τ^-, τ^+, τ^u and τ^R, where the values of r_s^-, r_s^+, and r_s^u determines the number of nodes in each of the first three subsets, respectively, and the remaining nodes are assigned to τ^R. Due to the definitions in Step 2.1.3 of the algorithm, the number of nodes in τ^R equals the total number of required links in the cutset $\delta(N_s')$.

We let G_s^M be a complete undirected graph. The demand of required arcs and edges in G'_s is assigned to edges in G_s^M as explained in the following. These assignments are done in such a way that no node in N_s^M is chosen more than once and no demand in G'_s is assigned more than once.

For each required edge, $(i,j) \in E'_s \cap E_R$ we choose a node $i' \in \chi(i) \cap \gamma^u$ and a node $j' \in \chi(j) \cap \gamma^u$ in N_s^M, and assign the demand, q_{ij}, of (i,j) to (i',j'). For each required arc, $(i \to j) \in A'_s \cap A_R$ we choose a node $i' \in \gamma^+(i)$ and a node $j' \in \gamma^-(j)$ in N_s^M, and assign the demand, q_{ij}, of $(i \to j)$ to (i',j').

For required edges (i,j) in $\delta(N'_s) \cap E_R$ (say without loss of generality that $i \in N'_s$ and $j \in U$) we choose a node $i' \in \chi(i) \cap \gamma^u$ in N_s^M and a node k in $T \cap \tau^R$ and assign the demand of (i,j) to the edge (i',k). For required arc (i,j) in $\delta^+(N'_s) \cap A_R$ we choose a node $i' \in \chi(i) \cap \gamma^+$ in N_s^M and a node k in $T \cap \tau^R$ and assign the demand of (i,j) to the edge (i',k). For required arc (i,j) in $\delta^-(N'_s) \cap A_R$ we choose a node $j' \in \chi(j) \cap \gamma^-$ in N_s^M and a node k in $T \cap \tau^R$ and assign the demand of (i,j) to the edge (k,j').

For every node i in $N'_s \cap N_R$ with $D(R,i) = 0$ we assign the demand q_i of the node to the edge $(\gamma^-(i), \gamma^+(i))$ in G'_s. All other edges in E_s^M have zero demand.

The cost of edges (i,j) in E_s^M are set to

$$c_{ij} = \begin{cases} \infty & \text{if } q_{ij} > 0 \\ & \\ 0 & \text{if } i,j \in S \text{ and } \omega(i) = \omega(j) \text{ and } i \in \gamma^- \text{ and } j \notin \gamma^- \\ 0 & \text{if } i,j \in S \text{ and } \omega(i) = \omega(j) \text{ and } i \in \gamma^+ \text{ and } j \notin \gamma^+ \\ 0 & \text{if } i,j \in S \text{ and } \omega(i) = \omega(j) \text{ and } i \in \gamma^u \\ & \\ \infty & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^- \text{ and } j \in \gamma^- \\ SPL(i,j) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^- \text{ and } j \notin \gamma^- \\ \infty & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^+ \text{ and } j \in \gamma^+ \\ SPL(j,i) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^+ \text{ and } j \notin \gamma^+ \\ SPL(j,i) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^u \text{ and } j \in \gamma^- \\ SPL(i,j) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^u \text{ and } j \in \gamma^+ \\ \min\{SPL(i,j), SPL(j,i)\} & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^u \text{ and } j \in \gamma^u \\ & \\ \infty & \text{if } i,j \in T \\ \infty & \text{if } i \in S \cap \gamma^+ \text{ and } j \in T \cap \tau^+ \\ m^-(i) & \text{if } i \in S \cap \gamma^+ \text{ and } j \in T \setminus \tau^+ \\ \infty & \text{if } i \in S \cap \gamma^- \text{ and } j \in T \cap \tau^- \\ m^+(i) & \text{if } i \in S \cap \gamma^- \text{ and } j \in T \setminus \tau^- \\ \min\{m^-(i), m^+(i)\} & \text{if } i \in S \cap \gamma^u \text{ and } j \in T \\ \infty & \text{if } i \in T \cap \tau^+ \text{ and } j \in S \cap \gamma^+ \\ m^-(j) & \text{if } i \in T \setminus \tau^+ \text{ and } j \in S \cap \gamma^+ \\ \infty & \text{if } i \in T \cap \tau^- \text{ and } j \in S \cap \gamma^- \\ m^+(j) & \text{if } i \in T \setminus \tau^- \text{ and } j \in S \cap \gamma^- \\ \min\{m^-(j), m^+(j)\} & \text{if } i \in T \text{ and } j \in S \cap \gamma^u \end{cases}$$

