Linköping Studies in Science and Technology
Dissertations No. 967

Optimal Truck Scheduling – Mathematical Modeling and Solution by the Column Generation Principle

Myrna Palmgren

Linköpings universitet
INSTITUTE OF TECHNOLOGY

Division of Optimization, Department of Mathematics
Linköping Institute of Technology, S-581 83 Linköping, Sweden

2005

ISBN 91-85457-17-5   ISSN 0345-7524

Abstract

We consider the daily transportation problem in forestry which arises when transporting logs from forest sites to customers such as sawmills and pulp and paper mills. Each customer requires a specific amount of a certain assortment, and the deliveries to the customers can be made within time intervals, known as time windows. Further, there are a number of supply points, each with a certain assortment, and a number of vehicles of a given capacity, to be used for transport.

The log truck scheduling problem consists of finding a set of minimal costs routes, one for each vehicle, such that the customers' demands are satisfied without exceeding the supplies available at the supplies. Each route has to satisfy a number of constraints concerning time windows, truck capacity, timetable of the driver, lunch breaks, et cetera. The model used to describe the log truck scheduling problem is based on the route concept, and each variable, or column, represents one feasible route. Since the number of feasible routes is huge, we work only with restricted versions of this problem, which are similar to restricted master problems in a Dantzig-Wolfe decomposition scheme.

We use three solution methods based on the column generation principle, together with a pool strategy which allows us to deal with the feasible routes outside the restricted master problem. The three methods proposed have a common structure; they use branch-and-price together with a column generator, followed by branch-and-bound. The column generators in the three methods differ. In the first method, the subproblem is based on a cluster-first-route-second strategy. The column generator in the second method involves solving a constrained shortest path problem, and finally, the third method builds on a repeated generation of clusters and routes.

The three methods are tested on real cases from Swedish forestry companies, and the third method has been adapted to a computerised system that utilises the Swedish national road data base, for computing travelling distances. The results obtained show that the optimisation methods succeed in finding significantly better solutions than those obtained by manual planning, and in a reasonable computing time.

Acknowledments

First and foremost, I would like to thank my advisor, Torbjörn Larsson, who helped me climb this mountain. He supported me enormously when I needed it the most. Without his help and encouragement you would not have been holding this thesis in your hand right now. I also thank Stig Danielsson, Peter Värbrand, and Bengt Ove Turesson for their optimism and support.

SkogForsk provided me with the test cases and Professor David Ryan inspired the work in Chapter 6.

Thanks also to colleagues and former colleagues at the Department of Mathematics at Linköping University, including of course Andreas Westerlund, Peter Broström, Jörgen Blomvall and Kennet Melin who helped making the good times more enjoyable and the bad times less unbearable.

My deepest gratitude to the monastery of St George in Mount Lebanon for their never ceasing prayers.

All my love to my wonderful family. Kimberly (who made the drawing on the front cover), Daniella, Rebecca and Jonathan - the coolest kids alive. Pär – nothing I write could ever be enough so I will only write this. My wonderful parents and brothers. I love you warriors!

Linköping, August 2005

Myrna Palmgren

Table of Contents

1 Forestry and operations research in forestry

1.1 Significance of forestry

Throughout human history, forests have had an essential role as a source of nutrition, fuel and timber, and also as a place for recreation, a subject for poetry, et cetera. Forests are of great emotional and economic importance to people in general and to the Swedish people in particular. The large forest areas that cover approximately 60% of Sweden offer immense opportunities for outdoor activities, such as picking mushrooms and berries, orienteering and hunting, since they are also the habitat of wild animals. Moreover, it is important to note the crucial role of forests in absorbing the carbon dioxide which pollutes the air we breathe and influences the global climate.

In Sweden, forestry is a base industry around which many other industries have developed, and it is therefore a driving force in the Swedish economy. The Swedish forest industry includes the pulp and paper industry, the sawn timber industry and the board industry. Sweden is the fourth largest exporter of pulp and paper in the world, and the second largest exporter of sawn timber. The forestry sector not only provides direct employment, but also provides work indirectly to many companies such as the information technology companies which develop process control systems, transport companies, companies that operate the forest machines, and service companies. According to the National Board of Forestry (Skogsstyrelsen), forestry and forestry products represent 20% of the total Swedish export income in 2003, and the number of people directly employed in forestry companies is about 100,000.

In Sweden, individuals own about 52% of the total forest area, while forestry companies own 24% (Swedish Forest Industries Federation, Skogsindustrierna, 2003). These companies naturally focus on the economic aspects of forestry, but are also expected to preserve nature and wild life, see Report 5 (1993), and therefore, they invest in studies and research in order to meet two goals: profitability and sustainability. The two goals of profitability and sustainability need not be in conflict, but are in fact complementary, since in the long run the only way to increase profit is by preserving nature. Fryk (in Report 5) stresses the importance of ecological and environmental considerations, and, today, many decisions have been taken in an attempt to reach the goal of sustainability. For example, forestry companies have set aside large areas in order to preserve the fauna and flora of the forest.

Today, long term planning has become a necessity in forestry, since managers wish to anticipate future problems and plan the future, rather than solve problems as they arise. As a result of this planning strategy, each phase of the rotation period of a stand is today carefully planned. One rotation period is between 70 and 120 years long in Sweden and consists of several phases: regeneration, which includes planting by sowing or natural regeneration, cleaning and thinning, and finally felling or harvesting. Since the consequences of the measures taken today have to be dealt with in 70 to 120 years, it is crucial to take the right ones. Some of the steps that are planned are the choice of seeds, the choice of forestry nutrition and the choice of regeneration methods. It is of course also of importance to use high quality forestry machines that cause minimal damage to the ground. One of the goals and also one of the consequences of sustainable forestry, is an increasing profit. A healthy forest that is well planned and well kept gives high quality wood, and this in turn positively affects profits. However, environmental issues that need to be taken into account in sound forestry can cause additional costs. In addition, international competition and increasing quality requirements from customers are all challenges that tend to decrease profitability.

The products or assortments obtained from forestry raw materials can be divided into three main groups: fuel-wood, pulpwood and saw logs. Each of these three main groups is in turn divided into a number of assortments, depending on properties such as size, quality, proportion of rot and species. A key factor that characterises customer demands is that highly specific assortments have to be delivered within specific time periods, since a shortage of raw material will lead to an industry having to interrupt. Another key factor is the requirement for fresh wood, and in order to maintain its freshness, storage time must be minimised. An example of this is pulp logs, which are sensitive to storage time, since their quality deteriorates over time. In this case, it is therefore necessary to maintain a continuous supply of pulp logs throughout the year. This is actually the case for most kinds of wood supply, which means that only a short storage time is generally allowed.

The profits of forestry companies are increasingly dependent on their ability to fulfil the requirements of the customers, and therefore forestry companies are tending to become increasingly customer oriented. As a result of this, today customer demands directly influence the harvesting plans of the companies. Among the consequences of a more customer-oriented policy are:

- Increased number of assortments, since the customer demands are becoming more specific.
- Increasing request for fresh wood because of higher quality requirements.
- Increasing request for precision in delivery time in order to streamline production.

The increasing customer requirements are obstacles for a greater profit; therefore the forestry companies are studying several aspects an attempt to make savings. With respect to the development of harvesting machines, it is considered that machine development has already reached its peak. However, great potential for making cost savings is expected if the tradition of solving different planning problems within forestry independent of each other is changed, and replaced by a more integrated planning strategy. An example of such an integrated planning problem is: given the condition of the present stand and the estimated future demands of the customers, how should the harvesting and transportation be planned in order to reduce overall costs?

Sophisticated planning may help reduce the costs by making more use of machines, people, forestry methods, et cetera. Efficient computerised systems may offer support to the planner, even if they are not the sole answer to the planning problem. Operations Research (OR) and optimisation methods can of course play an important role in solving many forestry management problems.

1.1 Operations research in forestry

Operations research is the scientific study of how to use mathematical models and mathematical algorithms to analyse and find solutions to decision problems and complex management situations. The tools that OR provides are in use in a wide range of areas such as finance, supply chain, medicine, staff scheduling, timetabling, network design, transportation, mechanics, and also in many applications within the forest industry (e.g. Epstein et al. 1999). The list below gives some examples of the various areas within forestry where OR is used.

- Roll cutting at paper mills.
- Strategic harvest planning over several rotations.
- Transportation planning.
- Long term production planning.
- Stock level planning.
- Evaluating the impact of environmental measures in forest harvesting.
- Investment in forest roads.
- Log bucking.
- Forest fire management.
- Optimisation of the supply chain.

Some of the applications mentioned in the list, like log bucking and roll cutting, are very specific whereas others are more general in the sense that they involve several coupled decision problems. Examples of such applications are harvest planning and supply chain planning.

Harvest planning (e.g. Epstein et al. 1999) consists of deciding which areas to harvest during the next planning period, and allocating crews and machines for this purpose. Here, many factors such as road conditions, weather, and transport costs need to be considered. This problem tends to become very complex because of the various considerations that have to be dealt with simultaneously. Because of this, OR methods are of great help and support to the managers.

Optimisation methods are also used in the forest during the bucking operation (e.g. Näsberg 1985). Harvesters are machines with two functions; they fell and buck the trees. A computer onboard the harvester measures different properties of the tree trunk and maximises the benefit of the logs in order to decide the most valuable assortment or product for each tree trunk. For example, the system measures the diameters at several positions along the stem, and then uses information about the existing assortments and their current market value to decide how to cut the tree into logs optimally. These computer programs rely on mathematical optimisation methods as a basis for their decisions.

Another way to increase profitability is to integrate decisions in several stages of the forestry industry. This results in a class of decision problems known as supply chain management (e.g. D’Amours, Frayret and Rousseau 2004 or Lohmander and Olsson 2003). Supply chain refers to the distribution channel of a product, from its sourcing to its delivery to the end consumer. For example, let us consider a company that buys wood from several suppliers, stores it and uses it to make some final products. Then the company stores these final products and distributes them to customers. The integration of these planning phases is one example of a supply chain problem. Figure 1 shows an example of the supply chain of a forestry company. It shows the flow of products between the different units: harvesting points, saw mill, pulp mill, storage, heating plant, and the unit where paper is recycled.

