A Model for Multiperiod Route Planning and a Tabu Search Method for Daily Log Truck Scheduling

Division of Optimization, Department of Mathematics
Linköping Institute of Technology

Christer Holm
Andreas Larsson

LiTH-MAT-EX-2004-01
Linköping 2004
A Model for Multiperiod Route Planning and a Tabu Search Method for Daily Log Truck Scheduling

Christers Holm, Andreas Larsson

The transportation cost of logs from forest to customers is a large part of the overall cost for the Swedish forestry industry. Finding good routes from harvesting points to saw and pulp mills is a complex task, where the total number of feasible routes is extremely high. In this thesis we present two methods for log truck scheduling.

The first is to, from a given set of routes, find the most valuable subset that fulfills the customers’ demand. We use a model that is similar to the set partitioning problem and a method that is referred to as a composite pricing coupled with Branch and Bound. The composite pricing based method prices the routes (columns) and chooses the most valuable ones that are then added to the LP relaxation. Once an LP optimum is found, the Branch and Bound method is used to find an integer optimum solution. We have tested this on a case of realistic size.

The second method is a tabu search heuristic. Here, the purpose is to create efficient and qualitative routes from a given number of trips (referred to as predefined trips). From a start solution tabu search systematically generates new solutions. This method was tested on a small problem and on a five times larger problem to study how the size of the problem affects the result. It was also tested and compared on two cases in which the backhauling possibilities (i.e. instead of traveling empty the truck picks up another load on the return trip) had and had not been studied. The composite pricing based method and the tabu search method proved to be very useful for this kind of scheduling.

Composite Pricing, Forestry, Log Truck Scheduling, Multiperiod Route Planning, Optimization, Predefined Trips, Tabu Search, Transportation
A Model for Multiperiod Route Planning and a Tabu Search Method for Daily Log Truck Scheduling

Division of Optimization, Department of Mathematics
Linköping Institute of Technology

Christer Holm
Andreas Larsson

LiTH-MAT-EX:2004-01

Supervisors: Myrna Palmgren, Division of Optimization, MAI, LiTH
Mikael Rönqvist, Division of Optimization, MAI, LiTH

Examiner: Mikael Rönqvist

Linköping Sweden 2004-01-27
Abstract

The transportation cost of logs from forest to customers is a large part of the overall cost for the Swedish forestry industry. Finding good routes from harvesting points to saw and pulp mills is a complex task, where the total number of feasible routes is extremely high. In this thesis we present two methods for log truck scheduling.

The first is to, from a given set of routes, find the most valuable subset that fulfils the customers demand. We use a model that is similar to the set partitioning problem and a method that is referred to as a composite pricing coupled with Branch and Bound. The composite pricing based method prices the routes (columns) and chooses the most valuable ones that are then added to the LP relaxation. Once an LP optimum is found, the Branch and Bound method is used to find an integer optimum solution. We have tested this on a case of realistic size.

The second method is a tabu search heuristic. Here, the purpose is to create efficient and qualitative routes from a given number of trips (referred to as predefined trips). From a start solution tabu search systematically generates new solutions. This method was tested on a small problem and on a five times larger problem to study how the size of the problem affected the result. It was also tested and compared on two cases in which the backhauling possibilities (i.e. instead of traveling empty the truck picks up another load on the return trip) had and had not been studied. The composite pricing based method and the tabu search method proved to be very useful for this kind of scheduling.
Acknowledgement

This thesis is our final project to in fulfilling the requirements for a Master’s degree in mathematics at Linköping Institute of Technology. The aim of this thesis is to utilize the knowledge gained during our studies in order to complete this work.

We would like to thank our supervisors Professor Mikael Rönnqvist andLicentiate Myrna Palmgren. They gave us excellent support during the whole project and helped us solve the problems that arose.

Furthermore, we would like to thank the members of the Division of Optimization for being helpful with various problems.

Finally, we would like to thank our opponents Jana Sochor and Cecilia Yu who read our thesis and gave us valuable critique and remarks.

Linköping, January 2004

Andreas & Christer
# Contents

Abstract ........................................ i

Acknowledgement ................................ iii

1 Introduction .................................. 1
   1.1 Forestry .................................. 1
   1.2 Forestry transportation ................. 1
       1.2.1 Transportation problem .......... 2
       1.2.2 Strategic, tactical and operative planning ... 3
       1.2.3 Scheduling and Despatching .... 3
   1.3 Our definitions ........................... 4
   1.4 Log Truck Scheduling .................. 4
   1.5 Transportation planning in the forestry industry today .... 5
   1.6 Purpose .................................. 6
   1.7 Method .................................. 6
   1.8 The structure of this report .......... 7

2 Mathematical Model for Multiperiod Route Planning .... 9
   2.1 Multiperiod log truck scheduling .... 9
       2.1.1 The first model [P1] .......... 9
       2.1.2 The second model [P2] ....... 10
       2.1.3 The third model [P3] .......... 11
       2.1.4 The final model [P4] .......... 13
3 Optimization Methods .................................................. 15
  3.1 Column generation ................................................. 15
  3.2 Composite pricing ............................................... 16
  3.3 Branch and Bound ................................................ 17
     3.3.1 Variable branching ..................................... 18
     3.3.2 Searching strategies ................................... 18

4 Our Method ............................................................ 19
  4.1 An approach based on composite pricing ...................... 19
  4.2 Initial problem ................................................... 19
     4.2.1 An example ............................................... 19
  4.3 An example of adding routes ................................... 20
  4.4 Algorithmic description ....................................... 21
  4.5 Addition of routes .............................................. 22

5 Results and Conclusions .............................................. 25
  5.1 The case ........................................................... 25
  5.2 Different tests and parameters ................................ 25
     5.2.1 A short time limit (15 minutes) ....................... 26
     5.2.2 A large time limit (60 minutes) ...................... 27
  5.3 Comparison and discussion ................................... 28
  5.4 Conclusion ........................................................ 28
  5.5 Further work ..................................................... 28

6 Daily Log Truck Scheduling with predefined trips ............... 29
  6.1 Predefined trips .................................................. 29
  6.2 The model .......................................................... 30

7 Optimization Methods ................................................ 33
  7.1 Local search methods ......................................... 33
     7.1.1 Tabu search ............................................... 33
## CONTENTS

8 **Our Heuristic** 37

8.1 Introduction of our heuristic ............................. 37
8.2 How to select routes ........................................... 38
8.3 How to change trips ............................................ 38
8.4 Summary ..................................................... 40
8.5 Criteria that affect the search direction .................... 40

9 **Results and Conclusions** 41

9.1 Data ...................................................... 41
9.2 Test cases .................................................. 42
9.2.1 Best versus Fast Change ................................. 43
9.2.2 Ordinary versus Random Tabu search ................. 43
9.2.3 Varying the parameter K and Tabu .................... 45
9.2.4 Larger case ............................................. 46
9.3 With Backhauling vs Without Backhauling ............... 47
9.4 Comparison and conclusion .................................. 47
9.5 Further work ................................................ 48
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Components involved in daily transportation planning</td>
<td>2</td>
</tr>
<tr>
<td>3.1</td>
<td>The column generation procedure</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>The structure of an unbalanced tree for the worst case with n variables</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Solution methodology for the Integrated Method</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Solution methodology for the Sequential Method</td>
<td>23</td>
</tr>
<tr>
<td>6.1</td>
<td>Illustration of backhauling</td>
<td>30</td>
</tr>
<tr>
<td>6.2</td>
<td>An illustration of a possible route where each node represents a predefined trip</td>
<td>30</td>
</tr>
<tr>
<td>7.1</td>
<td>An illustration of the tabu effect</td>
<td>34</td>
</tr>
<tr>
<td>7.2</td>
<td>A tabu and frequency matrix and solutions</td>
<td>34</td>
</tr>
<tr>
<td>8.1</td>
<td>An example of a route</td>
<td>37</td>
</tr>
<tr>
<td>8.2</td>
<td>An example of a single and double change and pivots</td>
<td>39</td>
</tr>
<tr>
<td>8.3</td>
<td>Methodology for Fast Change</td>
<td>39</td>
</tr>
<tr>
<td>8.4</td>
<td>Methodology for Best Change</td>
<td>39</td>
</tr>
<tr>
<td>9.1</td>
<td>The total cost as a function of time</td>
<td>44</td>
</tr>
<tr>
<td>9.2</td>
<td>The total cost as a function of iterations</td>
<td>44</td>
</tr>
<tr>
<td>9.3</td>
<td>Random Tabu search with Fast Change</td>
<td>46</td>
</tr>
</tbody>
</table>
## List of Tables

5.1 Results from the Integrated Method, 15 minutes time limit  
5.2 Results from the Sequential Method, 15 minutes time limit  
5.3 Results from the Integrated Method, 60 minutes time limit  
5.4 Results from the Sequential Method, 60 minutes time limit  

8.1 An example of an initial solution  

9.1 The Cases  
9.2 Initial solution for Case One  
9.3 Costs for the initial solutions  
9.4 Approximate CPU-times for making one change  
9.5 Ordinary Tabu search for Case One  
9.6 Random Tabu search for Case One  
9.7 Effects from different K and tabu for Ordinary Tabu search  
9.8 Solution to the marked example in Figure 9.7  
9.9 Random Tabu search for Case Two  
9.10 Random Tabu search with Fast Change
Chapter 1

Introduction

In this chapter we introduce the problem. We also present our purpose and methods.

1.1 Forestry

The Swedish forest has for a long time been an important source of material, work and money. Today the forestry sector generates about 100 000 jobs. The forest covers more than half of Sweden; 23 out of 41 million hectares are forest areas. 80% of Sweden’s wood products are exported to other countries and in 1999 this export generated 75 billion SEK for the forestry sector, which is nearly 13.5% of the total Swedish export. This makes the forest the country’s most important natural resource (see Ekelund & Liedholm, 2000).

The forestry industry today is more efficient than ever, mostly depending on new, more effective machines such as harvesters and forwarders. The harvester fells and bucks the trees and puts the logs into piles at the harvesting area. After that, the forwarder sorts and collects the piles and moves the logs to a larger supply point near a larger road so that reachable by the logging trucks. This transportation is called primary transportation. Since the competition in the market is increasing, there is still a large need for continued development. Some interesting problem areas, in which there is a great interest in optimization methods, are roll cutting at paper mills, road building, harvesting and transportation of products. When solving these planning problems, there are, besides the economic interest of the company, many aspects that must be taken into account. Examples are cultural and historical aspects, as well as how the environment is affected by different operations.