In order to tighten the bound, consider every pair of demand edges, (i,j) and (k,l) in E_s^M. If $q_{ij} + q_{kl} > Q$, we set the cost of the edges $(i,k),(i,l),(j,k)$, and (j,l) to ∞, since (i,j) and (k,l) cannot be serviced on the same tour. Next, remove all copies of nodes in $T \cap \tau^R$ and all edges incident to them.

To finalize the construction of G_s^M, let the number of nodes in X be determined by

$$|X| = \begin{cases} |S| - |T| & \text{if } |S| - |T| > 0 \\ 0 & \text{if } |S| - |T| \text{ is even} \\ 1 & \text{if } |S| - |T| \text{ is odd} \end{cases}$$

Each node j in X is connected by an edge to every node i in $S \cup X$ with a

cost given by

$$c_{ij} = \begin{cases} 0 & \text{if } i,j \in X \\ m^-(i) & \text{if } i \in \gamma^+ \text{ and } j \in X \\ m^+(i) & \text{if } i \in \gamma^- \text{ and } j \in X \\ \min\{m^-(i), m^+(i)\} & \text{if } i \in \gamma^u \text{ and } j \in X \end{cases}$$

Nodes in X are not connected to nodes in T. X can be considered to be extra copies of nodes in U. There are now enough nodes in $T \cup X$ for every node in S to be matched to one of these at cost $m^-(i)$ or $m^+(i)$. X is necessary because for any two nodes, i and j in S, it might be cheaper to match both i and j to something in U instead of matching them to each other, illustrating the vehicle driving back to subgraph U and then returning to S. Note that although the triangle inequality may not apply in the original graph, it does apply in this matching network as long as no edges with cost infinity are involved.

3.3 Correctness of the Lower Bound

Since the bound is an extension of the MCNDLB for the CARP, which was proved valid in [27], we focus on the changes that are made to the original algorithm.

The first change occurs in the calculation of p_s, i.e. the number of vehicles needed to service component s and the links connecting it to U in Step 2.1.1. In the original algorithm the demand was summarized over all demand edges. Because, in the NEARP, we have both required edges, arcs, and nodes, clearly the summation should be over all of these.

In Step 2.1.3, we calculate the number of artificial links needed. In the original algorithm, this was calculated as $r_s = \max\{0\,,\, 2p_s - q_s\}$. Needing at least p_s vehicles, each of which must both enter and leave the component, and having r_s required edges in the cut, this is clearly correct. For the NEARP, we will first consider entering vehicles. Note that we must have at least p_s of these. We have ψ_s^- entering arcs and ψ_s^u edges in the cut. Hence, up to $\psi_s^- + \psi_s^u$ vehicles can use these existing links and we need to construct $\max\{0\,,\, p_s - (\psi_s^- + \psi_s^u)\}$ artificial entering arcs. The argumentation for arcs leaving the component is symmetrical.

Needing at least $2p_s$ links in total, we can now add the number of undirected edges corresponding to the difference between $2p_s$ and the sum of all required

links (arcs and edges) and the number of artificial arcs added to the network. Thereby the correctness of Step 2.1.3 has been argued.

With these definitions in place, it follows directly that the estimate for servicing everything inside U, $L1$, in Step 2.4 is correctly generqalized to the NEARP.