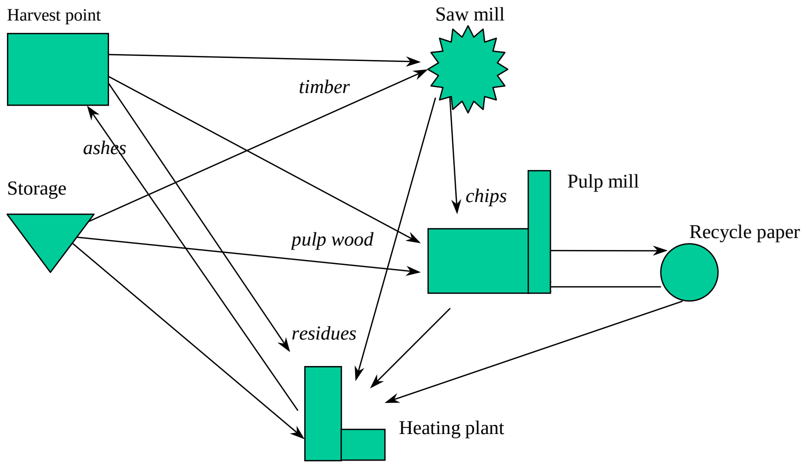


Figure 1: Different components in the supply chain of a forestry company

Another forestry problem, and one that has been studied since the sixties in Canada and the USA, is forest fire management and problems related to it (Martell 1982). Fires have always been considered as a threat to the lives of people working or living nearby the forest. At the same time, forest fires constitute a natural means for the healthy development of the forest. For these reasons, fire management attempts to reach a balance between the threats and the benefits of forest fires. One of the problems that is studied within fire management is the problem of deciding where to locate the airtanker bases in order to maximise the chances of efficiently extinguishing the fire (e.g. Martell and Tithecott 1991). Airtankers, which are used for example in Canada, fly to nearby lakes to collect water and drop it over the fire. The idea is to locate airtankers close to areas where they are most probably needed in order to minimise matters such as the transportation distance and the crews’ working time.

The climate in Sweden has a damaging effect on the forest roads; during spring, roads may become unusable when the ground is thawing, and during autumn, the roads often become muddy. These damaging effects and the blocking of the roads that they cause are handled by investment strategies that improve the quality of the roads. Forestry companies decide firstly on, the road segments they want to invest in, and secondly, the type of investment (e.g. Olsson 2004). In order to make these decisions, forestry companies take into consideration the customer demands and the risk and cost of keeping large stocks, which should be avoided.

1.2 Introduction to the log truck scheduling problem

As mentioned earlier, the Swedish forest industry is facing a number of challenges: precise customer requirements, environmental challenges, and an increasing world competition. In order to maintain the competitive edge, the forestry industry has to continuously increase its productivity, develop competitive products, and make customer orientation a priority. Customer orientation means taking into account the customers' requirements for the end products already at an initial stage of the wood procurement process. The wood procurement process includes several stages: breeding, planting, harvesting and transportation.

Transportation, which is the stage where there is greatest potential for savings, is the subject of this thesis. The transportation of logs is performed in two stages; transport from forest to road sides and transport from road sides to terminals or customers (such as sawmills and pulp mills). In the year 2002, 45 million tonnes of logs were transported by logging trucks, while 5.4 million tonnes were transported by rail (National Board of Forestry, Skogsstyrelsen). The transportation of logs is very costly and it forms a high percentage, in the range of 15 to 20%, of the cost of timber. In this thesis we consider the daily planning of the transportation of logs by truck, from forest road sides to customers.

The problem of planning the transportation from forest road sides to customers is complex. The goal of the transportation planning is to construct schedules for the trucks, such that all customer demands are satisfied on time. In Sweden, different companies plan their log transportation by different means; some let the drivers decide their own schedules, while others try to plan the whole schedule at a higher level in the company. The most common means is manual planning done by experienced transport managers or drivers. Manual planning is however very time consuming and inefficient, and transport managers usually work under stress.

The use of computer aided planning systems in countries like Chile and Finland has proved to have many advantages (e.g. Epstein et al. 1996). These include shorter travelling distances, less pollution, a higher percentage of satisfied customers, and better working conditions for the drivers. The potential savings are estimated to approximately 10% of the total transportation costs. Besides that, a planning system gives both better means for handling and overviewing data and helps visualising the problem, and therefore it helps the transport managers to a better understanding of the problem. In Sweden, the use of computerised planning systems is still limited. However, new opportunities are made possible today by the newly constructed Swedish National Road Database (Nationell VägDataBas or NVDB), which contains up-to-date information over the entire Swedish road network (information on http://www.vv.se/nvdb/en/index.asp).

There are two approaches for handling daily planning, namely despatching and scheduling. Despatching is the creating of a plan in real time, while scheduling solves the problem some time in advance, for example one day ahead. In the thesis, we consider the latter approach, which we refer to as the Log Truck Scheduling Problem (LTSP). To be precise, the LTSP is the problem of finding *routes*, one for each truck, in order to carry out the transportation from supply points in the forests to customers, at minimal cost. Further, a *route* is a sequence of pickups and deliveries: pickups at harvesting areas followed by deliveries to customers. A route is called *feasible* if it satisfies a number of rules or constraints required by the customers or by the nature of the problem itself. Each truck has a home base, and the cost of a route is determined by the distances travelled and the truck loads.

The LTSP is related to the well studied Vehicle Routing Problem (VRP). In fact, it is a generalisation of the VRP and therefore also NP-hard. It generalises the VRP in several respects. In particular, the LTSP involves two sorts of operations, pickups and deliveries, and, further, it includes split pickups and split deliveries. It is referred to as a scheduling problem, since a time component is considered; all operations must be performed within given time windows associated with service points (pickup/delivery points or home bases). The vehicles considered in the LTSP are logging trucks of different types and capacities, that is, the vehicle fleet is heterogeneous.

The LTSP has been a subject of research in countries with a significant forest industry, like Chile, Finland and Sweden. In Chile, for example, a computer aided system for planning the daily transportation has been in use since the early nineties (Epstein et al. 1996). Most methods incorporated in such systems are heuristics. Such heuristic methods have the advantage of finding acceptable (feasible and hopefully near-optimal) solutions to the problem in a short computing time. However, these heuristic methods are typically very specific for the local rules for transport, and are therefore hard, or impossible, to apply under other conditions or in other countries. Besides that, heuristics, contrary to optimisation based methods, do not give any quality guarantee for the solutions found.

In this thesis, we study a generic mathematical optimisation model for the LTSP and design three methods based on column generation for its solution. Each *feasible route* gives rise to a *column* in the generic model, and a corresponding binary *variable* describes whether the route is used or not. The feasible routes are generated either a priori or during the solution process, or both. As mentioned above, a route is feasible if it obeys a number of rules or constraints, such as time windows, precedence (of pickups to deliveries), and capacity constraints. The key advantage of the generic model is that its structure does not change when the local rules concerning the routes change. The number of variables in the generic model however is enormous, and hence, it is practically impossible to enumerate all feasible routes. We therefore work with a restricted version of the model, involving only a small subset of the variables, that we refer to as the *Restricted Master Problem, RMP*. The main decision involved then is to choose exactly one feasible route for each truck in order to carry out the transportation and satisfy all customer demands without exceeding the available supplies.

The three solution methods used in the thesis are based on exact, that is, optimising, methods, but where certain sub-problems are solved either heuristically, or approximately, or exactly. The methods can be characterised as optimisation-based heuristics, since they are founded on solid optimisation methodologies. The advantages of the optimisation based methods are their generality and their ability to produce lower and upper bounds for the objective value, which helps measure the quality of the solution. These optimisation based methods are however time-consuming, which of course is a drawback since the planning is made daily. However, the running time of these methods has been shortened considerably since the work on this thesis began, due to the rapid development of the hardware and the software used.

The methods presented in the thesis can be used as a part of the operative planning tools. They can also be used in simulation studies for analysing different scenarios obtained by changing the conditions in a planning situation. In addition, a near-exact method presented in the thesis can be used for evaluating heuristic based methods, since it offers a way of getting lower and upper bounds to the optimal cost of transporting logs.

1.3 Outline of the thesis

The thesis is organised into seven chapters. In Chapter 1, we give the forestry background, describe the wood procurement process, and give examples of forestry problems that have been approached with operations research techniques. In Chapter 2, we describe the real world planning situation that gives rise to the log truck scheduling problem and give a detailed explanation of the data needed in the problem. Then we present two ways for modelling the LTSP mathematically. The first model includes three types of variables: flow, time and load variables. The second model, in which the variables represent feasible routes, is a generalisation of the well-known set partitioning problem.

In order to formulate the second model, we relate feasible routes with columns and variables. The number of variables in the second model is huge; we estimate this number in a small example, for the purpose of illustrating the complexity of the problem. This model includes a set of global constraints that apply to all variables, that is to say routes, and these are explicit in the model. It also includes a large number of local constraints that are the rules that define which routes are feasible and not. These local constraints are implicitly taken care of. We give examples of such rules and illustrate the concepts of split pickups and deliveries. Chapter 2 ends with a comparison between the two suggested mathematical models and with a discussion concerning the drawbacks and merits of the respective models.

Chapter 3 compares the LTSP with some other routing problems. We classify the problem and discuss its relation to similar problems, such as the Travelling Salesman Problem (TSP), the Vehicle Routing Problem with Time Windows (VRPTW), the Pickup and Delivery Problem with Time Windows (PDPTW), and the highly related Split Pickup and Delivery Problem with Time Windows (SPDPTW). The literature concerning the LTSP is very limited; we give a brief literature review of related problems.

The three methods proposed in the thesis are all based on linear programming relaxation, column generation, that is, route generation, and approximate branch-and-price. A first method for finding acceptable solutions to the LTSP is proposed in Chapter 4. This method is based on an a priori enumeration of a limited set of feasible routes. This set is created by applying a cluster-first-route-second strategy. In Chapter 5, we use an initial limited set of a priori enumerated routes and, in addition to that, we use the dual prices for the global constraints to generate more routes, in a column generation fashion. We refer to the problem of generating feasible routes as the *subproblem*. In Chapter 5 this is stated and solved as a constrained shortest path problem. The third method, which is described in Chapter 6, also takes advantage of the information provided by the dual prices so as to generate new routes. Here, the subproblem involves solving a relaxed problem to find good clusters, within which feasible routes are thereafter enumerated. The third method can be characterised as a repeated cluster-first-route-second strategy. It has been adapted for a system called RuttOpt, which is a first step towards a computer aided system that can be used by transport managers and transport companies. (RuttOpt can be used to test and evaluate the optimisation based methods, since it gives a graphic illustration of the results which is easy to interpret and analyse.) In Chapter 7 we discuss the different conditions and rules that arise at different forestry companies and propose ways to deal with these differences. Finally, we discuss future prospects for improvements and developments.

1.4 Overview of the methods proposed and their relation

The three solution methods for the LTSP proposed in the thesis all work on restricted versions, RMP, of the generic mathematical model. As mentioned before, the problem we solve is to find one route for each truck in order to satisfy customer demands without exceeding the given supply, and at minimal cost. Choosing exactly one route for each vehicle is a binary decision, and the RMP involves both binary and continuous variables, the latter describing a surplus or deficit of logs. The restricted master problem is therefore a mixed integer problem. The main difference between the three methods lies in the subproblems, that is to say the way of generating feasible routes.