1.2 Forestry transportation

Today, approximately 25% of the land transports in Sweden is connected to the forestry. Most of it, more than 75%, is carried out by trucks and the rest by railway (Carlsson
and Rönneqvist, 1998). After that the primary transportation and the secondary transportation (i.e. the transportation of logs from forests to customers) are performed, most of the new products are transported to harbours and exported abroad, the rest is transported domestically. Problems related to the flow of wood are of great interest in the forestry sector, where one of the most interesting problem is the secondary transportation. Due to the huge costs for secondary transportation a small increase in efficiency would lead to large savings. To reduce the transport distance by 1% would lead to lower costs meaning that the estimated savings would then be about $10 million per annum (see Palmgren, 2001). In this thesis we focus on the secondary transportation.

![Image](image.png)

**Figure 1.1: Components involved in daily transportation planning**

### 1.2.1 Transportation problem

There are many different problems involved in planning transportation for the forestry sector. Some important factors that transport managers must take in account are described in this section.

There are two different types of trucks: with or without a loading arm. The truck with a loading arm can make pick-ups at any supply point without support, whereas the truck without a loading arm must be supported by a loader at the supply points. The trucks also differ in that the truck without a loading arm has a larger capacity.
1.2 Forestry transportation

The transportation can be carried out by trucks or trains (railcars). If trains are to be used, the logs must first be hauled by trucks from the forest to the railway. In cases where transport managers use railway and road carriers, the time windows and the specific regulations for these must be taken into account.

Supply points often have different assortments, e.g. spruce and pine. Some products can be stored at the supply points for longer periods than others since they are more or less sensitive to different weather conditions. The forestry companies lose value when the quality of the logs decreases, for instance when logs intended for saw mills are instead forced to be used for other less profitable purposes due to the quality degradation. Therefore lengthy storage in the forest is not acceptable.

The risk of queueing, i.e. the loading and unloading points visited by several trucks at the same time, must be considered.

The quality of the roads also affects the transportation planning and the shortest path is not always the most profitable one. Thawing of the ground in the spring and bad weather lead to closed roads. Therefore to be updated about the current circumstances in the area is of great importance. In 1994 the estimated cost for this problem was 750 million SEK and five years later the cost had increased to 900 million SEK (see Skogforsk-Nytt, nr 4, 1999).

1.2.2 Strategic, tactical and operative planning

In wood flow planning several operations are involved. Each operation is connected to a certain planning level: strategic, tactical or operative. Many problems that arise at these levels in the forestry industry are connected to the transportation of logs in one way or another. The strategic level is a long-term level, where the considered periods are from one to several years long. It is on the strategic level the forestry companies and the customers decide about future cooperation and on the basis of that the forestry companies must balance the expected demand against the supply. The planning periods for the tactical level are from one week to one year. On this level for instance decisions are made concerning which harvesting areas that will supply a certain industry. It is also on the tactical level that backhauling possibilities are investigated. The aim of using the backhauling strategy is to acquire more efficient routes. Read more about backhauling in Section 6.1. The operative level is a short-term level, where the considered periods are on a daily basis, i.e. from one day to one week long. Here primary- and secondary transportation planning (such as scheduling of individual logging trucks) are included.

1.2.3 Scheduling and Despatching

Scheduling and despatching are two approaches that can be used when solving transportation planning problems on an operative level. The aim of the first, scheduling, is to find efficient routes for one or several days. Each truck receive a schedule in advance for the current time period. The other method, despatching, successively builds up a route during the day. The driver just acquires one trip at a time. Here a trip consists
of one or several pick-ups followed by one delivery. The next trip is acquired after the first is accomplished. Despatching is valuable when something unpredictable happens, since it is easy to change the route. To make this system work, online communication between truck drivers and transport managers is necessary. In this thesis we focus on scheduling.

1.3 Our definitions

Following definitions are used in this thesis.

A supply point or harvesting point is the source point where the load is picked up. This is located in the neighbourhood of the harvesting area.

A customer (i.e. saw, paper and pulpmill) is the demand point where the load is delivered.

A load can consist of different assortments, where the assortments consist of different wood types and qualities.

A trip can be loaded or unloaded. The loaded trip consists of at least one pick-up and one delivery, whereas the unloaded trip is an empty transport between two destinations.

A predefined trip is a loaded trip from one harvesting point to one customer which must be carried out.

A route is a sequence of loaded and unloaded trips, starting and ending at the driver’s home location.

A schedule consists of a number of routes (one for each truck) for one day.

1.4 Log Truck Scheduling

The vehicle routing problem (VRP) is a known optimization problem. The problem has a large number of applications, post delivery and snow ploughing are only two examples of many. The problem consists of creating routes, one for each vehicle where each route must satisfies the following constraints:

1. Each customer must be visited exactly once.
2. The route starts and ends at the same depot.
3. The total demand for each route should not exceed the capacity of the vehicle.

In the vehicle routing problem with time windows (VRPTW) a time component is added to the VRP. The time window is represented with a time interval that must be respected i.e. the time where it is possible to visit the customer.
The pick-up and delivery problem with time windows (PDPTW) is similar to the VRPTW. In the PDPTW, pick-ups and deliveries are involved and therefore two additional constraints are added to the VRPTW. The first makes sure that a pick-up is carried out before delivering to the customer. The second assures that once a pick-up node is visited, the corresponding delivery node must be visited.

The log truck scheduling problem (LTSP) is closely related to PDPTW, but there are some differences. First, in the PDPTW, each destination can be visited only once, whereas the destinations in the LTSP can be visited several times by different vehicles. Second, in the LTSP there is also a decision to make concerning the size of the pick up or the delivery at a certain destination, whereas this size is fix for the PDPTW. LTSP is on the operative level. The aim of this problem is to create, at a minimal cost, a set of individual routes for each truck, where the following constraints are respected (see Palmgren, 2001).

1. The customers' demand must be fulfilled.
2. The supplies at the loading points are not exceeded.
3. The time windows for all destinations must be respected.
4. The capacity of the trucks may not be exceeded.
5. The customers receive the correct assortments.
6. A load is first picked up and then delivered.
7. All trucks have a home location, where the route begins and ends.

1.5 Transportation planning in the forestry industry today

Transportation planning is often a complex problem. Some difficulties and restrictions which complicate the planning are mentioned in Section 1.2.1. In countries such as Chile, Finland and New Zealand, planning systems have been developed. Several Chilean companies use a truck scheduling system called ASICAM. In Finland a system called EPO is used by one of the larger companies and in New Zealand a despatching system has been introduced that works in real time (see Carlsson & Rönqvist, 1998). In Sweden a system has been tested by Skogsäkarna for solving secondary transportation problems. It is a computer system based on optimization methods (see Palmgren, 2001). Even though there are several (well-working) systems, planning at the transportation companies is often more or less manually performed.

Today, forestry transportation planning in Sweden is in most cases decentralized. The planning is often done for each district within the company and then carried out independently of transports in other districts. This way of planning has its advantages where simplicity is one factor (e.g. that transport managers only focus on their districts). Due to that trucks are more or less localized in one area, the driver has a good overview of where harvesting has been carried out and can communicate directly at the loading points with the logging personnel to get information about when to start loading. Another positive effect is the local knowledge that the drivers have; they know about
the best roads etc. Local knowledge and the cooperation between drivers and logging personnel lead to less control by the forestry companies. The decentralized planning has also some drawbacks, the main is lack of coordination. To create backhauling routes or assign trucks to other areas would in several cases be valuable for the overall cost. One way to make this type of transportation planning more effective is to centralize the planning work and let the transport managers have access to all the necessary information. This would lead to an increased possibility for coordination. Today the centralized planning work is more interesting than before, since the computers have the capacity to solve the complex mathematical problems that arise for this kind of planning.

1.6 Purpose

This thesis’ purpose is to create individual truck schedules for two transportation problems using two different optimization methods. The first problem is to find a route for each truck (from a given set of routes) which fulfills the industries’ demand for a period of several days, at a minimal cost. The second problem is to create routes for one day, where a route is a sequence of loaded (predefined) and unloaded trips. Since the predefined trips are given and must be carried out, the objective is to minimize the cost of the unloaded part of the routes.

In order for our programs to be interesting for planning managers, we must keep the processing time down.

1.7 Method

In order to solve a problem using optimization methods, it must first be simplified (i.e. less detailed) to not get too large and complex and then modelled mathematically. With the model in mind, we develop and implement a method for solving the problem.

In our thesis we investigate two different log truck scheduling problem. Our purpose is to create schedules in advance, not in real time. Further we assume that all data is known.

- The first problem (multiperiod route planning) is the scheduling of trucks for several days, where a large number of the routes is given in advance. Here we differentiate between two procedures. First, we solve the problem integrated. In other words we focus on the whole period at once. Second, the problem is solved sequentially day by day, i.e. the problem is separately optimized for each day.

- The second problem (daily log truck scheduling with predefined trips) is to create routes for one day given a set of trips.

The multiperiod problem is formulated similar to a set partitioning typ problem. From this model we develop a method based on composite pricing coupled with Branch and
1.8 The structure of this report

Bound that is implemented in C++. We also use an LP solver (CPXprimopt) and a MIP (mixed integer programming) solver (CPXmipopt) from CPLEX 6.51.

For the daily log truck scheduling problem, we apply a heuristic method based on tabu search. From an initial schedule the heuristic makes changes of trips between the routes until a given stop criterion is reached. The implementation is done in C++.

1.8 The structure of this report

Here is a presentation of the report’s structure. Chapters 2 to chapter 5 focus on the multiperiod route planning problem and chapters 6 to chapter 9 focus on the daily log truck scheduling problem with predefined trips.

**Multiperiod Route Planning (MRP)**

**Chapter 2 - Mathematical Model for Multiperiod Route Planning.** In this chapter, a mathematical model for this problem is presented in four steps. **Chapter 3 - Optimization Methods.** In this chapter a general column generation procedure and a branch and bound procedure are described. **Chapter 4 - Our Method.** This chapter gives a detailed description of the algorithms that are used for solving the given optimization problem. A smaller example is also given. **Chapter 5 - Results and Conclusions.** Here we present the results, comparisons and conclusions.

**Daily Log Truck Scheduling with predefined trips (DLTS)**

**Chapter 6 - Daily Log Truck Scheduling with predefined trips.** This chapter includes a mathematical model for creating routes given predefined trips. **Chapter 7 - Optimization Methods.** Here we present a tabu search heuristic. **Chapter 8 - Our Heuristic.** This chapter describes our method used for solving the route planning problem, where the trips are fixed in advance. **Chapter 9 - Results and Conclusions.** This chapter includes results, comparisons and conclusions.
Chapter 2

Mathematical Model for Multiperiod Route Planning

In this chapter we present the mathematical model in four steps. The solving process of the multiperiod route planning problem is based on the final model.

2.1 Multiperiod log truck scheduling

In order to solve the multiperiod log truck scheduling problem we start by describing a mathematical model for the problem. We have chosen a mathematical model similar to the set partitioning formulation. A column in the model represents an entire route for a period of one day.