Left is only to argue that the construction of the matching network in Step 2.1.5 leads to a valid estimate for servicing G'_s and the cutset. The structure of the matching network is similar to the one in the original bound. For each original node, we add $D(R, i)$ nodes to S in the matching network. This is exactly the number of times we must either enter or leave the node due to arc and edge requirements. Clearly, we may partition these into nodes representing entering, leaving, and undirected demand. We use the same number of nodes in the sets T and X as in the original algorithm, but again, for the set T, we can partition the nodes into sets based on the knowledge described above. For required nodes with no adjacent required links, it is clearly legal to add two nodes to S - one for entering and one for leaving.

The assignment of required edges is done precisely as in the original algorithm, except that now we take the direction of arcs into account when selecting the nodes in each family to be matched. Furthermore, for required nodes we legally select the edge between the two copies of the original node to absorb the demand. As in the original algorithm, the cost of all these demand-assigned edges is set to infinity to prevent them from being used in the matching.

The remaining cost structure is far more complex in this algorithm than in the original one. This is due to the partitioning of families into entering, leaving and undirected sets. When two nodes are in the same family, the cost of the edge connecting them is zero if it is possible to enter through one of the copies and leave through the other. Otherwise, the cost is set to infinity, to prevent this connection from being used in the matching.

For two nodes in different families, the cost is also infinity if either both nodes are entering nodes or both are leaving nodes, as this combination is illegal. If the combination is legal, the cost between such nodes corresponds to the cost of a shortest path between the origins of the nodes, while taking possibility of directions into account.

When considering a node i in S and a nodes j in T or in X, we use the different $m(i)$ estimates as in the original algorithm, except that again, we need to take into account the different combinations of entering and leaving,

making the expression less pretty. Connections internal to T and X are handled as in the original algorithm.

As can be concluded from the above argumentation, the algorithm presented in this paper is indeed a feasible lower bound for the NEARP.

4 New NEARP benchmarks

Only one set of test instances exists for the NEARP: the CBMix instances [25]. These instances are all based on graphs with a grid structure. To ensure more variation of the test platform for future algorithm developments and for testing the lower bound algorithm described above, we present two new benchmarks. The first is based on classical CARP instances from the literature, and the second set is based on real-life instances of an industrial application of the NEARP. We adopt the convention for reporting results on the CBMix benchmark, i.e., the constant sum of processing costs should be subtracted from the total cost.

4.1 The BHW instances

This benchmark is generated from benchmark instances for the CARP. Specifically, we have used a number of instances from the Gdb instances, [6], the Val instances, [9], and the Eglese instances, [16].

For each instance, we have kept the underlying graph structure, the existing demand, and the vehicle capacity. We have made the following modifications to the instances: Some undirected edges are replaced by directed arcs. If the edge was required, the demand is transmitted to the arc. Other undirected edges are replaced by two directed arcs, one in each direction. If the edge was required, the demand is either transferred to one of the arcs or both arcs are made required, each with a demand equal to the demand of the edge. Finally, some edges are left unchanged. Furthermore, some of the nodes are made required.

Table 1 gives the most important properties for each instance. The first column states the name of the instance and the second provides a reference to the underlying CARP instance. The next three columns give the total number of nodes, undirected edges, and directed arcs in the graph, whereas the following three columns give the same information for required entities. The

next column states the vehicle capacity. Note that the vehicles are assumed to be identical. The remaining six columns provide statistical information regarding required entities. Pairwise these columns provide the mean and standard deviation of the demand of the required nodes, edges, and arcs in the graph. Note that only the required entities are included in these calculations. All instances have relatively sparse networks, as they simulate real life situations. The depot is located in node 1 in all BHW instances. The instance definition files are found at SINTEF's TOP web site [3].

4.2 The DI-NEARP instances

This benchmark is taken from six real life cases from the design of carrier routes for home delivery of subscription newspapers and other media products in the Nordic countries. The company Distribution Innovation AS (DI) [1] operates a web based solution for design, revision, management and control of carrier routes. Route design and revision is based on electronic road and address data that are provided by commercial GIS vendors. Sophisticated models for travel and service time are utilized. The Spider VRP solver provided by SINTEF [2] is integrated in the solution.