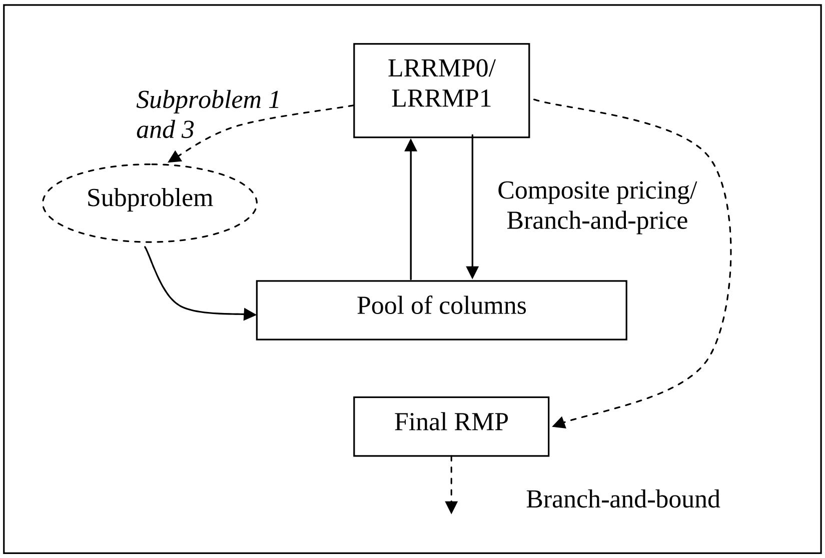


Figure 2: Common structure of all three methods.

The basic idea in all three methods is to start by solving an initial Linear Relaxation of the RMP (LRRMP), and to end by solving the final RMP to (near-) optimality. The three methods differ in how they generate the feasible routes of the model; in the first method, a limited set of feasible routes is generated a priori, in the second method, the routes are generated within the solution process by solving constrained shortest path problems, and finally, in the third method, routes are generated heuristically within the solution process. The feasible routes generated according to any of the three methods are stored in a *pool*. All three methods involve the same three phases: a first phase where *composite pricing* (which is similar to partial pricing) is applied, a second phase based on *branch-and-price* and finally, a third phase where *branch-and-bound* is used.

Composite pricing consists of computing the reduced costs of all variables corresponding to the feasible routes in the pool and returning the variable(s) with the negative reduced cost(s). In all three methods, this first phase starts by solving the Linear Relaxation of an *initial* RMP, LRRMP0. This initial problem only contains columns that represent "empty routes", where each vehicle stays at the home base during the whole day. The initial dual prices obtained after solving LRRMP0 are used to start the composite pricing procedure with respect to the limited set of variables stored in the pool. Composite pricing allows us to limit the size of the problem, LRRMP, since the pricing of the variables is done outside the problem and only variables with negative reduced costs are added to LRRMP. We refer to the problem obtained at the end of the first phase as LRRMP1.

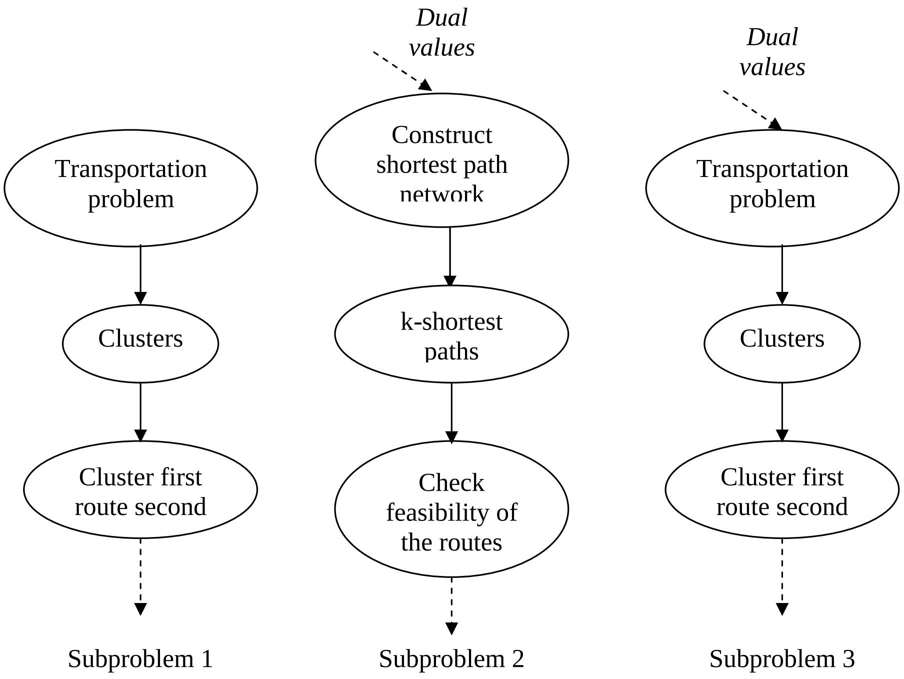


Figure 3: The three subproblems

In the second phase, a branch-and-price scheme is applied to LRRMP1. In this scheme we use constraint branching, which is especially suitable for set partitioning type problems. Branch-and-price allows new columns to be added to LRRMP1 from the subproblem. Pricing is also applied to all the columns currently stored in the pool.

When the second phase is interrupted, we have obtained a final mixed integer problem: Final RMP. Figure 2 illustrates the main idea behind the three methods; the difference between them lies in the subproblem and Figure 3 shows the different subproblems used in the three methods of the thesis.

1.1.1 The first solution method

The first solution method, which is presented in Chapter 4, is based on a priori enumeration of a large set of feasible routes. Since the number of feasible routes is enormous, we use a heuristic for enumerating only a relatively limited, but large, number of routes. The strategy of the heuristic is cluster-first-route-second. In order to build clusters that help construct high quality routes, we solve a transportation problem which includes all supply and demand points. In this way we get a transportation solution that connects a number of supply points to each demand point. These connections define the clusters. Feasible routes are afterwards enumerated within these clusters and are stored in a pool.

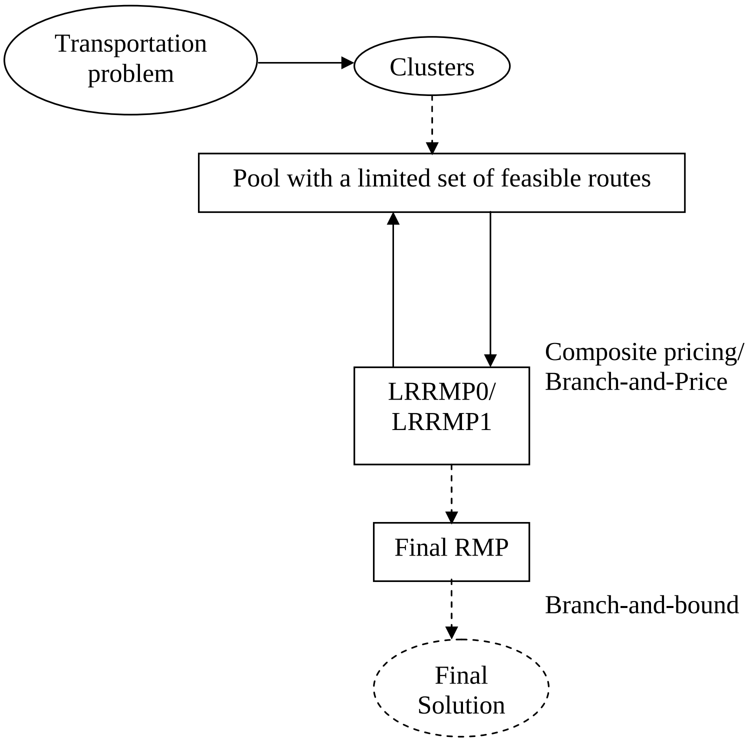


Figure 4: First solution method

After solving LRRMP0 and generating a set of feasible routes through this cluster-first-route-second strategy, the first phase outlined above can start. We iterate between the LRRMP and the pricing of all columns in the pool until no more variables with negative reduced costs are found in the pool. We have then succeeded in solving the LRRMP, within the limited set of variables, and at the end of the first phase we get LRRMP1. For the purpose of finding integer solutions we use branch-and-price, where the pricing is again performed within the set of variables in the pool. Since branch-and-price is time consuming, we interrupt the pricing after a predefined number of iterations. The Final RMP, obtained by the addition of variables through composite pricing and branch-and-price, is solved by branch-and-bound.

1.1.2 The second solution method

In the second method, which is described in Chapter 5, a constrained shortest path problem is solved in order to obtain new routes that are added to the pool. As in the first method, composite pricing, branch-and-price, and branch-and-bound are the procedures used, starting with composite pricing and ending with branch-and-price and branch-and-bound. In both the composite pricing and branch-and-price procedures, the dual prices for the constraints in LRRMP are used as components in arc costs in a network in which most of the constraints concerning the routes are embedded. This network is obtained by discretising the possible quantities that can be picked up or delivered by the truck. The possible clock times at which different operations can be performed are discretised in a similar way.