2.1.1 The first model [P1]

We start with a model showing the basic structure of the problem. Notations and variables are described as follow:

Decision variable:

\[ x_{ij} = \begin{cases} 
1 & \text{if truck } i \text{ drives route } j \\
0 & \text{otherwise} 
\end{cases} \]
Parameters:

\[ N_t \] = the number of trucks  
\[ N_s \] = the number of supply points  
\[ N_d \] = the number of demand points  
\[ M_i \] = the number of feasible routes for truck \( i \)  
\[ S_k \] = the supply at point \( k \)  
\[ D_p \] = the demand at point \( p \)  
\[ A_{ijk} \] = the quantity picked up at supply point \( k \) by truck \( i \) using route \( j \)  
\[ B_{ijp} \] = the quantity delivered at demand point \( p \) by truck \( i \) using route \( j \)  
\[ C_{ij} \] = the cost of route \( j \) used by truck \( i \)

\[
[P1] \quad \min \quad z = \sum_{i=1}^{N_t} \sum_{j=1}^{M_i} C_{ij} x_{ij}
\]

s.t.

\[
\sum_{j=1}^{M_i} x_{ij} = 1, \quad i = 1..N_t \tag{2.1}
\]

\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_i} A_{ijk} x_{ij} \leq S_k, \quad k = 1..N_s \tag{2.2}
\]

\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_i} B_{ijp} x_{ij} \geq D_p, \quad p = 1..N_d \tag{2.3}
\]

\[ x_{ij} \in \{0,1\}, \quad i = 1..N_t, \quad j = 1..M_i \]

The objective is to minimize the total cost of the transportation. Constraint 2.1 guarantees that each truck only drives one route. 2.2 assures that the totally quantity picked up does not exceed the supply. 2.3 assures that the demand is satisfied.

2.1.2 The second model [P2]

In order to always find feasible solutions we have introduced slack and surplus variables in the model. With these variables the demand can either be exceeded or not met. Added variables and parameters to the first model (P1) are presented below.

**Added Variables:**

\[ r_k \] = slack variables in the supply constraints \( k \)  
\[ q_p \] = slack variables in the demand constraints \( p \)  
\[ q_p^s \] = surplus variables in the demand constraints \( p \)

**Added Parameters:**

\[ P_k \] = penalty cost at supply point \( k \), per unit of weight  
\[ P_p^l \] = penalty cost for not meeting the demand \( p \), per unit of weight  
\[ P_p^u \] = penalty cost for exceeding the demand \( p \), per unit of weight
2.1 Multiperiod log truck scheduling

\[ P2 \]
\[
\min \ z = \sum_{i=1}^{N_t} \sum_{j=1}^{M_i} C_{ij} x_{ij} + \sum_{k=1}^{N_s} P_k r_k + \sum_{p=1}^{N_d} P^l_p q^l_p + \sum_{p=1}^{N_d} P^u_p q^u_p
\]
\[
s.t.
\sum_{j=1}^{M_i} x_{ij} = 1, \quad i = 1..N_t \tag{2.4}
\]
\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_i} A_{ijk} x_{ij} + r_k = S_k, \quad k = 1..N_s \tag{2.5}
\]
\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_i} B_{ijp} x_{ij} + q^l_p - q^u_p = D_p, \quad p = 1..N_d \tag{2.6}
\]
\[
r^l_k \geq 0, \quad k = 1..N_s \tag{2.7}
\]
\[
q^l_p \geq 0, \quad p = 1..N_d \tag{2.8}
\]
\[
q^u_p \geq 0, \quad p = 1..N_d \tag{2.9}
\]
\[
x_{ij} \in \{0,1\}, \quad i = 1..N_t, \quad j = 1..M_i
\]

In this model the objective function consists of four parts. The first part is like the previous model; to minimize the total route cost. The second is to minimize the total store cost at the supply points. If the penalty cost \( P_k \) is high, it means that we want to empty supply point \( k \), for instance if a product can not be stored in the forest without substantial value decrease. However, if \( P_k \) is close to zero, logs are only picked up at supply point \( k \) if this improves the solution. The user can decide "the priority" of different supply points. A high priority is used when it is important to empty the supply point. The third and fourth part is to minimize penalty costs that occur when the demand has either been exceeded or not met. When the variable \( q^l_p \) is larger then zero the demand is not met and when \( q^u_p \) is larger then zero the demand has been exceeded. To not meet the demand will be penalized harder then if the demand has been exceeded.

2.1.3 The third model \([P3]\)

It is possible to give the demand an upper and lower limit. The demand \( D_p \) of the previous model (\( P2 \)) has been replaced by a demand interval \([D^l_p, D^u_p]\) where \( D^l_p \) is the lower limit of the demand and \( D^u_p \) is the upper limit of the demand for a single day. Below the added and updated variables and parameters to the model are presented.

**Added and Updated Variables:**

- \( l^p \) = slack variable in the lower limit demand constraint \( p \)
- \( l^u \) = surplus variable in the lower limit demand constraint \( p \)
- \( u^l_p \) = slack variable in the upper limit demand constraint \( p \)
- \( u^u_p \) = surplus variable in the upper limit demand constraint \( p \)
Added and Updated Parameters:

\( D^l_p \) = lower limit of the demand at point \( p \)
\( D^u_p \) = upper limit of the demand at point \( p \)
\( P^l_p \) = penalty cost for not meeting the lower limit of demand \( p \), per unit of weight
\( P^u_p \) = penalty cost for exceeding the upper limit of demand \( p \), per unit of weight

\[
[P3] \quad \text{min} \quad z = \sum_{i=1}^{N_t} \sum_{j=1}^{M_t} C_{ij}x_{ij} + \sum_{k=1}^{N_d} P^l_k r_k + \sum_{p=1}^{N_d} P^l_p u^l_p + \sum_{p=1}^{N_d} P^u_p u^u_p
\]

s.t.

\[
\sum_{j=1}^{M_t} x_{ij} = 1, \quad i = 1..N_t
\]

(2.10)

\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_t} A_{ijk}x_{ij} + r_k = S_k, \quad k = 1..N_s
\]

(2.11)

\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_t} B_{ijp}x_{ij} + l^l_p - l^u_p = D^l_p, \quad p = 1..N_d
\]

(2.12)

\[
\sum_{i=1}^{N_t} \sum_{j=1}^{M_t} B_{ijp}x_{ij} + u^l_p - u^u_p = D^u_p, \quad p = 1..N_d
\]

(2.13)

\[
r_k \geq 0, \quad k = 1..N_s
\]

(2.14)

\[
l^l_p \geq 0, \quad p = 1..N_d
\]

(2.15)

\[
l^u_p \geq 0, \quad p = 1..N_d
\]

(2.16)

\[
u^l_p \geq 0, \quad p = 1..N_d
\]

(2.17)

\[
u^u_p \geq 0, \quad p = 1..N_d
\]

(2.18)

\[
x_{ij} \in \{0, 1\}, \quad i = 1..N_t, \quad j = 1..M_t
\]

Constraint 2.6 in model P2 has now been divided into two new constraints 2.12 and 2.13. The variable \( q^u_p \) and \( q^l_p \) are replaced with \( l^u_p \) and \( l^l_p \) in the lower limit of the demand \( (D^l_p) \) and \( u^u_p \) and \( u^l_p \) in the upper limit of the demand \( (D^u_p) \). We penalize if the demand is not met within the lower and upper limits interval, i.e. when the total quantity delivered is less than the demand in the lower limit constraint or when the total quantity delivered is larger than the demand in the upper limit constraint.
2.1 Multiperiod log truck scheduling

2.1.4 The final model [P4]

Here we present the final model which is used to solve the multiperiod log truck scheduling problem. All previous models address the problem over a time period of one day. In order to handle a time period of several days we introduce a time index \( d \) into the model. Here, a decision must be made regarding which day the trucker drives the route. Noticeable in this model is that the demand accumulates, i.e. the demand is given up to the current day.

**Decision variable:**

\[
x_{ijd} = \begin{cases} 
1 & \text{if truck } i \text{ drive route } j \text{ on day } d \\
0 & \text{otherwise} 
\end{cases}
\]

**Added and Updated Variables:**

- \( r_{kd} \) = slack variable in the supply constraint \( k \) on day \( d \)
- \( l_{pd}^l \) = slack variable in the lower limit demand constraint \( p \) on day \( d \)
- \( l_{pd}^u \) = surplus variable in the lower limit demand constraint \( p \) on day \( d \)
- \( u_{pd}^l \) = slack variable in the upper limit demand constraint \( p \) on day \( d \)
- \( u_{pd}^u \) = surplus variable in the upper limit demand constraint \( p \) on day \( d \)

**Added and Updated Parameters:**

- \( N_t \) = the number of trucks
- \( N_k \) = the number of supply points
- \( N_d \) = the number of demand points
- \( M_i \) = the number of feasible routes for truck \( i \)
- \( N_e \) = the number of days
- \( S_{kd} \) = the supply at point \( k \) on day \( d \)
- \( D_{pd} \) = the demand at point \( p \) on day \( d \)
- \( A_{ijkd} \) = the quantity picked up at supply point \( k \) by truck \( i \) using route \( j \) on day \( d \)
- \( B_{ijpd} \) = the quantity delivered at demand point \( p \) by truck \( i \) using route \( j \) on day \( d \)
- \( C_{ij} \) = the cost of route \( j \) used by truck \( i \)
- \( D_{pd}^l \) = lower limit of the **accumulated demand** at point \( p \) on day \( d \)
- \( D_{pd}^u \) = upper limit of the **accumulated demand** at point \( p \) on day \( d \)
- \( P_{pd}^l \) = penalty cost at supply point \( k \) per unit of weight day on \( d \)
- \( P_{pd}^s \) = penalty cost for not meeting the lower limit of demand \( p \), per unit of weight, on day \( d \)
- \( P_{pd}^u \) = penalty cost for exceeding the upper limit of demand \( p \), per unit of weight, on day \( d \)
\[ \begin{align*}
\text{min } & \quad z = \sum_{d=1}^{N_d} \sum_{i=1}^{N_t} \sum_{j=1}^{M_i} C_{ij} x_{ijd} + \sum_{d=1}^{N_d} \sum_{k=1}^{N_s} P_{kd} r_{kd} + \sum_{d=1}^{N_d} \sum_{p=1}^{N_s} P_{pd}^l + \sum_{d=1}^{N_d} \sum_{p=1}^{N_s} P_{pd}^u \tag{P4} \\
\text{s.t.} & \quad \sum_{j=1}^{M_i} x_{ijd} = 1, \quad d = 1..N_d, i = 1..N_t \\
& \quad \sum_{i=1}^{N_t} \sum_{j=1}^{M_i} A_{ij} x_{ijd} - r_{kd-1} + r_{kd} = S_{kd}, \quad d = 1..N_d, k = 1..N_s \\
& \quad \sum_{i=1}^{N_t} \sum_{j=1}^{M_i} B_{ij} x_{ijd} + (D_{pd-1}^l - l_{pd-1}^l + l_{pd-1}^u) + l_{pd-1}^l - l_{pd}^l = D_{pd}^l, \quad d = 1..N_d, p = 1..N_s \\
& \quad \sum_{i=1}^{N_t} \sum_{j=1}^{M_i} B_{ij} x_{ijd} + (D_{pd-1}^u - u_{pd-1}^l + u_{pd-1}^u) + u_{pd-1}^l - u_{pd}^l = D_{pd}^u, \quad d = 1..N_d, p = 1..N_s \\
& \quad r_{kd} = 0, \quad k = 1..N_s \\
& \quad l_{pd}^l = 0, \quad p = 1..N_d \\
& \quad l_{pd}^u = 0, \quad p = 1..N_d \\
& \quad u_{pd}^l = 0, \quad p = 1..N_d \\
& \quad u_{pd}^u = 0, \quad p = 1..N_d \\
& \quad x_{ijd} \in \{0, 1\}, \quad i = 1..N_t, \quad j = 1..M_i, \quad d = 1..N_d \\
\end{align*} \]