The GIS road network data may have been improved by the user through manual editing due to errors or lack of detail. All delivery points are geocoded, and the enhanced road network data are transformed into an internal graph representation in Spider. The basic node routing problem cases typically have a large number of points. Through road topology based aggregation heuristics in Spider, the original problem has been transformed to a NEARP with side constraints. The graph topology of the instances is taken directly from the Spider graph.

Data for the six instances was retrieved from the DI web server in 2011. In these particular cases there are only required edges and nodes, no required arcs. The edges have symmetrical travel costs. All nodes have coordinates, but this is only for visualization purposes. The travel and service costs are set to the travel and service times calculated by the models in the DI system, as there is a close correlation between total route duration and route plan cost in reality. The index of the depot node is given explicitly.

The industrial problem does not have active capacity constraints, but there is a constraint on route duration. We have transformed the duration constraint to a capacity constraint and selected four different values for capacity

that produce a reasonable range for the number of routes that includes the number used in reality. Hence, the DI-NEARP benchmark consists of 24 instances. They are named DI-NEARP-n*abc*-Q*q*k, where *abc* is the total number of required nodes, edges, and arcs and q is capacity in thousands. Table 2 gives the most important properties for each instance. The structure of this table is similar to that of Table 1 except for the second column which is not relevant for the DI-NEARP instances. The instance definition files are found at SINTEF's TOP web site [3].

5 Computational Results

We have implemented the lower bound algorithm in two versions. A version where all nodes neighboring U are added at once, and a version where the addition of each node is tested separately before all nodes are added, as explained in Section 3.1. In this section, we give the results for both implementations, while referring to the latter as *AD1*. All lower bound calculations are performed on a PC with an Intel Core 2 Duo CPU, running at 2.53 GHz and with 2GB of RAM.

The results obtained for the three benchmark sets are given in Tables 3, 4, and 5. In each table, the second column provides the best known upper bound for the instance, hereafter referred to as UB^*. For the CBMix instances these are obtained from [25], [22], and [21]. For the BHW and the DI-NEARP instances the first upper bounds were obtained with the Spider solver [21]. Please remember that the constant sum of processing costs has been subtracted from the total cost.

For each of the two lower bound versions, we give the obtained value of the algorithm (LB), and the percentage gap from the of the best known upper bound to the lower bound, as calculated by the following formula:

$$\text{gap} = \frac{UB^* - LB}{(UB^* + LB)/2} \cdot 100$$

Finally, we provide the running time of the lower bound algorithm in seconds. We imposed a time limit of 9600 seconds. For the large-size DI-NEARP instances, the calculation of the AD1 LB was not completed within this time limit. Hence, the AD1 column has been omitted in Table 5.

For the CBMix benchmark, gaps vary between 3.0% and 39.5% with an average of 25.2%. The variation is larger for the BHW benchmark, where

the average and maximum gaps are 25.2% and 55.4%, respectively. The instances BHW4 and BHW6 have been solved to optimality. The average lower bound for the large size DI-NEARP benchmark instances is 27.0%, with a minimum of 4.2% and a maximum of 55.4%.

6 Concluding Remarks

The VRP literature has often been criticized for being based on idealized assumptions that render the proposed models inadequate for real life applications. In particular, there has been a strict separation of node routing and arc routing problems in the literature until 2004. In [25] Prins and Bouchenoua proposed the Node, Edge, and Arc Routing Problem (NEARP). They argued that there are real-life applications that can neither be adequately modeled as strict arc routing, nor as strict node routing problems.

In this paper, we have reinforced the claims of Prins and Bouchenoua and argued that the NEARP represents an important, new dimension of VRP model richness. We have also argued that the tradition of modeling applications such as newspaper delivery, mail delivery, and communal waste collection as arc routing problems is problematic. For real-life, large-size instances of such applications, where demand is basically located in nodes, abstraction techniques such as aggregation of demand may be needed to provide high quality solutions. Reasonable aggregation heuristics will typically produce instances with demand on nodes, edges, and arcs.