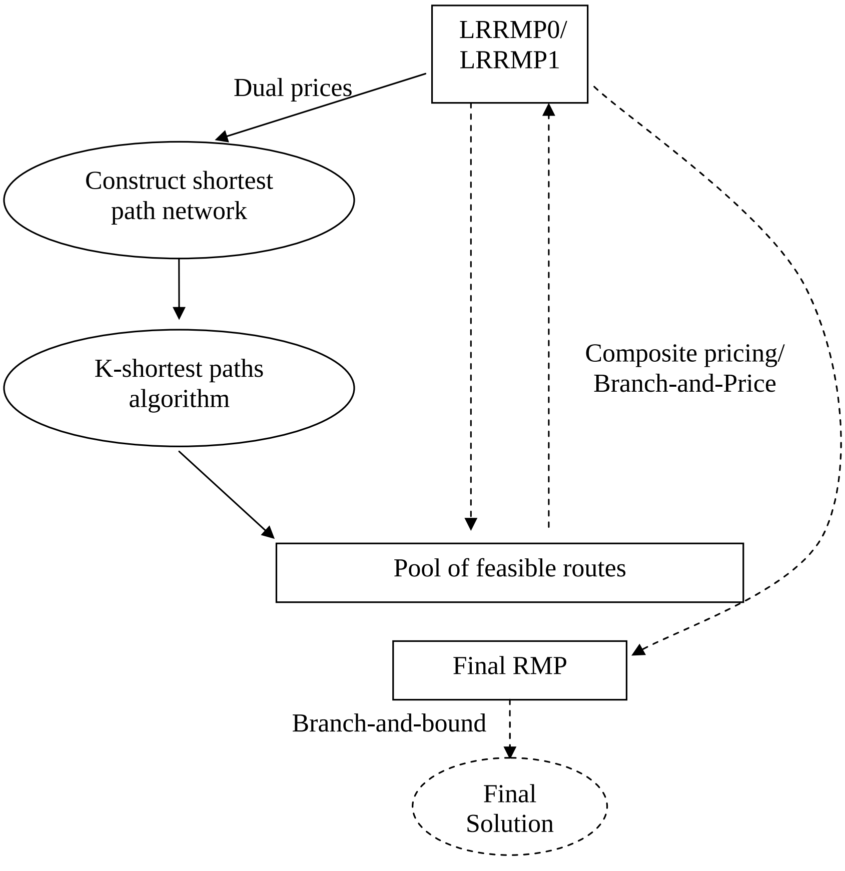


Figure 5: Second solution method

One constraint is thought difficult to implicitly take into account in the network, this is the constraint that ensures that the total amount picked up at a supply point during the route does not exceed the amount available. For this reason, we solve a constrained shortest path problem by applying a k-shortest paths algorithm in the network described above. This algorithm finds the first, second, …, up to the k^{th} shortest path, where *k* is a predefined integer number. Those shortest paths that fulfil the constraint on total available supply are added to the pool; they constitute candidate variables to be added to the LRRMP. The final integer problem, Final RMP, obtained by adding variables through composite pricing and branch-and-price, is solved by branch-and-bound. The important difference between the methods of Chapters 4 and 5 respectively is that in the former, the dual prices are only used *to compute the reduced costs* of already generated variables, while in the latter method, the values of the dual prices are used *to generate new variables.* The second solution method is illustrated in Figure 5.

1.1.3 The third solution method

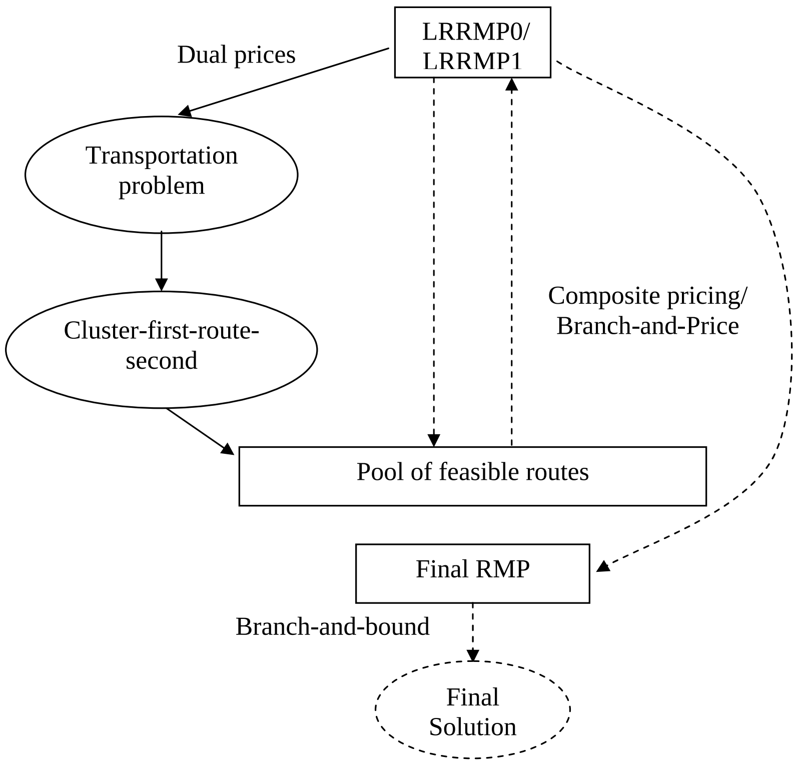


Figure 6: Third solution method

The third method, presented in Chapter 6, combines ideas from both Chapter 4 and Chapter 5. The advantage of the first method is its short computing time in comparison with the long computing time required to solve the constrained shortest path problem in the second method. At the same time, the solution quality of the first method is restrained by the quality of the routes enumerated a priori. This weakness is avoided in the second method, since the columns are there generated during the solution process by the solution of a near-exact subproblem that takes advantage of the information provided by the dual prices. Like the latter method, the third method also takes advantage of the information provided by the dual prices. At the same time it also takes advantage of the short computing time required to enumerate a limited number of routes instead of solving the constrained shortest path problem. In this way, the third method combines the advantages of the other two methods.

The basic idea, and the one that is illustrated in Figure 6, is to repeatedly use the current values of the dual variables for the constraints in LRRMP to modify the costs in the transportation problem. Each transportation problem is solved in order to create clusters in the same way as in the first method, and the clusters are used to enumerate new feasible routes that are added to the pool. All feasible routes in the pool are priced during the composite pricing and branch-and-price procedures. Branch-and-bound is again used to solve the final integer problem, Final RMP. The method in Chapter 6 has shown to be more robust than the first method and faster than the second method. It has been adapted to a planning system prototype and tested on real-life data.

1.6 Contributions and significance

The contributions and significance of the thesis are the following:

- We attack a complex scheduling problem of large real-life significance: the log truck scheduling problem, or LTSP.
- We demonstrate that this complex problem can be approached successfully by optimisation-based methods.
- We propose a traditional mathematical optimisation model for the log truck scheduling problem based on flow, load, and time variables.
- We use the traditional model to classify the log truck scheduling problem and relate it to other known routing problems.
- We give a mathematical formulation of the subproblem obtained when applying price-directive decomposition to the traditional mathematical model. Further, this subproblem is transformed into a constrained shortest path problem.
- We consider a column generation-oriented formulation for the LTSP, where the columns correspond to feasible truck routes, and propose three methods for its solution. In two of these the column generation is made heuristically by cluster-first-route-second strategies. The third method is a near-exact column generation scheme.
- Our approaches provide means for obtaining lower and upper bounds to the optimal value of the LTSP. This is useful for benchmarking heuristic solution methods for LTSP.
- We study route pool strategies as a means for reducing the computational effort.
- We develop branch-and-price schemes, with constraint branching, for the solution of the integer problem.
- We adapt one of the proposed solution methods to a planning system.

- We evaluate the three solution methods by applying them to real-life instances of the LTSP from Swedish forestry companies; the characteristics of these instances are very different from each other.
- We show that is possible to find good feasible solutions to real-life instances of the LTSP by using optimisation-based heuristics which utilise route-generation.
- Based on the conclusion drawn, we point out some promising directions for future work in this field.

Chapters 4 and 5 are based on material that has been published in Palmgren et al. (2002), and Palmgren et al. (2003), respectively, and on material from Palmgren (2001). These papers have been presented at the following conferences: IFORS'99, China, 1999; Nordic MPS, Sweden,1999; CO2000, England, 2000; INFORMS, USA, 2001; Nordic MPS, Denmark, 2001; Syposium on Models and Systems in Forestry, Chile, 2002; IFORS 2002, Scotland, 2002. Material from Chapters 6 and 7 have been presented at the following conferences: ROUTE 2003, Denmark, 2003; ISMP 2003, Denmark, 2003; The 2003 Symposium for Systems Analysis in Forest Resources, USA, 2003.

2 The Log Truck Scheduling Problem

2.1 Forestry transportation problems

The problem of planning the wood flow in the forestry industry involves several time horizons, and it is in general treated on three different levels: strategic, tactical and operative.

2.1.1 Strategic and tactical transportation planning

Strategic transportation planning deals with long term decisions. In this work a strategic planning period is for one year, since contracts between the different parties, like forestry companies and customers, are typically settled for one year at a time. If the customer demands are not known in advance then forestry companies need to estimate the total customer demand for the coming period. Then, they simply decide what harvesting areas to use by balancing supplies against the expected demand and by taking transportation costs into consideration. Here, harvesting planning influences transportation decisions, and visa versa.

On the tactical level, which can for example be a monthly planning, the main decisions involve which of the harvesting areas is going to be used to satisfy the demands of a certain mill. It is also on this level that backhauling possibilities are studied and planned. Backhauling aims at reducing the number of unloaded journeys thus increasing the efficiency of the transportation. This is achieved by searching for possibilities for loading a truck at a point close to the delivery point.

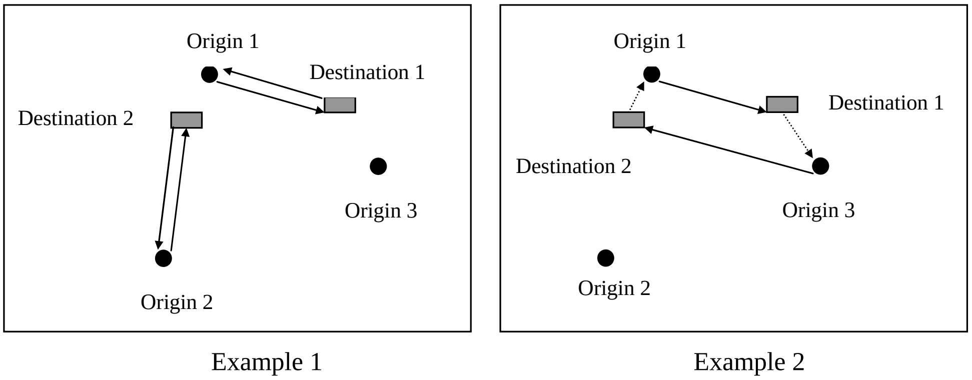


Figure 7: Illustration of the effect of backhauling considerations

Today, most transportation planning is made manually. In order to handle that, the forestry areas are divided in smaller districts and one transport manager is responsible for the transportation planning within each district, independent of the other districts. This makes communication and co-ordination difficult, and therefore it has been hard to increase backhauling, since most of the loaded trips cover different districts. Moreover, most backhauling possibilities are often found within areas belonging to different forestry companies, which make co-operation between these companies a necessity.

Backhauling is obviously an important issue for the daily transportation planning, but it is also important on the tactical level, since the allocation problem that this deals with depends heavily on backhauling decisions. Figure 7 shows two allocations, made with and without backhauling. In the first example, each origin is linked to the nearest destination point whereas in the second, backhauling possibilities lead to a different solution.

Tactical planning, including backhauling has been studied during the last few years. It is explained and discussed in detail in Carlsson and Rönnqvist, 1998. Obviously, the result of the tactical planning is of great importance for the next planning level, the operative, or daily, planning.

2.1.2 Operative transportation planning

Operative planning considers the daily transportation of logs. This transportation is made in two steps, primary transportation which involves moving the harvested logs from the actual felling point to piles located close to the roads, and secondary transportation where logs are transported from the piles at the road sides to their final destination. The primary transportation problem is also known as the log extraction problem.