Here, the objective function differs from that of the objective function in model (P3) in that the new index \( d \) takes the time frame days into consideration. In Constraint 2.11 the slack variable \( r_{kd-1} \) has been added. This variable provides information about what has been picked up the day before. Constraints 2.21 and 2.22 are a bit more complex but they are based on the same principle as the supply constraint 2.20. In this way we give the model opportunity to meet the total demand over a period of several days. Here the variables \( l_{pd-1}^l, l_{pd-1}^u, u_{pd-1}^l \) and \( u_{pd-1}^u \) provide information about what has been delivered the day before. Since day zero does not exist, all variables containing day zero are forced to zero (see constraints 2.23, 2.24, 2.25, 2.26 and 2.27).
Chapter 3

Optimization Methods

In this chapter we give a theoretical background to the optimization methods used for solving multiperiod log truck scheduling.

3.1 Column generation

Column generation is an optimization method that is common for solving problems with a large number of variables. The difficulty for these problems are in generating and storing the constraint matrix. In order to illustrate this procedure we start with a general problem (P):

\[
[P] \quad \max \quad \sum_{i=1}^{N} c_i x_i \\
\text{subject to} \quad \sum_{i=1}^{N} A_i x_i = b \quad |y \\
\quad x_i \in \{0, 1\}, \quad i = 1..N
\]

c is a cost vector of size N where N is a large number, x is a vector of decision variables of size N, A is an \( M \times N \) matrix, b is a column vector and y is a vector of dual variables, both of size M. To apply column generation we need to relax the problem to a linear programming (LP) problem. The LP relaxation of the general problem is:

\[
[LP] \quad \max \quad \sum_{i=1}^{N} c_i x_i \\
\text{subject to} \quad \sum_{i=1}^{N} A_i x_i = b \quad |y \\
\quad 0 \leq x_i \leq 1, \quad i = 1..N
\]

If N is too large the original matrix A of size \( M \times N \) becomes too large to solve. Therefore we reduce the matrix A to the size \( M \times n \) (\( n \ll N \)) where an LP solution can be found. The new restricted LP problem (R), called the Master problem is given below:
\[
[R] \quad \text{max} \quad \sum_{i=1}^{n} c_i x_i \\
\text{subject to} \quad \sum_{i=1}^{n} A_i x_i = b \quad |y| \\
0 \leq x_i \leq 1, \quad i = 1..n
\]

The process of generating columns that can be added to the Master problem is another optimization problem, called the Sub problem. The objective is to maximize (minimize) the reduced cost \( \tau_r = c_r - y^T A_r \), where \( c_r \) is the cost for column vector \( A_r \). A new column with a reduced cost \( \tau_r > 0 \) (\( < 0 \) if it is a minimization problem) improves the objective value of the Master problem.

In order to solve the LP problem we continue alternately solving the Master problem and the Sub problem. When there is no reduced cost \( \tau_r > 0 \), optimum to the LP problem is found.

**The column generation process can be written as:**

1. Solve the Master problem. Calculate the dual variables \( \bar{y} \).
2. Solve the Sub problem using the dual variables \( \bar{y} \).
3. Select the column with the greatest reduced cost \( \tau_r > 0 \), update the constraint matrix with column \( A_r \) and its cost \( c_r \) in the Master problem and go to step 1. If there is no column with \( \tau_r > 0 \), STOP! Optimum to the LP relaxation is found.

Column generation is a two-level method where the Master problem solves the LP problem and the Sub problem generates new columns.

![Figure 3.1: The column generation procedure](image)

### 3.2 Composite pricing

In our thesis we do not use column generation by solving a proper subproblem to produce new columns since a pool of columns is already given. Instead we use the approach referred to as composite pricing. A column where the reduced cost is computed is called
3.3 Branch and Bound

If the optimum of the LP relaxation is found, an Integer programming (IP) method can be used to acquire an optimal integer solution. Branch and Bound is a standard method used to solve IP problems and this method is only briefly described in this section. The solution is optimal to the IP if the optimum of the LP relaxation is of type integer. Otherwise, if the optimum to the LP relaxation is not of type integer, the feasible region of the LP relaxation is divided into two subsets. There are several branching strategies which decide how to divide these feasible regions. Variable branching is the strategy we use in this thesis and is described more in detailed in next section (3.3.1). When a branching strategy is chosen, a search strategy is used to decide the order to search through the nodes.

It is not always necessary to search through all nodes to find the optimum to the integer problem. Sometimes it is possible to "prune" the branches when no better integer solutions can be found in these. There are three criteria for pruning the branches. The first is when the LP solution of the subproblem is larger than the best integer solution found, the second when the LP solution is infeasible and the third when the solution is of type integer.

The algorithm for the Branch and Bound method

1. Solve the LP relaxation. If the solution is of type integer, STOP! Integer solution found.
2. Branch. Divide the feasible region into two subsets.
3. Choose a subproblem. Solve this subproblem. If any pruning criteria is fulfilled, go to step 4. Otherwise go to step 2.
4. Prune the tree and update the lower and upper bounds.
5. If there are no nodes left to explore, STOP! Return the best integer solution if it exists. Otherwise go to step 3.
3.3.1 Variable branching

Assume that we have a fractional variable $\bar{x}_j$ to the LP solution. Variable branching divides $x_j$ into two nodes: one node where the variable $x_j \geq \lfloor \bar{x}_j \rfloor$ and one node where $x_j \leq \lfloor \bar{x}_j \rfloor + 1$. When there are several variables with fractional values after solving the LP relaxation, for instance $\bar{x} = [4.5, 0.3, 1, 2.9]$, the value closest to an integer is chosen to branch on (here $x_4$). The motivation for doing this is that the fractional value, for instance $x_4$ in this case is more likely to become three than two in an integer solution.

When the variables $x_j$ are binary, the variable $x_j = 0$ in one node and $x_j = 1$ in the other node. An unbalanced tree is a disadvantage that sometimes occurs with this branching strategy. This is a problem because in one of the branches a prune criteria is reached immediately while in the other branch nothing happens. It can arise when there is a convexity constraint equal to 1 with binary variables. The worst case is when no pruning criteria is fulfilled until all nodes have been searched through. This leads to lot of time is required for finding an binary solution.

![Figure 3.2: The structure of an unbalanced tree for the worst case with n variables](image)

3.3.2 Searching strategies

The most common search strategies are depth first, best first and breadth first. Depth first is used when we want to find a feasible solution quickly. It searches for nodes at the bottom of the tree, but the disadvantage is that the solution may be far from the optimal solution. Best first chooses the node with the best LP solution. This strategy finds good solutions, but requires a lot of memory for creating large trees. Finally, breadth first investigates all nodes at one level before proceeding to the next level, where the process is repeated. Best first is the strategy used in this thesis. For more information of these strategies see Parker, 1988.
Chapter 4

Our Method

In this chapter we introduce two approaches for solving the multiperiod route planning problem based on the final mathematical model in Chapter 2.

4.1 An approach based on composite pricing

Log truck scheduling (LTS) is often a very large and complex problem to solve. Typically, the constraint matrix contains a large number of columns. In our model where a column represents a route for a whole day, column generation is a common approach suitable for solving this type of problem. Moreover since a pool of columns already is given, we use a composite pricing approach (see Section 3.2).

Our strategy for solving the LTS problem is divided into two steps. In the first step we solve the LP relaxation of the problem using a composite pricing approach. In the second step we solve the mixed integer programming (MIP) problem based on the columns that have been added to the problem in the first step. The Branch and Bound method (see Section 3.3) is used for solving the MIP problem.

4.2 Initial problem

In order to get started we must have an initial problem. We start with a complete model containing the quantities of the demands and the supplies, as well as slack and surplus variables and their penalty costs. Furthermore, empty routes with cost zero that pick up and deliver nothing are included in the initial problem for each truck. These routes allow a truck to stay at home.

4.2.1 An example

In order to illustrate an initial problem we present an example where we have one truck, two supplies and one demand. The time period reaches over two days. In the model,
two empty routes \((x_{111}, x_{112})\) have been added. The objective function consists of the costs for these two routes (with cost zero), the penalty costs for not meeting (9575 and 8025) and for exceeding (600 and 650) the demand for days one and two respectively. We have chosen to not add any penalty cost for storing logs in the forest. The quantity at the first supply point is 50 units while the quantity of the second supply point is 30 units day one. Furthermore, during day two 10 units are added to the supply point two. The demand for day one is between \([30,45]\) units and the demand for day two is 20 units. This results in that the total demand for these two days are \([30+20,45+20]\) = \([50,65]\).