The main contribution of this paper is, to our knowledge, the first lower bound for the NEARP. Also, we provide two new sets of test instances: the BHW benchmark derived from 20 well-known CARP instances, and the DI-NEARP benchmark with 24 instances derived from real life data from carrier routing of subscription newspapers and other media products. These complement the grid based CBMix benchmark proposed by Prins and Bouchenoua, for which two other papers also provide upper bounds. For the BHW and DI-NEARP benchmarks the first upper bounds have been produced by Hasle et al. [21], so we now have lower and upper bounds for all test instances. For the instances BHW4 and BHW6, the gaps have been closed.

In our opinion, the NEARP is a theoretically interesting problem with high industrial relevance. We strongly encourage the research community to develop heuristic solution procedures as well as exact algorithms taking ad-

vantage of the structure of this important problem. Moreover, NEARP extensions should be proposed on the basis of important industrial aspects.

7 Acknowledgments

The authors would like to thank the company Distribution Innovation AS for giving us access to real-life case data from their newspaper delivery routing system. We would also like to thank Morten Smedsrud at SINTEF ICT for his assistance in extracting and processing the data for the DI-NEARP benchmark and for running the Spider solver on all three benchmarks.

The work presented here has been funded by the Research Council of Norway as a part of the Effekt project (contract number 187293/I40, SMARTRANS), and the DOMinant II project (contract number 205298/V30, eVita).

References

[1] Distribution innovation home page. `http://www.di.no/?lang=en`. Accessed: 19/10/2011.

[2] Spider web pages. `http://www.sintef.no/Projectweb/Transportation-planning/Software/Spider/`. Accessed: 19/10/2011.

[3] Top web pages. `http://www.sintef.no/TOP`. Accessed: 19/10/2011.

[4] Dino Ahr. *Contributions to multiple postmen problems*. PhD thesis, University of Heidelberg, 2004.

[5] Anita Amberg and Stefan Voss. A hierarchical relaxations lower bound for the capacitated arc routing problem. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 3, 2002.

[6] Edward K. Baker, James S. DeArmon, and Bruce L. Golden. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10(1):47–59, 1983.

[7] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set parti-

tioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[8] José M. Belenguer and Enrique Benavent. A cutting plane algorithm for the capacitated arc routing problem. *Computers and Operations Research*, 30(5):705–728, 2003.

[9] Enrique Benavent, Vicente Campos, Angel Corberán, and Enrique Mota. The capacitated arc routing problem: Lower bounds. *Networks*, 22:669–690, 1992.

[10] P. Breslin and A. Keane. The Capacitated Arc Routing Problem: Lower Bounds. Master's Thesis. *University College Dublin, Department of Management Information Systems*, 1997.

[11] Angel Corberán, Adam N. Letchford, and José M. Sanchis. A cutting plane algorithm for the general routing problem. *Mathematical Programming*, 90:291–316, 2001.

[12] Angel Corberán, Enrique Mota, and José M. Sanchis. A comparison of two different formulations for arc routing problems on mixed graphs. *Computers and Operations Research*, 33:3384–3402, 2006.

[13] Angel Corberán, Isaac Plana, and José M. Sanchis. A branch & cut algorithm for the windy general routing problem and special cases. *Networks*, 49:245–257, 2007.

[14] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 80(6), 1959.

[15] M. Dror. *Arc routing: theory, solutions, and applications*. Kluwer Academic, 2000.

[16] Richard W. Eglese and Leon Y.O. Li. An interactive algorithm for vehicle routeing for winter-gritting. *Journal of the Operational Research Society*, 47:217–228, 1996.

[17] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo L. Reis, and Renato Fonseca F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006.