Extraction of logs

The problem of extracting logs from the harvesting points where the trees are felled to road sides is an important application that is being studied intensively today. Since the yearly extraction costs in Sweden amount to $ 200-250 million (Carlsson, Rönnqvist and Westerlund, 1998), even a small cost reduction can lead to significant savings.

After felling and bucking the trees, the harvester puts the logs into piles at several points in the forest. The forwarder collects the piles and loads them according to some pattern which depends on the assortment type. Once the forwarder is fully loaded, it moves the logs to a larger road that can be accessed by logging trucks. The tracks or roads used by the forwarder are usually the same ones travelled by the harvester while felling. This problem is illustrated in Figure 5 where a harvesting area and the paths used by the harvester are shown.

The objective of the log extraction problem is to find efficient routes in the network defined by those forestry roads. For this purpose, distances and travel times between different points in the network are measured and these are then used in the optimisation methods to find good routes. It is important to note that the routes are dependent on the loading pattern and thus on the number of assortments collected. As the number of assortments increases, the costs of the extraction operation will also increase and thus there is then a need for more efficient routes.

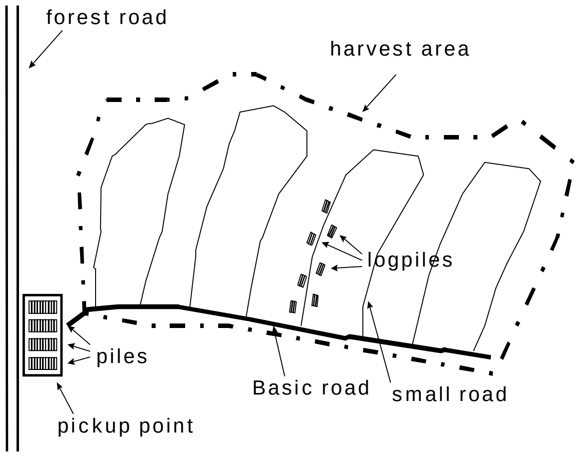


Figure 8: Illustration of the problem of extracting logs

Log transportation

Secondary transportation, that is, the transportation of logs from road sides at the harvesting area to customers, is the operative problem that has received most attention, since it involves large saving possibilities. According to Report 5 (1993), reducing the distance travelled by 1% leads to a 0.75% cost saving and the savings that can be achieved are estimated to about $10 million annually.

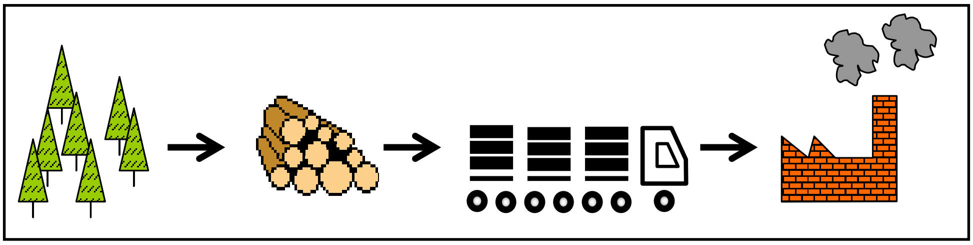


Figure 9: The different links in the daily log transportation

The problem consists in transporting logs from the harvesting area to customers, such as sawmills and pulp mills. Most of this transportation is done by logging trucks. The transport managers assign routes, one for each truck, starting and ending at the driver's home base. A route should respect the drivers' working hours, the customers' opening hours, and the time when service can begin at the harvesting areas. This problem is the subject of this thesis, and in the next section we begin by describing the problem.

1.2 Description of the log transportation problem

The Log Truck Scheduling Problem (LTSP) amounts to finding feasible routes, one for each vehicle, in order to satisfy all customer demands on time without exceeding the available supplies at the harvesting areas. We refer to the set of feasible routes used in the plan as a *schedule*. In what follows, we define all the components that are essential in the LTSP. The components that are used as input to the methods proposed in the thesis are the following:

- Pickup points.
- Assortments and groups of assortments.
- Supplies.
- Customers.
- Demands.
- Time windows.
- Home bases.
- Working hours for the drivers.
- Road network and travelling distances.
- Trucks.
- Preferences or priorities.
- Transport orders: predefined destinations.

2.1.1 Pickup points, assortments, groups of assortments, and supplies

Pickup points are a geographical location where logs are piled. These locations are usually reachable for logging trucks and they are situated at the side of forestry roads. The geographical location is determined by a set of coordinates that make it possible to compute the distances between different locations.

Logs are sorted at the pickup points according to their species (spruce, pine), length, diameter and quality. This gives rise to various assortments, and today the number of assortments is increasing, since the requirements of the customers have become more specific. While logs are sorted according to their assortment, they are ordered by customers according to a *group of assortments* , which is a set of assortments that meet the requirements for a customer order.

A supply is defined as a certain assortment at a certain pickup point. Since several assortments can be available at a pickup point, several supplies can be located at the same geographical location.

2.1.2 Customers and demands

The customers are, like the pickup points, are defined by specific geographical locations. It is at these locations that the pulp and paper mills, board mills or sawmills are situated. The number of customers is very small in comparison with the number of pickup points; there are around 55 pulp, paper and board mills in Sweden while the number of sawmills is around 2000. Figure 10 shows geographical locations of larger sawmills in the country.

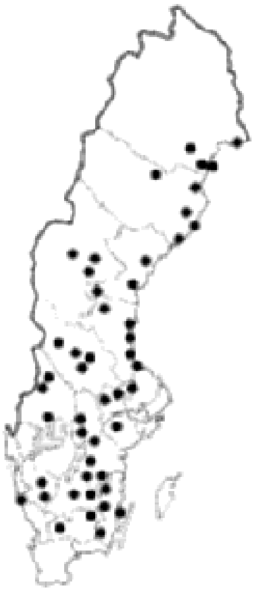


Figure 10: Larger sawmills in Sweden

Each customer might have several demands. A demand is defined by a certain assortment or a group of assortments, a specific quantity, and a time window within which the logs should be delivered.

2.1.3 Time windows and home bases

A time window is an interval, [*a*, *b*], where *a* is the time when service can begin at a certain location, while *b* is the time when the service ends. If a truck arrives before the starting hour it is allowed to wait until the opening of the time window. It is however forbidden to perform any operation after the closing hour. It is usual that the time windows at pickup points are wide, while the time windows at customers are narrower; it is possible for trucks with a crane (self loading) to pick up logs available at road sides at any time during the day, since they are not dependent on the presence of loading cranes.

The drivers of the trucks usually start and end their working day at their home base. A home base is a geographical location given by a set of coordinates. Sometimes a truck is used 24 hours a day by different drivers. In such a case, two drivers meet at a certain location to shift with each other.

2.1.4 Working hours

The drivers usually work 8 hours a day, and if the truck is run 24 hours a day then 3 drivers are needed. The number of hours a truck is run during the day differs from one company to another.

2.1.5 Road network, travelling distances, and trucks

One of the most crucial pieces of data required for planning the transportation of logs is the distances between all geographical locations. This data is not trivially available. A system developed by the Swedish Forestry Research Institute, SkogForsk, uses the Swedish National Road Data Base (NVDB) to compute these distances and provide us with the information needed.

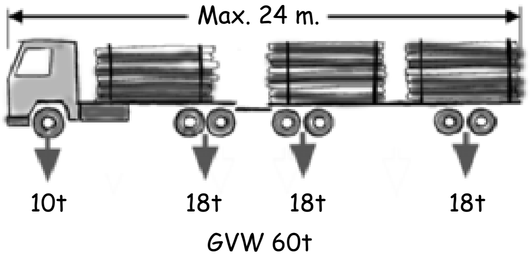


Figure 11: A logging truck

The vehicles used are logging trucks. These can differ in several ways: capacity, presence of crane, weight, size, and special equipment, like tyre type. Usually a logging truck consists of three blocks and the loading capacity is around 40 tonnes. Trucks with a self loading function are free to pick up logs at any position where logs are piled, whereas trucks that lack this function are dependent on the presence of loading cranes. In this thesis we are able to deal with a heterogeneous fleet.

2.1.6 Preferences and priorities

There are different types of preferences that influence the planning decision. These can be divided into two categories: preferences that influence what demands to fulfil first and preferences concerning the piles to be picked up.

The first category deals with the customer needs and requirements, and the importance of the customer for the forestry company selling the logs. Some customers have strict requirements for a constant flow of logs, since a shortage might lead to interruption in the production.

The second category of preferences arises in situations where it is necessary to pick up a certain pile of logs prior to others. One such a situation is when the harvesting in an area is reaching its end, and it is necessary to clear the area of all logs. Another such situation is when log piles have been stored for a long time and their quality has started to deteriorate.

All preferences are taken into consideration when formulating the problem mathematically, by means of appropriate penalties. In the following section we give two mathematical models for the LTSP.

2.2 Mathematical formulation

The log truck scheduling problem is related to other vehicle routing problems and it can be modelled in various ways. Magnanti (1981) distinguishes between different modelling possibilities for the vehicle routing problem: the set covering formulation, the vehicle flow based formulation, and the commodity flow based formulation. Different solution approaches are each especially suitable for different models (although some approaches are more general and work together with different formulations). We present two models for the LTSP. The first, which is a traditional model, contains three types of variables, while the second essentially contains only one type of variables: variables, which represent feasible routes.

2.2.1 Traditional model

We start by defining a network, upon which the model is based, the variables, the constraints, and the cost function, before we state the mathematical formulation.

Network

The traditional model is based on the network defined as below:

Let S and D be the sets of original supply and demand points, respectively.Then, the set of nodes N is defined by

H	=	set of start home nodes, one for each home base
E	=	set of end home nodes, one for each home base
S_i	=	set of copies of supply node $i \in$ S
AS	=	set of all supply nodes $= \bigcup S_i$
D_i	=	set of copies of demand node $i \in$ D
AD	=	set of all demand nodes $= \bigcup D_i$

Introducing copies of the supply and demand nodes allows us to use binary visiting variables in the model. Each copy may be visited at most once by each truck. The number of copies of each supply or demand node is dependent on the quantities available or demanded, respectively, at the nodes and on an approximation of the maximum number of times a vehicle can visit the same node during the day. The set AS and the arcs included within it are illustrated in Figure 12.

We assume that the nodes in H are numbered from 1 to v, which is the number of vehicles, so that node k corresponds to vehicle k. We also assume that the nodes in E are $n + 1$ to $n + v$ where n is the total number of nodes in AS $\bigcup$ AD$\bigcup$ H, again the end home base of vehicle k is represented by node $n + k$.