The only feasible solution to this problem is to let the truck stay at home for both days. The objective value obtained after solving the problem is the minimal cost of all penalties in the model. When no transportation is carried out, columns based solely on penalty minimization are added. The initial problem is illustrated below:

\[
[Ex1] \quad \min \quad 0x_{111} + 0x_{112} + 9575x_{111} + 8025x_{122} + 600u_{111} + 650u_{122} \\
\text{s.t.} \\
x_{111} = 1 \\
x_{112} = 1 \\
+ r_{11} + r_{12} = 50 \\
- r_{11} + r_{12} = 0 \\
+ r_{21} + r_{22} = 10 \\
- r_{21} + r_{22} = 0 \\
+ t_{11}^v - t_{11}^i = 30 \\
+ u_{11}^v - u_{11}^i = 45 \\
- t_{11}^v + t_{11}^i + t_{12}^v - t_{12}^i = 20 \\
- u_{11}^v + u_{11}^i + u_{12}^v - u_{12}^i = 20 \\
r_{11}, r_{12}, r_{21}, r_{22} \geq 0 \\
t_{11}^i, t_{11}^v, t_{12}^i, t_{12}^v \geq 0 \\
u_{11}^i, u_{11}^v, u_{12}^i, u_{12}^v \geq 0 \\
x_{111}, x_{112} \in \{0,1\}
\]

4.3 An example of adding routes

Here we present an example that illustrates when routes have been added to the model. In the model below, two routes for day one \((x_{121}, x_{131})\) and one route for day two \((x_{122})\) have been added to model (Ex1). Route \(x_{121}\) picks up at supply point one and
4.4 Algorithmic description

delivers to the customer. Routes $x_{121}$ and $x_{122}$ pick up at supply points one and two before delivering to the customer.

$$[Ex2] \ \text{min} \ \ 429x_{121} + 529x_{131} + 625x_{122} + 9575l_{11} + 8025l_{12} + 600u_{11}$$
$$+ 650u_{12}$$

s.t.

$$x_{11} + x_{121} + x_{131} = 1$$
$$x_{12} + x_{122} = 1$$

$$35x_{121} + 10x_{131} + r_{11} = 50$$
$$10x_{122} - r_{11} + r_{12} = 0$$
$$30x_{131} + r_{21} = 30$$
$$10x_{122} - r_{21} + r_{22} = 10$$

$$35x_{121} + 40x_{131} + t_{11} - t_{11} = 30$$
$$35x_{121} + 40x_{131} + u_{11} - u_{11} = 45$$
$$20x_{122} - t_{11} + t_{12} + l_{12} - t_{12} = 20$$
$$20x_{122} - u_{11} + u_{12} + u_{12} - u_{12} = 20$$

$$r_{11}, \ r_{12}, \ r_{21}, \ r_{22} \geq 0$$
$$l_{11}, \ l_{12}, \ l_{12}, \ l_{12} \geq 0$$
$$u_{11}, \ u_{11}, \ u_{12}, \ u_{12} \geq 0$$

$$x_{111}, \ x_{121}, \ x_{131}, \ x_{112}, \ x_{122} \in \{0,1\}$$

4.4 Algorithmic description

The problem we aim to solve is [P4] (see Section 2.1.4). To solve this model we present two different methods. The first method minimizes the problem for all days at once. The second method solves each day separately, e.g. when day one has been minimized, the solution from day one is fixed and added to day two. As a result, the solution from day one cannot be updated when solving day two and days one and two cannot be updated when solving day three, etc. We have decided to name the first the Integrated Method and the second the Sequential Method. The algorithms for solving these methods are summarized below.

The algorithm for the Integrated Method

1. Solve LP relaxation of [P4] including all days.

2. Calculate reduced costs $\bar{c}$ for all feasible routes.

3. Add routes with reduced cost less than zero to the model and go to step 1. If there are no such routes, go to step 4.

4. Change all variables back to binary and solve the MIP problem.
The algorithm for the Sequential Method

1. Start with day one.

2. Solve the LP relaxation of [P4] for current day.

3. Calculate reduced costs $\bar{c}$ for all feasible routes belonging to the current day.

4. Add routes to the model with reduced cost less than zero and go to step 2. If there are no routes with reduced cost less than zero, go to step 5.

5. Change all variables back to binary and solve the MIP problem for the current day.
   If last day or an infeasible MIP solution, STOP!

6. Fix the variables belonging to the current day according to the solution obtained in step 5.

7. Change to next day and go back to step 2.

4.5 Addition of routes

After the routes have gradually been added to the LP relaxation, the next step is to solve the MIP problem. Here a problem is that the number of routes can be too few to find an integer solution that covers the demand. Therefore we need to add more routes and instead of adding the route with the most negative reduced cost, we add the first route we find with a negative reduced cost. This leads to that more routes than necessary are included to optimize the LP relaxation. Moreover, to pick the first saves a lot of time since it is not necessary to loop through all reduced costs in order to find the most negative.
4.5 Addition of routes

Figure 4.1: Solution methodology for the Integrated Method

Figure 4.2: Solution methodology for the Sequential Method
Chapter 5

Results and Conclusions

In this chapter we present the solutions of the Integrated Method and the Sequential Method based on a realistic case.

5.1 The case

Our log truck scheduling problem lasts over a period of three days. We have five trucks that are available every day, resulting in 15 (5 \times 3) constraints. Besides that, we have 68 supply points which have a quantity for each day. The total number of supply constraints is therefore 204 (68 \times 3). Furthermore, we have 15 demand points, each of which is divided into an upper and lower limit per day, which gives us 90 (15 \times 3 \times 2) demand constraints. The number of feasible routes that have been created in this case is 127402 per day i.e. 382206 (127402 \times 3) routes (columns) for the entire period. All data was obtained from the transportation company Sydved.

5.2 Different tests and parameters

There are two parameters in our problem where the first decides the number of routes with negative reduced cost that will be added to the problem and the second decides the time limit for solving the MIP problem. The reason for introducing a time limit for the MIP problem is because the MIP solver demands too much time to solve the current problem to optimality. We have two time limits; 15 and 60 minutes. For the Integrated Method, the time limit lasts over all days, whereas for the Sequential Method the time limit is splitted up into three smaller limits, one for each day. For instance when the overall time limit is 15 minutes, the limit for each day is five minutes.

The initial problem contains 429 columns (InitCol), these are included in the total set of columns. The penalty cost for exceeding and not meeting the demand is relatively large in comparison to the cost of the routes and the reason for this is to fulfil the demand as much as possible when solving the problem. We have no priority for emptying a supply
Results and Conclusions

point. We present the results of adding 1, 10, 100, 200 and 500 columns. In the tables below we display the number of incoming columns (InCol), the total number of columns that have been included in the problem (TotCol), the objective value (Objective Value), the penalties obtained in the solution (Penalty), and the number of trucks used (Trucks Used). Also displayed is the time for solving the problem i.e. the CPU-time for solving the LP relaxation plus the CPU-time for solving the MIP problem.

The time limit is a maximum limit for solving a given MIP problem, not a time limit for solving the entire problem. Assume that the time limit is 30 minutes and the CPU-time for solving the LP relaxation is 5 minutes. If the time limit for solving the MIP problem exceeds, it results in that the total time for solving the problem is 35 (30+5) minutes.

Every time columns are added, an LP problem is solved. The total number of times it is solved can be calculated as:

$$\text{No of Times} = \left\lfloor \frac{\text{TotCol} - \text{InitCol}}{\text{InCol}} \right\rfloor$$

(5.1)

5.2.1 A short time limit (15 minutes)

In the following two tables we have run both methods with a time limit of 15 minutes. The results below are used for making comparisons (Section 5.3) between our methods when there is a short amount of time.

<table>
<thead>
<tr>
<th>InCol</th>
<th>TotCol</th>
<th>Objective Value</th>
<th>Penalty</th>
<th>Trucks Used</th>
<th>CPU-Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2058</td>
<td>556536</td>
<td>0</td>
<td>26</td>
<td>53,5</td>
</tr>
<tr>
<td>10</td>
<td>3492</td>
<td>557808</td>
<td>0</td>
<td>26</td>
<td>24,5</td>
</tr>
<tr>
<td>100</td>
<td>10334</td>
<td>791208</td>
<td>240000</td>
<td>26</td>
<td>20,2</td>
</tr>
<tr>
<td>200</td>
<td>14924</td>
<td>949588</td>
<td>400000</td>
<td>26</td>
<td>19,9</td>
</tr>
<tr>
<td>500</td>
<td>25074</td>
<td>1074210</td>
<td>520000</td>
<td>27</td>
<td>19,2</td>
</tr>
</tbody>
</table>

Table 5.1: Results from the Integrated Method, 15 minutes time limit

<table>
<thead>
<tr>
<th>InCol</th>
<th>TotCol</th>
<th>Objective Value</th>
<th>Penalty</th>
<th>Trucks Used</th>
<th>CPU-Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1930</td>
<td>576036</td>
<td>0</td>
<td>26</td>
<td>13,7</td>
</tr>
<tr>
<td>10</td>
<td>3389</td>
<td>589116</td>
<td>0</td>
<td>27</td>
<td>9,1</td>
</tr>
<tr>
<td>100</td>
<td>8720</td>
<td>574008</td>
<td>0</td>
<td>27</td>
<td>8,5</td>
</tr>
<tr>
<td>200</td>
<td>14953</td>
<td>575820</td>
<td>0</td>
<td>27</td>
<td>12,6</td>
</tr>
<tr>
<td>500</td>
<td>23818</td>
<td>580116</td>
<td>0</td>
<td>27</td>
<td>12,6</td>
</tr>
</tbody>
</table>

Table 5.2: Results from the Sequential Method, 15 minutes time limit

Analysis from Table 5.1 shows that solving a MIP problem containing 10334 columns (TotCol) or more includes penalties in the solution. The reason is that the problem is too large for the MIP solver to find an integer solution within 15 minutes. When
5.2 Different tests and parameters

the MIP problem contains 3492 columns or less the Integrated Method finds solutions without penalties. Moreover, the results presented for the Sequential Method (Table 5.2) shows that the solver is capable of finding solutions without penalties. Since the Sequential Method divides the problem into three smaller problems, each problem can now be solved without penalties.

According to the formula 5.1, we see that the number of times we need to solve a LP problem decreases when we increase the number of incoming columns. The MIP solver does not solve any of the five tests to optimality i.e. the time limit is exceeded for all these tests. Due to this the total solving time (CPU-Time) decreases for larger tests. For the Sequential Method the time limit exceeds only for certain days. When the number of columns increases, the MIP solver needs more time for finding an integer solution.

The best objective value found with the Integrated Method (556536) is 3.0% better than the best found with the Sequential Method (574008), but the time difference was large.

5.2.2 A large time limit (60 minutes)

In these tables we have run both methods with a time limit of 60 minutes. Here the results are used for making comparisons (Section 5.3) when there is plenty of time.