[18] B.L. Golden, S. Raghavan, and E.A. Wasil. *The vehicle routing problem: Latest advances and new challenges*. Operations Research Computer Science Interfaces Series. Springer, 2010.

[19] Bruce L. Golden and Richard T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

[20] Geir Hasle and Oddvar Kloster. Industrial vehicle routing. In Geir Hasle, Knut-Andreas Lie, and Ewald Quak, editors, *Geometric Modelling, Numerical Simulation, and Optimization - Applied Mathematics at SINTEF*, pages 397–435. Springer, 2007.

[21] Geir Hasle, Oddvar Kloster, and Morten Smedsrud. Experiments on the node, edge, and arc routing problem. Technical report, SINTEF, 2011.

[22] Hisafumi Kokubugata, Ayako Moriyama, and Hironao Kawashima. A practical solution using simulated annealing for general routing problems with nodes, edges, and arcs. In *Proceedings of the 2007 international conference on Engineering stochastic local search algorithms: designing, implementing and analyzing effective heuristics*, SLS'07, pages 136–149, Berlin, Heidelberg, 2007. Springer-Verlag.

[23] Jens Lysgaard, Adam N. Letchford, and Richard W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[24] C.S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4:35–64, 1974.

[25] Christian Prins and Samir Bouchenoua. A memetic algorithm solving the vrp, the carp and general routing problems with nodes, edges and arcs. In W Hart, N Krasnogor, and J Smith, editors, *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, pages 65–85. Springer, 2004.

[26] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[27] Sanne Wøhlk. New lower bound for the capacitated arc routing problem. *Computers and Operations Research*, 33(12):3458–3472, 2006.

[28] Sanne Wøhlk. A decade of capacitated arc routing. In Bruce L. Golden, S. Raghavan, and Edward A. Wasil, editors, *The vehicle routing problem: Latest advances and new challenges*. Springer, 2008.

Instance	Basis Instance	Total number			Required			Capacity	Node demand		Edge demand		Arc demand	
		Nodes	Edges	Arcs	Nodes	Edges	Arcs		Mean	Std.	Mean	Std.	Mean	Std.
BHW1	GDB 1	12	11	22	7	11	11	5	1.0	0.0	1.0	0.0	1.0	0.0
BHW2	GDB 6	12	0	25	4	0	25	5	1.0	0.0	-	-	1.0	0.0
BHW3	GDB 12	13	8	30	5	8	7	35	7.2	3.5	8.0	4.1	9.0	2.3
BHW4	GDB 22	11	0	44	6	0	44	27	5.0	2.4	-	-	4.0	2.7
BHW5	Val 7a	40	0	132	30	0	132	200	8.5	4.3	-	-	8.0	3.3
BHW6	Val 7a	40	37	58	15	37	58	200	7.3	4.8	8.0	2.5	7.0	3.0
BHW7	Val 10d	50	0	194	35	0	194	75	9.1	4.6	-	-	7.0	2.3
BHW8	Val 10d	50	0	194	20	0	97	75	7.6	3.3	-	-	7.0	1.7
BHW9	Val 10d	50	26	142	10	26	142	75	8.6	3.1	6.0	1.6	7.0	2.3
BHW10	Egl e1 A	77	0	196	40	0	102	305	30.0	13.3	-	-	28.0	15.3
BHW11	Egl e1 A	77	0	196	20	0	51	305	28.3	11.1	-	-	28.0	10.8
BHW12	Egl s1 B	140	0	380	40	0	75	150	29.7	11.1	-	-	18.0	6.9
BHW13	Egl s1 B	140	0	380	25	0	150	150	29.0	11.2	-	-	18.0	9.7
BHW14	Egl e4 C	77	0	196	25	0	196	130	31.2	13.0	-	-	25.0	20.7
BHW15	Egl e4 C	77	0	196	30	0	98	130	33.0	11.9	-	-	25.0	14.6
BHW16	Egl s4 C	140	0	380	30	0	380	120	24.4	16.9	-	-	22.0	16.9
BHW17	Egl s4 C	140	0	380	50	0	190	120	22.3	7.1	-	-	22.0	11.9
BHW18	Egl e3 B	77	0	196	20	0	174	190	25.0	8.1	-	-	25.0	19.6
BHW19	Egl e3 B	77	0	196	20	0	87	190	24.1	9.0	-	-	25.0	13.8
BHW20	Egl s2 A	140	51	278	50	51	192	235	19.0	9.1	23.0	13.4	20.0	13.3

Table 1: Details on the new BHW instances.