The set of directed arcs, A, consists of the following arcs:

- Arcs connecting all nodes in H to all nodes in AS.
- Arcs connecting the nodes in AS to other nodes in AS with the same assortment. Arcs connecting nodes representing the same original supply are not allowed.
- Arcs connecting the nodes in AS to all the demand nodes in AD with the same assortment.

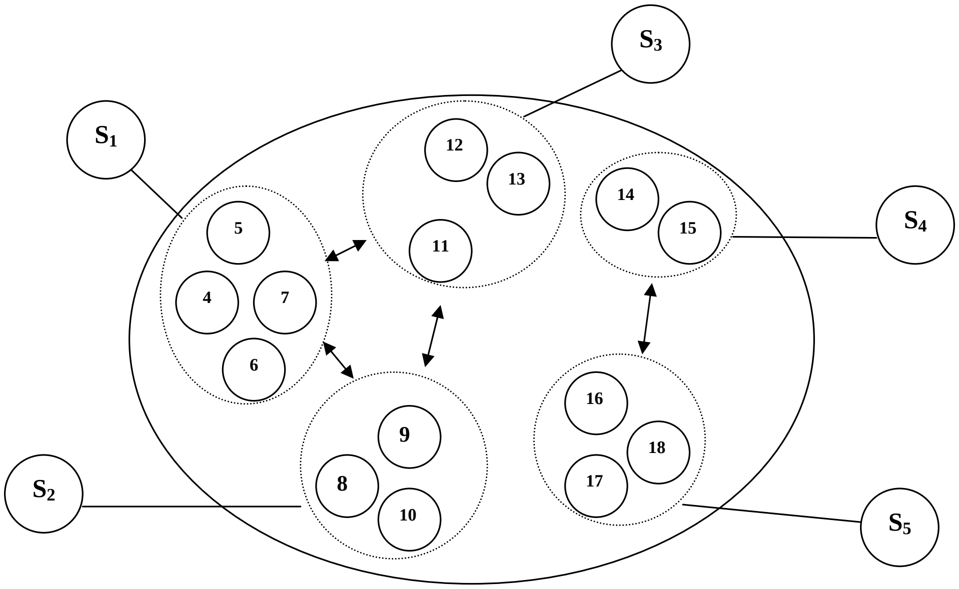


Figure 12: An example of the set AS where 5 supplies are presented.

- Arcs connecting all nodes in AD to all nodes in AS.
- Arcs connecting all nodes in AD to all nodes in E.
- Arcs connecting a node *i* in H to node *n*+ *i* in E, where *n* is the cardinality of the set of nodes AS $\cup$ AD $\cup$ H. This is done to allow each truck to be able to stay at the home base during the whole day.

Variables

As mentioned earlier three types of variables are used in the traditional model. These are *flow*, *time* and *load* variables. The model also includes variables describing quantities that are picked up at supply points; these variables are however only auxiliary and can be eliminated from the problem. The variables used are:

x_{ijk}	=	1 if vehicle *k* uses the arc (*i*,*j*), and 0 otherwise
y_{ik}	=	quantity picked up by vehicle *k* at supply node *i*, for $i \in$ AS
q_{ijk}	=	load of vehicle *k* when traversing arc (*i*,*j*)
s_{ik}	=	arrival time of vehicle *k* at node $i \in$ N.

The additional notations and constants used in the model are:

V	=	set of available vehicles
t_{ij}	=	the time it takes to travel between nodes *i* and *j*
$c_{ijk}(x_{ijk}, q_{ijk}, s_{jk})$	=	the cost for vehicle *k* to travel from node *i* to *j*
$[a_i, b_i]$	=	time interval for each node $i \in$ N
p_k	=	capacity of vehicle $k \in$ V
d_i	=	total demand required at node $i \in$ D

r_i = total supply available at node $i \in$ S.

Constraints

Some of the constraints in the problem are implicitly taken into consideration in the network. Examples of such constraints are:

- Constraints ensuring that the correct assortment is delivered to the customer.
- Constraints ensuring that the truck is loaded with only one assortment.
- Constraints ensuring that the vehicles visit at least one supply point before visiting any customer.

The non-network constraints in the problem can be summarised as follows:

- Path constraints: each vehicle path must start and end at the home base.
- Capacity constraints.
- Time windows.
- Precedence constraints ensuring that pickups are performed before a delivery.

Cost function

The cost for vehicle k to travel from node i to node j is nonlinear, and it depends on several factors. Firstly, it depends on the distance between the two nodes. Secondly, the cost depends on the load of the truck, and, thirdly, on the time when pickups, deliveries or transportations are carried out. The cost depends nonlinearly on load and time (due to overtime costs). In turn, a transport operation can either be a loaded transport or an empty transport.

Mathematical formulation

The assumptions and notations introduced above yield the following model:

Traditional model [TM]

minimise $\sum_{(i,j)\in A} \sum_{k=1}^{v} c_{ijk}(x_{ijk}, q_{ijk}, s_{jk})$

subject to

$$\sum_{j:(i,j)\in A} x_{ijk} = 1 \qquad \forall i \in \text{H},\ \forall k \in \text{V}: i = k \qquad (1.1)$$

$$\sum_{j:(i,j)\in A} x_{ijk} = 0 \qquad \forall i \in \text{H},\ \forall k \in \text{V}: i \neq k \qquad (1.2)$$

$$\sum_{l:(l,i)\in A} x_{lik} - \sum_{j:(i,j)\in A} x_{ijk} = 0 \qquad \forall i \in \text{N} \setminus \text{E},\ \forall k \in \text{V} \qquad (1.3)$$

$$\sum_{l:(l,i)\in A} x_{lik} = 1 \qquad \forall i \in \text{E},\ \forall k \in \text{V}: k = i - n \qquad (1.4)$$

$$\sum_{l:(l,i)\in A} x_{lik} = 0 \qquad \forall i \in \text{E},\ \forall k \in \text{V}: k \neq i - n \qquad (1.5)$$

$$q_{ijk} = 0 \qquad \forall k \in \text{V},\ \forall (i,j) \in \text{A}: i \in \text{H} \qquad (1.6)$$

$$q_{ijk} = 0 \qquad \forall k \in \text{V},\ \forall (i,j) \in \text{A}: j \in \text{E} \qquad (1.7)$$

$$\sum_{j:(i,j)\in A} q_{ijk} = 0 \qquad \forall\, i \in \mathrm{AD},\ \forall\, k \in \mathrm{V} \qquad (1.8)$$

$$x_{ijk}\ \Big(\sum_{l:(l,i)\in A} q_{lik} + y_{ik} \Big) = q_{ijk} \qquad \forall\, j \in \mathrm{N},\ \forall\, k \in \mathrm{V},\ \forall\, i \in \mathrm{AS} \qquad (1.9)$$

$$q_{ijk} \leq p_k\ x_{ijk} \qquad \forall\, k \in \mathrm{V},\ \forall\, (i\,,j) \in \mathrm{A} \qquad (1.10)$$

$$\sum_{(i,j)\in A: j\in D_m} \ \sum_{k\in V} q_{ijk} \geq d_m \qquad \forall\, m \in \mathrm{D} \qquad (1.11)$$

$$\sum_{i\in S_j} \sum_{k\in V} y_{ik} \leq r_j \qquad \forall\, j \in \mathrm{S} \qquad (1.12)$$

$$s_{ik} = a_k \qquad \forall\, i \in \mathrm{H}\colon i = k \qquad (1.13)$$

$$x_{ijk} \left(s_{ik} + t_{ij} \right) \leq s_{jk} \qquad \forall\, (\,i\,,j\,) \in \mathrm{A}\,,\ \forall\, k \in \mathrm{V} \qquad (1.14)$$

$$a_i \leq s_{ik} \leq b_i \qquad \forall\, i \in \mathrm{N},\ \forall\, k \in \mathrm{V} \qquad (1.15)$$

$$s_{ik}\,,\, y_{ik} \geq 0 \qquad \forall\, i \in \mathrm{N},\ \forall\, k \in \mathrm{V} \qquad (1.16)$$

$$q_{ijk} \geq 0 \qquad \forall\, (\,i\,,j\,) \in \mathrm{A},\ \forall\, k \in \mathrm{V} \qquad (1.17)$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, (\,i\,,j\,) \in \mathrm{A},\ \forall\, k \in \mathrm{V} \qquad (1.18)$$

This model formalises the problem in a traditional way, and allows us to relate it to other known optimisation problems. It is clearly strongly related to vehicle routing problems with time windows. In contrast to the model presented next, the variables and the constraints of the traditional model are defined *a priori*, by the network previously described and which is well defined. A disadvantage of the model is that the cost function is nonlinear, and not tractable as it is. Another disadvantage is that the model is rigid in the respect that small additions or changes in the problem might imply a change of the whole network or a change of the structure of the model. The rules concerning the routes can differ significantly from one company to another; this fact also implies that this traditional model is not so useful. In the following section, we present a more flexible model, where the variables represent feasible routes.

2.2.2 Model based on variables representing feasible routes

As mentioned earlier, this model involves mainly one type of variables, namely variables that represent feasible routes. But it also involves some other variables, the purpose of which we discuss before formulating the problem.

Variables representing feasible routes

A route usually starts and ends at a driver's home base and it consists of a sequence of pickups and deliveries. A feasible route is a route that obeys a number of rules or restrictions, these are:

- Precedence.
- Assortments constraints.
- Capacity.
- Time windows.
- Available supplies.
- Other more specific rules.

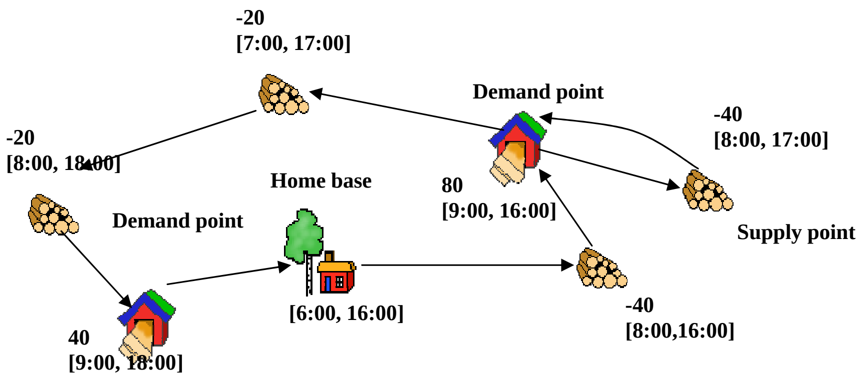


Figure 13: Example of a feasible route

The precedence constraints make sure that a pickup is carried out before a delivery. The assortment constraints ensure that correct assortments are delivered to the customers. It is of course important that the truck is not loaded over its capacity, and that all tasks are done within the limits of the given time windows. If a route is feasible then the total quantity picked up during the route at any supply point should not exceed the supply available. Finally, there is a number of rules that can be specific for certain companies, which can be added to the list above. Examples of such rules are given in Chapter 7. An example of a feasible route is given in Figure 13 where the capacity of the truck is assumed to be 40 tonnes. For each location the time window is given as well as the quantity picked up (negative) or delivered (positive). The route starts at the home base, visits a first supply point within the time window [8:00, 16:00], where 40 tonnes are picked up, and continues to the first demand point within the time window [9:00, 16:00], to deliver the 40 tonnes. The total amount delivered during the whole route to the first demand point is 80 tonnes. Then, the truck drives to the next supply point for another pick-up operation. This continues until the truck drives back to the home base, which is the end of the route.