<table>
<thead>
<tr>
<th>InCol</th>
<th>TotCol</th>
<th>Objective Value</th>
<th>Penalty</th>
<th>Trucks Used</th>
<th>CPU-Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2058</td>
<td>556536</td>
<td>0</td>
<td>26</td>
<td>98,5</td>
</tr>
<tr>
<td>10</td>
<td>3492</td>
<td>554784</td>
<td>0</td>
<td>26</td>
<td>69,4</td>
</tr>
<tr>
<td>100</td>
<td>10334</td>
<td>555036</td>
<td>0</td>
<td>26</td>
<td>65,4</td>
</tr>
<tr>
<td>200</td>
<td>14924</td>
<td>557604</td>
<td>0</td>
<td>26</td>
<td>64,9</td>
</tr>
<tr>
<td>500</td>
<td>25074</td>
<td>582372</td>
<td>0</td>
<td>27</td>
<td>64,7</td>
</tr>
</tbody>
</table>

Table 5.3: Results from the Integrated Method, 60 minutes time limit

<table>
<thead>
<tr>
<th>InCol</th>
<th>TotCol</th>
<th>Objective Value</th>
<th>Penalty</th>
<th>Trucks Used</th>
<th>CPU-Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1930</td>
<td>576036</td>
<td>0</td>
<td>26</td>
<td>13,7</td>
</tr>
<tr>
<td>10</td>
<td>3389</td>
<td>589116</td>
<td>0</td>
<td>27</td>
<td>24,0</td>
</tr>
<tr>
<td>100</td>
<td>9164</td>
<td>573240</td>
<td>0</td>
<td>27</td>
<td>28,1</td>
</tr>
<tr>
<td>200</td>
<td>14953</td>
<td>575040</td>
<td>0</td>
<td>27</td>
<td>33,9</td>
</tr>
<tr>
<td>500</td>
<td>23594</td>
<td>576168</td>
<td>0</td>
<td>27</td>
<td>43,2</td>
</tr>
</tbody>
</table>

Table 5.4: Results from the Sequential Method, 60 minutes time limit

An analysis of Table 5.3 shows that when the time limit increases no penalties occur for the Integrated Method, but the time limit for all five tests is still exceeded. Noticable in Table 5.4 is that the Sequential Method finds a solution with a objective value 1.1% better than the Integrated Method when there is 500 incoming columns. Furthermore, this solution is found within even less time used.

The best objective value found with the Integrated Method (554784) is 3.2% better than the best objective value found with the Sequential Method (573240). The time
difference was more than double for finding the best value with the Integrated Method.

5.3 Comparison and discussion

The Integrated Method finds better solutions than the Sequential Method for both time limits. The gap between the best objective value for the respective methods is approximately 3.0% for the short time limit and 3.2% for the larger time limit. A considerable improvement of the solutions is more obvious for the Integrated Method than for the Sequential Method for almost all tests when the time limit increases. This depends on the fact that the Integrated Method needs more time to solve one large MIP problem than the Sequential Method needs to sequentially solve three small MIP problems. We saw that when there are many columns (500 incoming) included, the Sequential Method finds a better solution than the Integrated Method. Furthermore, an advantage with the Sequential Method that is important to mention is the possibility to find a solution for one day even if a solution for the whole period is not found.

5.4 Conclusion

For the multiperiod route planning problem we have presented two methods; one method called the Integrated Method with the purpose of minimizing the total route cost for all days and another method called the Sequential Method with the purpose of minimizing each day separately. In our case where the period reaches over three days, a natural and expected result is that the Integrated Method solves the problem more satisfactorily than the Sequential Method. The best objective value for the solution found with the Integrated Method was more than 3.2% better than the best found with the Sequential Method. The gap between the best objective value with the Integrated Method (554784 given in Table 5.3) and the optimum to the LP relaxation (541997) is 2.3%. This gap gives us an indication that we may not be far away from an actual optimum.

When there is a large number of columns added, the Sequential Method handles the problem better. If the case gets more extensive, our results indicate that the Sequential Method is more useful in most cases.

5.5 Further work

As mentioned earlier we do not generate our own routes and therefore a very interesting thing to try next would be to implement a complete sub problem that generates routes. It is hard to say how this would affect the solution time but certainly this would improve the objective value.

It is also important to test different parameter values, for instance updating and changing priorities and penalties at demand and supply points.
Chapter 6

Daily Log Truck Scheduling with predefined trips

In this chapter we present a mathematical model for creating routes for one day using a set of predefined trips. This model underlies further development of the method for solving the daily log truck scheduling problem.

6.1 Predefined trips

Before the felling process begins, the companies can decide where the logging piles should be transported. Using these predefined destinations, loaded trips from the supply points to the demand points are created. These loaded trips must be carried out and are referred to as predefined trips in this thesis.

There are in some cases an interest in making these trips more efficient and one way is to investigate the backhauling possibilities. To find these types of backloads (carry a load on the return trip), it is necessary to search for supply points in the neighbourhood of demand points. Figure 6.1 shows the advantage of loading the truck next to a delivery point. Assume that dist(S1,D1) = 9, dist(S2,D2) = 10, dist(S3,D3) = 9, dist(S3,D1) = 10 and dist(S1,D3) = 11 where dist(m,n) is the distance between destination m and n. Then the total distance for the loaded trips in Procedure 1 is 28 (9+10+9) whereas the total distance in Procedure 2 is 31 (10+10+11). Further, assume that dist(D1,S2)=1 and dist(D2,S3)=1 then the total distance for the routes in Procedure 1 is 56 whereas the distance for the routes in Procedure 2 is 44. Even though the total distance for the loaded trips are longer in Procedure 2 than Procedure 1 the total distance for the routes in Procedure 2 is less than Procedure 1.

The purpose of backhauling is to reduce the distance for the unloaded trips. For more information see Carlsson and Rönnqvist, 1998.
6.2 The model

In this section we present a model for creating individual routes, one for each truck, given a set of predefined trips. The model has a similar structure as of a VRP (see Section 1.4) where a node represents a predefined trip (a pick-up and delivery). A route starts at the truck's home location, visits a number of nodes and then returns to its home location. The model is only presented to illustrate our problem, we do not actually solve the model. Instead we use a heuristic algorithm for creating routes for each truck (described in Chapter 8).

Figure 6.2: An illustration of a possible route where each node represents a predefined trip
6.2 The model

Notations and variables are described as follow:

**Decision variable:**

\[ y_{ijt} = \begin{cases} 
1 & \text{if node } i \text{ to node } j \text{ is driven by truck } t \\
0 & \text{otherwise} 
\end{cases} \]

**Parameters:**

- \( \mathcal{H} \) = The set of trucks
- \( \mathcal{D} \) = The set of nodes representing a predefined trip
- \( o_t \) = The node representing the home location of truck \( t \)
- \( A_t \) = \( \mathcal{D} \cup o_t \)
- \( C_{ij} \) = The cost of driving from node \( i \) to node \( j \)
- \( N_{it} \) = The time for pick-up and delivering in node \( i \), \( N_{ot} = 0 \)
- \( T_{ij} \) = The time for driving from node \( i \) to node \( j \)
- \( \max \) = Maximum time limit for a route

\[
\min w = \sum_{t \in \mathcal{H}} \sum_{i \in A_t} \sum_{j \in A_t \setminus \{i\}} C_{ij} y_{ijt}
\]

s.t.

\[
\sum_{j \in \mathcal{D}} y_{oij} = 1, \quad \forall t \in \mathcal{H} \tag{6.1}
\]

\[
\sum_{j \in \mathcal{D}} y_{ijo} = 1, \quad \forall t \in \mathcal{H} \tag{6.2}
\]

\[
\sum_{i \in A_t \setminus \{j\}} y_{ij} - \sum_{i \in A_t \setminus \{j\}} y_{ji} = 0, \quad \forall j \in \mathcal{D}, \quad \forall t \in \mathcal{H} \tag{6.3}
\]

\[
\sum_{i \in \mathcal{H}} \sum_{j \in A_t \setminus \{i\}} y_{ij} = 1, \quad \forall i \in \mathcal{D} \tag{6.4}
\]

\[
\sum_{i \in \mathcal{H}} \sum_{j \in A_t \setminus \{i\}} (T_{ij} + N_{i}) y_{ij} \leq \max, \quad \forall t \in \mathcal{H} \tag{6.5}
\]

\[ y_{ijt} \in \{0, 1\}, \quad \forall i, j \in A_t : i \neq j, \quad t \in \mathcal{H} \]

The aim of the objective function is to minimize the total travel cost, i.e. minimize the total cost of the unloaded trips. Constraints 6.1 and 6.2 handle the fact that a truck must start at its home location and as well as return to its home location. 6.3 forces the truck to leave a node that has been visited. Furthermore, all nodes must be visited once, therefore 6.4 is added to the model. 6.5 insures that the maximum time limit for each route is not exceeded.
Chapter 7

Optimization Methods

This chapter includes some theoretical discussions about the method we use to solve the daily log truck scheduling problem.

7.1 Local search methods

For the model described in chapter 6, we have used a common local search method suitable for this type of problem. A local search method starts from some initial solution (this can be created in different ways). The initial solution is then locally improved in several iterations until a given stop criterion is reached. Local search methods are also known as neighbourhood search methods, where the neighbourhood is defined by a number of rules of how the solution can be changed. For local search methods, at each update, the neighbourhood of the current solution is investigated and the best solution is selected as a new solution. The structure of the neighbourhood is of great importance for the behaviour of the search process. Simulated annealing and tabu search are two known local search methods.

7.1.1 Tabu search

There are local search algorithms where the search stops in local minima; tabu search has the ability to avoid this (see Ropke, 2002). The method searches in the direction that improves the objective value the most and when a better solution is found, the solution is updated. Tabu search keeps on like this until it reaches a local minimum. One way to continue when there are no improving solutions found is to search in a direction where the solution value is equal to the value of the current solution. Should this procedure not be viable, choosing a worse solution is the way to continue. After such an update is made, tabu search finds an improving search direction back to the previous local minimum. To avoid this, a tabu list is introduced. A solution is tabu after the update, i.e. that solution is not allowed to be visited during a given number of iterations. The effect of tabu is illustrated in Figure 7.1.
Another problem is the cyclic behaviour that can occur in this heuristic, i.e. the search finds earlier explored local minima after a number of iterations. To handle this a penalty is added if the solution already has been visited. The size of the penalty depends on the visiting frequency (the higher the visiting frequency the larger the penalty). The purpose with this is to avoid earlier visited solutions and instead investigate new areas.

Tabu search does not stop when an optimum is found, it proceeds until a stop criterion is reached. The following is often used: stop after a given time, stop after a given number of changes (iterations) or stop if no improvement is found after a given number of changes. It is possible to combine these criteria with others to create new stop criteria.

In Figure 7.2 an example of a tabu and frequency matrix is presented. This kind of matrix is useful when illustrating the solution process. We use it for the implementation of the heuristic in C++.