Instance	Number of			Required			Capacity	Node demand		Edge demand		Arc demand	
	Nodes	Edges	Arcs	Nodes	Edges	Arcs		Mean	Std.	Mean	Std.	Mean	Std.
DI-NEARP-n240-Q2k	563	815	0	120	120	0	2000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n240-Q4k	563	815	0	120	120	0	4000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n240-Q8k	563	815	0	120	120	0	8000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n240-Q16k	563	815	0	120	120	0	16000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n422-Q2k	710	871	0	302	120	0	2000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n422-Q4k	710	871	0	302	120	0	4000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n422-Q8k	710	871	0	302	120	0	8000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n422-Q16k	710	871	0	302	120	0	16000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n442-Q2k	761	917	0	294	148	0	2000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n442-Q4k	761	917	0	294	148	0	4000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n442-Q8k	761	917	0	294	148	0	8000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n442-Q16k	761	917	0	294	148	0	16000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n477-Q2k	667	837	0	203	274	0	2000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n477-Q4k	667	837	0	203	274	0	4000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n477-Q8k	667	837	0	203	274	0	8000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n477-Q16k	667	837	0	203	274	0	16000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n699-Q2k	982	1103	0	335	364	0	2000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n699-Q4k	982	1103	0	335	364	0	4000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n699-Q8k	982	1103	0	335	364	0	8000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n699-Q16k	982	1103	0	335	364	0	16000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n833-Q2k	1120	1450	0	347	486	0	2000	31.9	72.7	104.2	216.9	-	-
DI-NEARP-n833-Q4k	1120	1450	0	347	486	0	2000	31.9	72.7	104.2	216.9	-	-
DI-NEARP-n833-Q8k	1120	1450	0	347	486	0	2000	31.9	72.7	104.2	216.9	-	-
DI-NEARP-n833-Q16k	1120	1450	0	347	486	0	2000	31.9	72.7	104.2	216.9	-	-

Table 2: Details on the DI-NEARP instances.

Instance	Best Known	Lower Bound			Lower Bound AD1		
	UB^*	LB	Gap	Runtime	LB	Gap	Runtime
CBMix1	2589	2409	7.2	1.0	2409	7.2	3.1
CBMix2	12190	9742	22.3	76.7	9742	22.3	353.4
CBMix3	3646	3014	19.0	7.5	3014	19.0	30.6
CBMix4	7583	5302	35.4	20.9	5302	35.4	118.8
CBMix5	4531	3747	18.9	3.8	3789	17.8	13.1
CBMix6	6970	4983	33.2	16.2	5201	29.1	43.1
CBMix7	9615	7296	27.4	58.7	7296	27.4	193.6
CBMix8	10524	7956	27.8	33.4	7956	27.8	196.8
CBMix9	4005	3460	14.6	2.5	3460	14.6	7.8
CBMix10	7582	6409	16.8	37.5	6432	16.4	113.0
CBMix11	4494	2998	39.9	4.6	3031	38.9	43.9
CBMix12	3235	3138	3.0	2.1	3138	3.0	12.9
CBMix13	9110	6489	33.6	19.4	6524	33.1	238.3
CBMix14	8553	5719	39.7	15.7	5731	39.5	107.5
CBMix15	8280	6270	27.6	10.9	6318	26.9	64.3
CBMix16	8886	7416	18.0	24.5	7416	18.0	172.6
CBMix17	4037	3654	10.0	1.8	3654	10.0	22.0
CBMix18	7098	6089	15.3	25.7	6089	15.3	120.9
CBMix19	16347	11065	38.5	110.5	11143	37.9	549.6
CBMix20	4844	3400	35.0	2.3	3452	33.6	15.7
CBMix21	18069	12474	36.6	61.8	12474	36.6	221.5
CBMix22	1941	1825	6.2	1.8	1825	6.2	4.8
CBMix23	780	667	15.6	0.1	667	15.6	0.8

Table 3: Results obtained for the CBMix instances.