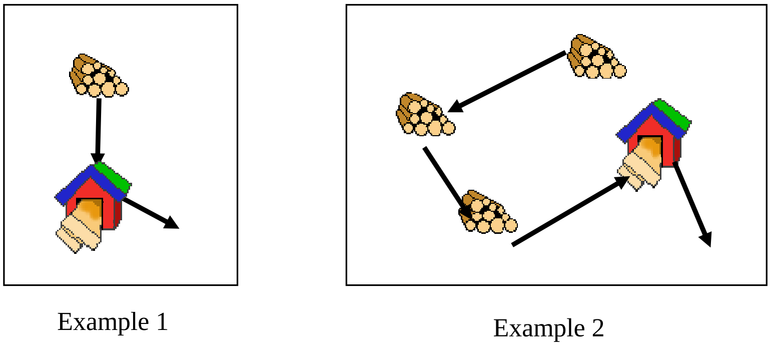


Figure 14: Examples of simple and split trips

In principle, the number of feasible routes is infinite, since it is always possible, in theory, to pick up or deliver fractions of tonnes. In order to limit the number of feasible routes we assume that the quantities that are picked up or delivered are integers. By discretising the quantities we get a finite number of variables in the model, but this finite number is still huge. In the example below, we show how the number of variables can be estimated and how it grows enormously with the number of supply and demand points.

In order to get an idea of the size of the problem, we define as a *trip* a number of pickups followed by one delivery to a customer. In principle, it is possible to deliver to several customers but since the demand is often much larger than the truck capacity the former definition of a trip is more reasonable. Example 1 in Figure 14 shows a *simple* trip where the truck visits only one supply point before making the delivery, and example 2 shows a more complex trip involving *split pickups*; three consecutive pickups followed by a delivery.

Let us consider a realistic example with 3 assortments, 210 supplies, and 15 demands. We assume that each vehicle can drive 4 trips a day (this number usually lies between 4 and 6), that 70 supplies offer each of the assortments, and that 5 customers require each of the assortments. In addition, we assume that the truck becomes fully loaded after two consecutive visits to two different supply points. In real life situations the truck gets loaded after one to, in the worst case, six or seven consecutive visits. Taking all these assumptions into consideration, the number of possible routes for *each* vehicle is approximately

$$(210 * 69 * 5)^4 \approx 2.75 * 10^{19}$$

This simple example shows that it is clearly not possible to enumerate all possible routes. The size of the problem can be dealt with by limiting the number of columns (for example by applying column generation).

The model also involves other types of variables, which we call *auxiliary variables* since they help taking the preferences into consideration.

Goal variables

There are two types of preferences: preferences concerning the importance of a demand and preferences indicating the supplies that must be picked up prior to others.

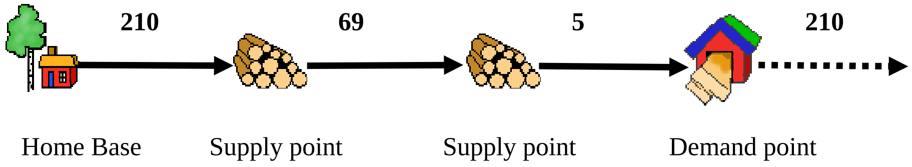


Figure 15: The number of possibilities in one trip

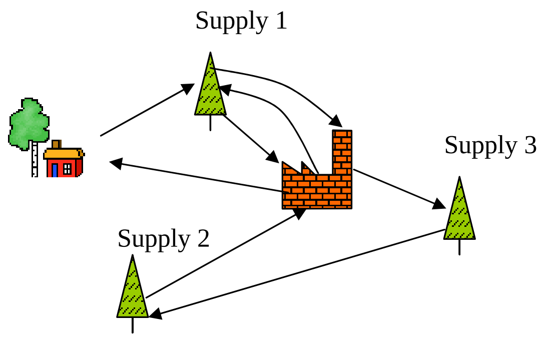
These preferences can be viewed as soft constraints in our model. We deal with these preferences by allocating priorities to each demand and supply. A priority is an integer describing how important it is to fulfil the corresponding demand or to empty the corresponding supply. The first type of preferences is handled by defining auxiliary *goal variables* that register the quantities that are not already delivered to the customers and associating a penalty to these goal variables. The penalty term that is added to the objective function is large when the demand that is not yet fulfilled has a high priority, and small otherwise. In a similar way, we introduce variables that register the quantities left at the supply points and penalise high priority supply points that are not emptied.

It is important to note that the model allows infeasibility, in the sense that demands must not be fulfilled. This is of course penalised heavily in the objective function, but it happens that the total supply is not sufficient for the demand to be met and we want our model to produce practically useful solutions anyway. We also allow infeasibility caused by delivering a quantity that exceeds the customer demand. This is allowed by using surplus variables that can be penalised in the objective function. The number of goal variables depends on the number of supplies and demands, which, unlike the number of feasible routes, is small.

Relation between a feasible route and a column in the model

We have now described the main variables in the model and defined feasible routes by giving a number of restrictions that they must satisfy. The example shown in Figure 16 illustrates the relation between a feasible route and its corresponding column in the model.

The truck in the example in Figure 16 starts by visiting supply point 1, picks up 40 tonnes and drives to the customer where it is totally emptied. The truck drives back to the same supply point for another trip. After delivering 40 more tonnes to the demand point, it visits supply point 3 and picks up 30 tonnes there. Since the truck is not fully loaded after that visit, it continues to supply point 2 where it picks up 10 tonnes. Finally, the truck delivers the last load and travels back home.



1	Truck 1
120	Demand 1
80	Supply 1
10	Supply 2
30	Supply 3

Figure 16: Example of the relation between a route and a column

The column corresponding to this route consists of zeros in all rows other than the ones shown in the figure. The information in the column includes only *the sum of the quantities picked up or delivered along the whole route*. The amount of information related to a feasible route clearly exceeds the information needed to construct the columns present in the mathematical model. This means that the feasible routes are not uniquely determined by their columns; it is typically impossible to reconstruct a real route by only considering the information in the column. It is therefore necessary to store all the data concerning the feasible routes. This includes the time of start, the order and time each node is visited and the quantities picked-up or delivered at these nodes.

Objective function

The objective function in the mathematical model involves four types of costs. Firstly, the cost of a feasible route, which is computed while creating the route. This cost depends on the distance travelled, the load of the truck and the hour when each task is performed. In practice, we use four different costs, these are:

- Cost of travelling with an empty truck and during normal working hours.
- Cost of travelling with a loaded truck during normal working hours.
- Cost of travelling with an empty truck during overtime hours.
- Cost of travelling with a loaded truck during overtime hours.

We note that in some cases we need to take some extra considerations into account for computing the cost. This happens for example when the limit for overtime is passed between two nodes, say *i* and *j*. Here we use the arrival time at node *j* to approximately decide the cost of travelling between the two nodes.

The second cost type involves the penalties that control the demand priorities. In the same way, the third type of costs controls the priority of how important it is to empty a certain supply. Finally, the fourth cost term penalises the excess at the customers.

Mathematical model

The following notations are used:

n	=	number of trucks
m_i	=	number of feasible routes for truck *i*, for $i = 1, 2, \dots, n$
n_s	=	number of supplies
n_d	=	number of demands
d_p	=	demand at point *p*, for $p = 1, 2, \dots, n_d$
s_k	=	supply at point *k*, for $k = 1, 2, \dots, n_s$
a_{ijk}	=	quantity picked up at supply point *k* by truck *i* using route *j*
b_{ijp}	=	quantity delivered at demand point *p* by truck *i* using route *j*
c_{ij}	=	cost of route *j* for truck *i*
p_k	=	penalty for not emptying supply *k*
g_p	=	penalty for not satisfying demand *p*
g'_p	=	penalty for exceeding demand *p*

The variables are:

x_{ij}	=	1 if truck *i* uses route *j*, and 0 otherwise

y_k	=	the quantity left over at supply k
u_p	=	the deficit in tonnes at demand p
u'_p	=	the excess in tonnes at demand p

The problem can be formulated as follows:

Generic model [GM]

$$\begin{array}{llllr}
\textit{min} & \sum_{i=1}^{n}\sum_{j=1}^{m_i} c_{ij}x_{ij} + \sum_{k=1}^{n_s} p_k y_k + \sum_{p=1}^{n_d} g_p u_p + \sum_{p=1}^{n_d} g'_p u'_p & & & \\
\textit{subject to} & \sum_{j=1}^{m_i} x_{ij} & = 1 & i = 1,2,\dots,n & (1) \\
 & \sum_{i=1}^{n}\sum_{j=1}^{m_i} a_{ijk}x_{ij} + y_k & = s_k & k = 1,2,\dots,n_s & (2) \\
 & \sum_{i=1}^{n}\sum_{j=1}^{m_i} b_{ijp}x_{ij} + u_p - u'_p & = d_p & p = 1,2,\dots,n_d & (3) \\
 & x_{ij} \text{ binary} & & \forall i, j & (4) \\
 & y_k, u_p, u'_p & \geq 0 & \forall k, p & (5)
\end{array}$$

A variant of this model is presented in Rönnqvist and Ryan (1995). Three sets of constraints must be met. The first set forces the model to choose exactly one route for each truck. Each truck is permitted to stay at the home base during the whole day, therefore a column with zero cost representing this alternative is added to the set of feasible routes. The second set of constraints ensures that the total quantity picked up does not exceed the supply. Finally, the third set aims at satisfying the demand.

Note that the constraints concerning the feasibility of the routes are not explicit in the model. We refer to these constraints as *local,* while the three sets of constraints expressed in the model are *global*. Local constraints are associated with each column, or variable in the model, while global constraints involve a set of variables. This structure of global and local constraints gives flexibility to the model, since all rules concerning the routes do not influence the model or the solution method. In other words, this model can be considered to be *generic* since the global constraints are country and company independent; it applies even when the rules concerning the routes differ from one company to another. Recall that local constraints are for example time windows, lunch breaks, capacity constraints, precedence constraints, and visiting constraints that make sure that certain destinations are visited. Note that the traditional model and the generic model are both valid for the LTSP, that is they are equivalent from an integer programming point of view.