![Diagram showing the tabu effect](image)

**Figure 7.1: An illustration of the tabu effect**

<table>
<thead>
<tr>
<th>Tabu</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4 3</td>
<td>2 6 8</td>
</tr>
<tr>
<td>2 1 3 4 5 6 7 8</td>
<td>Initial solution</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7 8</td>
<td>Solution 1</td>
</tr>
<tr>
<td>1 2 3 6 4 8</td>
<td>Solution 2</td>
</tr>
<tr>
<td>1 5 3 7 2 6 4 8</td>
<td>Solution 3</td>
</tr>
<tr>
<td>4 5 1 7 2 6 3 8</td>
<td>Solution 4</td>
</tr>
</tbody>
</table>

**Figure 7.2: A tabu and frequency matrix and solutions**

The matrix illustrates the number of changes that have been carried out, the order of the latest changes and the size of the tabu parameter. It is updated each time a change is made.
A tabu search algorithm for a minimization problem

1. Create an initial solution $Z_{initial}$, tabu list $= \emptyset$, frequency list $= \emptyset$.

2. $Z = Z_{initial}$.

3. Search in the neighbourhood for a better solution $	ilde{Z}$ that is not included in the tabu list, where cost for $\tilde{Z} < \text{cost for } Z$ or a solution that fulfils the given criteria.

4. Update the tabu list and the frequency list.

5. $Z = \tilde{Z}$.

6. Go to stage (3) or stop if a stop criterion is reached.

7. Return $Z$. 
Chapter 8

Our Heuristic

Here we present our method to solve the scheduling for one day using a tabu search heuristic. The given data is in form of a set of predefined trips

8.1 Introduction of our heuristic

The distance we want to minimize is the distance for the unloaded trips, i.e. the trips from the home locations to the supply points, the final trips from the customers back to the home locations and the trips from the customers to the supply points. The predefined (loaded) trips are fixed and must be carried out. In our heuristic the trucks are either fully loaded or empty; fully loaded after visiting a supply point and empty after delivery to a customer. Due to this there do not exist any trips where two or more customers or harvesting points are in series. In Figure 8.1 a route is created on the basis of three predefined trips.

![Diagram of a route](image)

Figure 8.1: An example of a route

In our case we have an initial solution where each route is carried out by one truck and where all predefined trips are, equally divided among the routes. For example if there are 13 predefined trips and two routes, then would trips 1-7 and 8-13 be assigned to route one respective route two. From the initial solution, trips are switched between the
routes, i.e. two routes are selected and a change (improving, non-improving or worse) is made. A change can be performed in three ways. First an improving change that improve the objective value is made. If this is not possible a non-improving change which does not effect the objective value is carried out. Further, if not any of previous mention changes are possible a worse change is carried out which decreases the objective value.

This procedure continues until a given number of iterations is carried out or a given time is reached. There are also different ways to select routes and perform a change. Our strategies are described in sections 8.2 and 8.3.

The trucks in this thesis are stationed at the drivers’ home destinations. Sometimes in a real world application it could be valuable to locate the trucks at larger depots, this can be done if it is profitable for the overall cost. We decided not to take the time to load and unload a truck into consideration in our tabu search, but the time can be added to the travel time, if needed.

### 8.2 How to select routes

From a given initial solution we use two different procedures to select routes. The first procedure, called *Ordinary Tabu search*, investigates all pairs of routes. If there are \( p \) routes, the number of pairs is \( p(p - 1)/2 \). The other procedure, *Random Tabu search*, randomly picks one pair of routes.

<table>
<thead>
<tr>
<th># Route</th>
<th>Predefined trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>2</td>
<td>5 6 7 8</td>
</tr>
<tr>
<td>3</td>
<td>9 10 11 12</td>
</tr>
</tbody>
</table>

Table 8.1: An example of an initial solution

For Ordinary Tabu search changes of trips between route \((1,2)\), \((1,3)\) and \((2,3)\) are investigated. Whereas for Random Tabusearch a randomly chosen pair, e.g. route \((2,3)\), is investigated.

### 8.3 How to change trips

After two routes are selected, the next step is to make a change. When making changes, we choose either *Fast Change* or *Best Change*, where single and double changes followed by pivots are included for both. A single change is a change where a trip from route \( \mathcal{X} \) is moved into the first place in route \( \mathcal{Y} \) and route \( \mathcal{Y} \) returns nothing back to route \( \mathcal{X} \). Whereas for a double change, a trip from route \( \mathcal{X} \) change places with a trip from route \( \mathcal{Y} \). To pivot is to place the involved trip(s) into the most profitable place(s).
Fast Change and Best Change are described and illustrated next. The difference between these is the number of changes that is investigated before a decision is made.

**Fast Change:**
- All single and double changes from route $\mathcal{X}$ and $\mathcal{Y}$ are investigated. A change (improving, non-improving or worse) is made.
- The change is followed by a pivot

**Best Change:**
- All single and double changes from route $\mathcal{X}$ and $\mathcal{Y}$ followed by pivots are investigated. A change (improving, non-improving or worse) is made.

---

**Figure 8.2:** An example of a single and double change and pivots

**Figure 8.3:** Methodology for Fast Change

**Figure 8.4:** Methodology for Best Change
8.4 Summary

We use Ordinary or Random Tabu search with Fast or Best Change. An advantage of using Ordinary Tabu search is that all pair of routes are investigated before a change is made. For Random Tabu search only one pair is investigated. To make a change using Fast Change is less time demanding than Best Change since only one pivot is carried out, whereas all changes that have been investigated for Best Change are followed by a pivot.

8.5 Criteria that affect the search direction

To develop an efficient and well-working heuristic, the method must have the ability to avoid getting caught in local minima. For this purpose we have used two parameters tabu and K described in Section 7.1.1. The first used is to not allow a change back to the previous solution, after that a non-improving change is performed. In other words an updated solution is defined as a tabu solution and may not be visited for a number of iterations.

The second used is to avoid cyclic behaviour, i.e. only a small number of local minima is investigated. To avoid this we introduced a penalty where a "frequency counter" is multiplied by a parameter K. The aim is to add a penalty each time an earlier explored solution is visited, where the penalty cost depends on the frequency. The idea of this strategy is to find new unexplored areas.

The time for carry out a route $j$ is denoted $T_j$. In case the maximum time limit $M$ for the routes is exceeded ($T_j > M$), an extra cost is added to the overall cost. The cost can be formulated as follows: $C \cdot (T_j - M)^2 / M$, where $C$ is a constant. The aim of this formula is to add large penalties for large exceedings of $M$. This leads to that the time of the routes is kept down.
Chapter 9

Results and Conclusions

Here we present the results from our tabu search procedures. We also make comparisons between results for different running times.

9.1 Data

We have used data from different sources in the heuristic. The data is presented in the form of cases, Case One to Case Four. Case One is based on an example with six trucks, ten supply points, 12 demand points and 39 predefined trips. Our data is illustrated in Table 9.1.

<table>
<thead>
<tr>
<th>Case</th>
<th># Trucks</th>
<th># Supply points</th>
<th># Demand points</th>
<th># Predefined trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>10</td>
<td>12</td>
<td>39</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>10</td>
<td>12</td>
<td>195</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 9.1: The Cases

Case Two is five times larger than Case One and consists of similar trips with 195 (5·39) predefined trips and 30 (5·6) vehicles, where each home location has five vehicles.

Case Three and Four are obtained from the forestry company Södra and are based on the results from ordinary transportation planning. The predefined trips in Case Three is produced with backhauling possibilities whereas in Case Four no backhauling possibilities have been consider. As previous mentioned, the aim of creating routes with the backhauling strategy is to reduce the length of unloaded trips and therefore keep the overall cost down. In these cases we have 20 trucks, 80 predefined trips, 30 supply points and four demand points.
Our initial solution for Case One is shown in Table 9.2. This configuration consists of six routes where routes one to three get seven predefined trips each and routes four to six get six predefined trips each. Overtime cost occurs when the route’s maximum time limit M is exceeded. In our heuristic $M = 1500$ and the constant $C = 2$ (see Section 8.5). The overtime cost is included in the route cost.

<table>
<thead>
<tr>
<th># Route</th>
<th>Route cost</th>
<th>Overtime cost</th>
<th>Predefined trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5387</td>
<td>2514</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>2</td>
<td>4714</td>
<td>1992</td>
<td>8 9 10 11 12 13 14</td>
</tr>
<tr>
<td>3</td>
<td>2449</td>
<td>401</td>
<td>15 16 17 18 19 20 21</td>
</tr>
<tr>
<td>4</td>
<td>1712</td>
<td>40</td>
<td>22 23 24 25 26 27</td>
</tr>
<tr>
<td>5</td>
<td>1177</td>
<td>0</td>
<td>28 29 30 31 32 33</td>
</tr>
<tr>
<td>6</td>
<td>1193</td>
<td>0</td>
<td>34 35 36 37 38 39</td>
</tr>
<tr>
<td></td>
<td>16632</td>
<td>= 4947</td>
<td></td>
</tr>
</tbody>
</table>

Table 9.2: Initial solution for Case One

The initial solution for Case Two has a similar appearance apart from the number of trucks and predefined trips. In the initial solutions for Case Three and Case Four, every route has four predefined trips each. Furthermore for these two cases, the vehicles are not connected to a certain home location, instead it is possible to place them at the most valuable supply points. In all cases no consideration is taken to the order of the predefined trips in the initial solution to improve the result.

<table>
<thead>
<tr>
<th>Case</th>
<th>Total cost</th>
<th>Overtime cost</th>
<th>Cost for predefined trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16632</td>
<td>4947</td>
<td>5646</td>
</tr>
<tr>
<td>2</td>
<td>109070</td>
<td>45621</td>
<td>30162</td>
</tr>
<tr>
<td>3</td>
<td>399552</td>
<td>351660</td>
<td>9092</td>
</tr>
<tr>
<td>4</td>
<td>815677</td>
<td>758315</td>
<td>8628</td>
</tr>
</tbody>
</table>

Table 9.3: Costs for the initial solutions

### 9.2 Test cases

In this section we investigate the consequences of using Ordinary and Random Tabu search for both Best and Fast Change. The CPU-times to make one move for Case One is presented below. Later in this chapter we make comparisons between the results for these procedures.

<table>
<thead>
<tr>
<th>Case</th>
<th>CPU-Time (minutes)</th>
<th>Tabu search</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.03</td>
<td>Random</td>
<td>Fast</td>
</tr>
<tr>
<td>1</td>
<td>0.67</td>
<td>Random</td>
<td>Best</td>
</tr>
<tr>
<td>1</td>
<td>0.50</td>
<td>Ordinary</td>
<td>Fast</td>
</tr>
<tr>
<td>1</td>
<td>8.00</td>
<td>Ordinary</td>
<td>Best</td>
</tr>
</tbody>
</table>

Table 9.4: Approximate CPU-times for making one change
9.2 Test cases

9.2.1 Best versus Fast Change

Fast Change investigates fewer combinations compared to Best Change, therefore one change is performed faster with Fast Change than with Best Change.

The purpose of Table 9.5 is to make comparisons between Ordinary Tabu search with Fast and Best Change. The best combination where \( K = 20 \) and \( tabu = 3 \) is used.