Instance	Best Known	Lower Bound			Lower Bound AD1		
	UB^*	LB	Gap	Runtime	LB	Gap	Runtime
BHW1	337	324	3.9	0.3	324	3.9	1.3
BHW2	511	470	8.4	0.4	470	0.4	0.9
BHW3	426	326	26.6	0.2	326	26.6	0.5
BHW4	240	240	0.0	0.5	240	0.0	3.8
BHW5	514	498	3.2	5.4	502	2.4	52.1
BHW6	388	388	0.0	2.9	388	0.0	32.3
BHW7	1100	930	16.7	41.7	930	16.7	347.2
BHW8	673	644	4.4	6.8	644	4.4	118.8
BHW9	895	791	12.3	28.0	791	12.3	346.5
BHW10	8556	6810	22.7	21.6	6810	22.7	123.1
BHW11	5021	3986	23.0	6.9	3986	23.0	40.0
BHW12	10981	6346	53.5	33.4	6346	53.5	207.7
BHW13	14610	8746	50.2	86.3	8746	50.2	576.7
BHW14	25700	17762	36.5	113.0	17762	36.5	737.5
BHW15	15743	12193	25.4	20.7	12193	25.4	214.2
BHW16	45248	26014	54.0	787.2	26014	54.0	4905.3
BHW17	27195	15396	55.4	162.7	15396	55.4	900.6
BHW18	16042	11202	35.5	77.9	11202	35.5	435.3
BHW19	9470	7065	29.1	14.1	7080	28.9	101.6
BHW20	16930	10730	44.8	269.9	10730	44.8	1388.4

Table 4: Results obtained for the BHW instances.

Instance	Best Known	Lower Bound		
	UB^*	LB	Gap	Runtime
DI-NEARP-n240-Q2k	24417	16376	39.4	368
DI-NEARP-n240-Q4k	14984	14362	4.2	311
DI-NEARP-n240-Q8k	14984	13442	10.8	324
DI-NEARP-n240-Q16k	14873	13116	12.6	334
DI-NEARP-n422-Q2k	19095	11623	48.6	1571
DI-NEARP-n422-Q4k	15953	11284	34.3	1337
DI-NEARP-n422-Q8k	14523	11220	25.7	1049
DI-NEARP-n422-Q16k	14455	11198	25.4	1702
DI-NEARP-n442-Q2k	52393	35068	39.6	1689
DI-NEARP-n442-Q4k	45725	33585	30.6	1715
DI-NEARP-n442-Q8k	45725	32985	32.4	1736
DI-NEARP-n442-Q16k	42877	32713	26.9	1816
DI-NEARP-n477-Q2k	23272	19722	16.5	1572
DI-NEARP-n477-Q4k	20308	18031	11.9	1574
DI-NEARP-n477-Q8k	18652	17193	8.1	1582
DI-NEARP-n477-Q16k	18039	16873	6.7	1575
DI-NEARP-n699-Q2k	60233	34101	55.4	7249
DI-NEARP-n699-Q4k	40686	26891	40.8	6921
DI-NEARP-n699-Q6k	30797	23302	27.7	7133
DI-NEARP-n699-Q8k	27074	21967	20.8	7400
DI-NEARP-n833-Q2k	57476	32435	55.7	8239
DI-NEARP-n833-Q4k	42069	29381	35.5	8739
DI-NEARP-n833-Q8k	35331	28453	21.6	8675
DI-NEARP-n833-Q16k	33192	28233	16.1	8157

Table 5: Results obtained for the DI-NEARP instances.