<table>
<thead>
<tr>
<th>Case</th>
<th>Change</th>
<th>K</th>
<th>Tabu</th>
<th>CPU-Time</th>
<th>Total cost</th>
<th>Overtime cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Best</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>16632</td>
<td>4947</td>
</tr>
<tr>
<td>1</td>
<td>Fast</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>9261</td>
<td>141</td>
</tr>
<tr>
<td>1</td>
<td>Best</td>
<td>20</td>
<td>3</td>
<td>10</td>
<td>13356</td>
<td>2103</td>
</tr>
<tr>
<td>1</td>
<td>Fast</td>
<td>20</td>
<td>3</td>
<td>10</td>
<td>8956</td>
<td>75</td>
</tr>
<tr>
<td>1</td>
<td>Best</td>
<td>20</td>
<td>3</td>
<td>60</td>
<td>9735</td>
<td>114</td>
</tr>
<tr>
<td>1</td>
<td>Fast</td>
<td>20</td>
<td>3</td>
<td>60</td>
<td>8770</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 9.5: Ordinary Tabu search for Case One

After five, ten and 60 minutes, Ordinary Tabu search with Best Change has an total cost that is not acceptable in comparison to Ordinary Tabu search with Fast Change.

In Table 9.6 the results from Random Tabu search are presented. The purpose of this table is to compare Random Tabu search with Best Change and Fast Change.

<table>
<thead>
<tr>
<th>Case</th>
<th>Change</th>
<th>CPU-Time</th>
<th>Total cost</th>
<th>Overtime cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Best</td>
<td>5</td>
<td>10770</td>
<td>824</td>
</tr>
<tr>
<td>1</td>
<td>Fast</td>
<td>5</td>
<td>8843</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>Best</td>
<td>10</td>
<td>8953</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>Fast</td>
<td>10</td>
<td>8783</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>Best</td>
<td>60</td>
<td>8789</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Fast</td>
<td>60</td>
<td>8754</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 9.6: Random Tabu search for Case One

From Table 9.6, we see that Fast Change obtains a better total cost at all time levels. Due to the results in Table 9.5 and 9.6, no further tests are performed with Best Change in this thesis.

We note the best total cost 8754 is reached after 60 minutes. This cost consists of an overtime cost 4, a cost for the predefined trips 5646 and a cost for the unloaded trips 3104 (8754-5646-4). According to this, the cost for the unloaded trips constitutes about 35% of the total cost.

9.2.2 Ordinary versus Random Tabu search

In this section we compare Ordinary and Random Tabu search with Fast Change. In figures 9.1 and 9.2 the total cost is a function of time respective iterations.
When the total cost is a function of time, Random Tabu search reaches smaller values faster than Ordinary Tabu search (K = 20 and tabu = 3). On the other hand if the total cost is a function of iterations, then the results are reversed. From this we conclude that Ordinary performs more efficient changes, whereas Random performs faster changes.
9.2 Test cases

9.2.3 Varying the parameter K and Tabu

Due to the fact that Random Tabu search does not get stuck in local minima (because routes are picked randomly), we only investigate the consequences for different tabu and K for Ordinary Tabu search. In Table 9.7 several solutions are presented to illustrate how different K and tabu affect the solution. The best solution we found is marked.

<table>
<thead>
<tr>
<th>Iter</th>
<th>K</th>
<th>Tabu</th>
<th>Total cost</th>
<th>Overtime cost</th>
<th>CPU-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>5</td>
<td>3</td>
<td>8825</td>
<td>15</td>
<td>22.73</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>6</td>
<td>8903</td>
<td>6</td>
<td>17.08</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>3</td>
<td>8796</td>
<td>4</td>
<td>22.18</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>6</td>
<td>8828</td>
<td>13</td>
<td>17.01</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>3</td>
<td>8825</td>
<td>15</td>
<td>46.25</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>6</td>
<td>8895</td>
<td>0</td>
<td>33.50</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>3</td>
<td>8770</td>
<td>1</td>
<td>44.47</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>6</td>
<td>8793</td>
<td>13</td>
<td>33.42</td>
</tr>
<tr>
<td>200</td>
<td>5</td>
<td>3</td>
<td>8825</td>
<td>15</td>
<td>91.87</td>
</tr>
<tr>
<td>200</td>
<td>5</td>
<td>6</td>
<td>8895</td>
<td>0</td>
<td>66.62</td>
</tr>
<tr>
<td>200</td>
<td>20</td>
<td>3</td>
<td>8770</td>
<td>1</td>
<td>88.93</td>
</tr>
<tr>
<td>200</td>
<td>20</td>
<td>6</td>
<td>8776</td>
<td>1</td>
<td>65.28</td>
</tr>
</tbody>
</table>

Table 9.7: Effects from different K and tabu for Ordinary Tabu search

We note that the heuristic finds less profitable values for K = 5. To increase K gives us better values, where K = 20 and tabu = 3 is the most profitable combination. The appearance of the marked solution is presented in Table 9.8 (compare this and the initial solution in Table 9.2).

<table>
<thead>
<tr>
<th>Route #</th>
<th>Route cost</th>
<th>Overtime cost</th>
<th>Predefined trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1433</td>
<td>0</td>
<td>12 21 16 30 32 27</td>
</tr>
<tr>
<td>2</td>
<td>1408</td>
<td>0</td>
<td>19 9 20 1 24</td>
</tr>
<tr>
<td>3</td>
<td>1507</td>
<td>1</td>
<td>28 14 7 8 5 18 3 37</td>
</tr>
<tr>
<td>4</td>
<td>1430</td>
<td>0</td>
<td>31 13 6 25 36 23</td>
</tr>
<tr>
<td>5</td>
<td>1495</td>
<td>0</td>
<td>39 10 2 33 11 22 35</td>
</tr>
<tr>
<td>6</td>
<td>1497</td>
<td>0</td>
<td>17 4 26 34 29 38 15</td>
</tr>
</tbody>
</table>

Table 9.8: Solution to the marked example in Figure 9.7
9.2.4 Larger case

The aim of Case Two is to investigate the consequences of the solving process when a given problem is enlarged. The CPU-time for making one change affects when the number of vehicles increases for Ordinary Tabu search. From Section 8.2 we know that 30 vehicles generates $30(30 - 1)/2 = 435$ combinations of routes compared to 15 combinations for Case One. To search through these combinations is too time demanding when using this method for Case Two. Accordingly we use Random Tabu search. In Table 9.9 we present the results for Case Two after five, ten and 60 minutes.

<table>
<thead>
<tr>
<th>Case</th>
<th>Change</th>
<th>CPU-Time</th>
<th>Total cost</th>
<th>Overtime cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Fast</td>
<td>5</td>
<td>54736</td>
<td>2919</td>
</tr>
<tr>
<td>2</td>
<td>Fast</td>
<td>10</td>
<td>53167</td>
<td>2083</td>
</tr>
<tr>
<td>2</td>
<td>Fast</td>
<td>60</td>
<td>51785</td>
<td>1533</td>
</tr>
</tbody>
</table>

Table 9.9: Random Tabu search for Case Two

Here, the unloaded trips from the best solution constitute about 39% of the total cost. The figure below illustrates the procedure over 60 minutes for Random Tabu search with Fast Change.

![Figure 9.3: Random Tabu search with Fast Change](image)

As indicated in Figure 9.3, the improvement of the total cost decreases after about ten minutes and a more stabilized behaviour follows. We also note that the graph for Case Two in Figure 9.3 is more extended compared to the graph in Figure 9.1 (where Random Tabu search with Fast Change is used).
9.3 With Backhauling vs Without Backhauling

The cases based on the results from ordinary transportation planning with (Case Three) or without (Case Four) backhauling are presented next. Due to the size of Case Three and Case Four we use Random Tabu search with Fast Change. The best solutions found during 1000 iterations are given in the table below.

<table>
<thead>
<tr>
<th>Case</th>
<th>Iter</th>
<th>Total cost</th>
<th>Overtime Cost</th>
<th>CPU-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1000</td>
<td>17027</td>
<td>33</td>
<td>15,55</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>17113</td>
<td>22</td>
<td>15,72</td>
</tr>
</tbody>
</table>

Table 9.10: Random Tabu search with Fast Change

Even though the solutions were affected by the random procedure, the best solutions found for Case Three was somewhat less than the best solution for Case Four.

9.4 Comparison and conclusion

To estimate how close to the real optimum our solutions get, we need to know the optimum. Due to the difficulties in finding this, we only compare the results from the heuristics to each other.

Results show that it is more effective to use Fast Change than Best Change. Random Tabu search with Fast Change performs one change (for Case One) much faster than Random Tabu search with Best Change. Even though Best Change performs more effective changes than Fast Change does, this process is too time consuming. From now on only Fast Change will be used.

We see that Random Tabu search is more efficient than Ordinary Tabu search. To vary the parameter K and tabu affected both the CPU-time and the total cost, but Random Tabu search was always more effective.

Random Tabu search gives us a good solution for Case One within 10 minutes, where a good solution means a solution near the best found. For the enlarged case, the time for making one change is nearly 15 minutes for Ordinary Tabu search and 2 seconds for Random Tabu search. Due to this, the last method is used. We note that the total cost decreases fast at the beginning and that only small improvements are made after about ten minutes for Case One. According to this, Random Tabu search can also be used to find good solutions within a reasonable time for Case Two.

For the best solutions found, the cost for the unloaded trips constitute about 35% of the total cost for Case One and 39% of the total cost for Case Two. To estimate these values are difficult, but in that case transports would be performed as in procedure 1 in Section 6.1 the cost would be more than 50% of the total cost.

The solutions from Case Three was supposed to become more efficient than the solutions from Case Four. There was some difference between the results, where the best total cost achieved for the case with backhauling trips was only marginally better than the
best total cost for the case without backhauling trips. Why the difference was not larger is difficult to say. Maybe the number of vehicles, trips, supply points, demand points was not divided proportionally.

Random Tabu search with Fast Change gave us the best solutions (schedules) for all cases. Even for the larger case, good solutions were reached within a reasonable amount of time. The investigations of Case Three and Case Four using this heuristic gave us a lower overall cost for Case Three even though the costs for the predefined trips in Case Three was higher than in Case Four.

9.5 Further work

The most valuable factor in acquiring a good solution is to perform one change in as short a time as possible. A very interesting thing to try next would be to implement a tabu for pair of predefined trips when using Random Tabu search. This would lead to less combinations to search through and therefore faster changes.

In this thesis no consideration is taken to the order of the predefined trips in the initial solution. It seems probable that the solution time can be kept down if we structure the predefined trips from the same area into the same route from the beginning.

Furthermore, we decided to use all vehicles; no one has permission to stay at home. To allow trucks to stay at home would probably in some cases be profitable for the result.


Bibliography