It is interesting to observe that the second and third sets of constraints in the generic model are similar to the constraint in the transportation problem; satisfy the demand without exceeding the supply. Note also that it is possible to approximate the generic model by a set partitioning problem. This is done by reducing all supplies and demands into units of full loads. If a demand exceeds a full load, it can be divided into several demand points, each with a quantity that equals a full load. In contrast to that, if a supply quantity is less than the capacity of the truck, then several supply points can be aggregated into one supply point with a quantity equal to a full load. In this way, we can approximate the problem by a set partitioning problem. The similarity of this model to the set partitioning problem is used in this thesis to motivate the use of a particular branching strategy in a branch-and-price scheme.

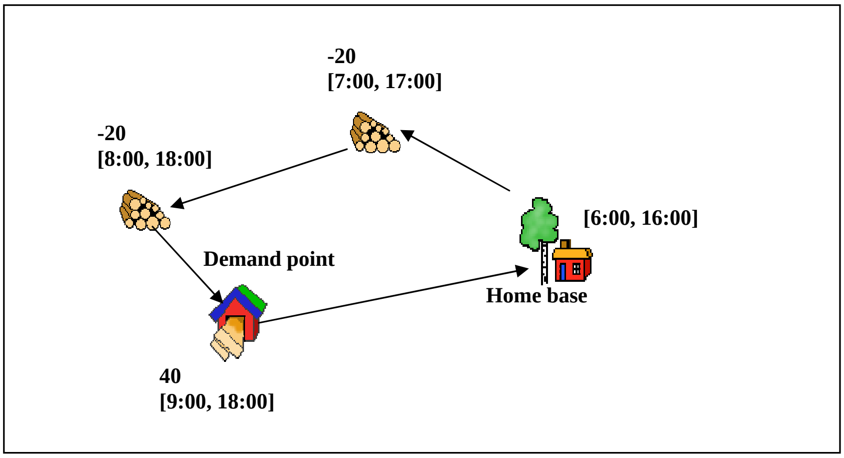


Figure 17: route with split pickups

Examples of routes

The routes in the problem can be composed of simple trips where the truck becomes fully loaded after exactly one visit at a supply point, or they can involve *splits*. There are two types of splits: split pickups and split deliveries. Split pickup refers to the events when more than one pickup is needed at a supply point, or when a truck picks up less than a full truck load at a supply point. Split deliveries occur when more than one full truck load is required to fulfil a demand. The former split type can be caused by a supply that exceeds the truck capacity or a supply that is less than the capacity. Split deliveries occur often since the demand is usually more than one load. Splits can occur either within the same route or when the same location is visited by several trucks. The examples in Figures 17 and 18 illustrate different types of splits. In the first example, two supply points are visited in order to fill the truck. In the second example, the same supply point and the same demand point are visited twice, since the quantities exceed the truck capacity, which is 40 tonnes.

In Figure 19 we also illustrate two routes that visit the same locations in the same order, but result in different columns. This is due to the fact that the truck has a choice of picking up 30 tonnes or 20 tonnes at the first supply point visited. The load of the truck after visiting the supply points is shown and the two columns corresponding to these two routes are given

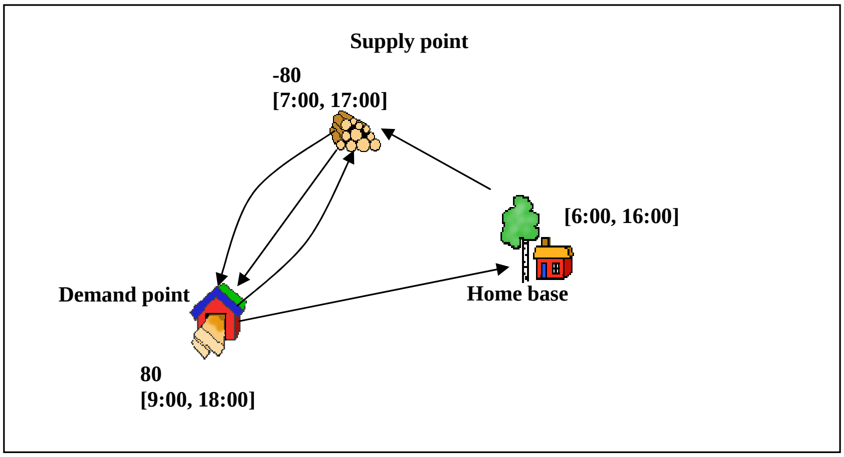


Figure 18: route with split pickups and split deliveries

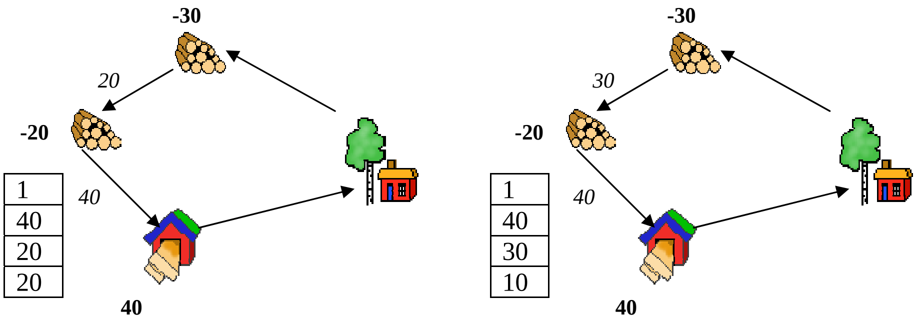


Figure 19: example of different split pickups

Figure 20 shows an example of two different routes that give rise to two columns that have the same constraint coefficients but different costs. This special case occurs when two trucks visit the same locations, but in a different order, and the quantities picked up and delivered are the same in total.

2.3 Comparison between the two models

In this section, we define three procedures, P1, P2 and P3, and we use them to compare the traditional model [TM] to the generic model [GM]. We present the relations between the LP solutions and the integer solutions that the three procedures give.

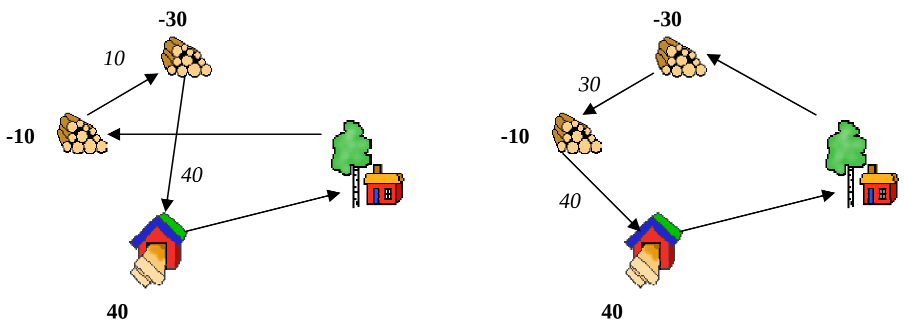


Figure 20: example of different routes that give rise to almost similar columns

2.3.1 Procedure P1: solving [TM] by Dantzig-Wolfe decomposition

In order to compare the traditional model and the generic model, we start by applying Dantzig-Wolfe (D-W) decomposition to the former. By decomposing this model we obtain two problems: a master problem involving the constraints (1.11) and (1.12), and a subproblem consisting of the remaining constraints. The master problem involves the same constraints as the global constraints in the generic model (apart from the auxiliary variables). We refer to the restricted version of the master problem as RMP1. Let V(LP1) denote the optimal objective value obtained after solving RMP1 by D-W decomposition. The subproblem, SUB1, separates with respect to vehicles and it involves the constraints of the traditional model that deal with time windows, path constraints, and load constraints; for each vehicle, this subproblem is a shortest path problem with side constraints. These side constraints can be taken into account implicitly by modelling them in a network. Each node of this network is characterised by three labels: the first represents a geographical location, the second defines a period of time within which the node can be visited, and the third label is the load of the truck *after* visiting the node. The network is designed in such a way that any path through it can neither violate the load nor the time windows constraints. In this way, the subproblem SUB1 becomes a mere shortest path problem in this network. The shortest path problem is known to have *integrality property*, i.e. the variables automatically get integer values in the solution. This means that the solution obtained by solving SUB1 is always a set of paths, one for each vehicle, starting and ending at the drivers start and end home bases. It is here important to note that the shortest paths that can be obtained by solving the subproblem *are not* equivalent to the set of feasible routes defining the variables in the generic model, [GM]. In other words, the set of variables in RMP1 differ from the set of variables in [GM]. In Figure 21 an example of a path obtained by solving SUB1 is given. We note that the quantity available at Supply 5 is 60 tonnes and the quantity picked up by the path is 80 tonnes. The example shows that the paths obtained by solving SUB1 might violate the supply constraints, simply because this set of constraints is not present in the subproblem.

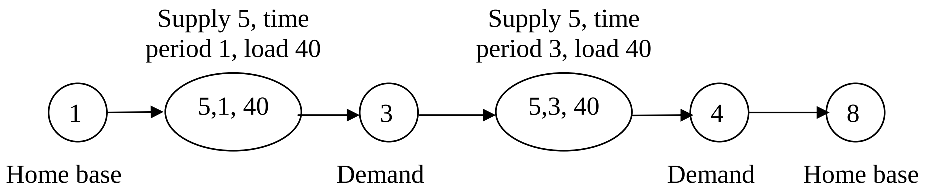

Supply 5, time period 1, load 40
Supply 5, time period 3, load 40
1
5,1, 40
3
5,3, 40
4
8
Home base
Demand
Demand
Home base

Figure 21: Example of a path obtained by solving SUB1

2.3.2 Procedure P2: solving [TM] augmented with a set of redundant constraints by Dantzig-Wolfe decomposition

Since the paths obtained by solving SUB1 might violate the supply constraints, we choose to add a set of redundant constraints to [TM] before applying D-W decomposition. This set of redundant constraints is a copy of the supply constraints, and it is placed in the subproblem. In this way, it makes sure that the quantity picked up at any supply point within a path does not exceed the quantity available. Again we consider a restricted version of a master problem, RMP2, which involves the same sets of constraints as RMP1 and [GM]. The subproblem, SUB2, differs from SUB1 since an additional set of constraints is added to SUB1. We use the (same) network (as the one) defined in the previous paragraph in order to solve SUB2. This means that SUB2 is a shortest path problem with side constraints: the supply constraints. The addition of this set of constraints destroys the integrality property of the shortest path problem, which means that it is no longer evident that we obtain shortest paths by solving SUB2. In Figure 22 we illustrate a fractional solution to SUB2. The two flow variables both have the value 0.5 and the total quantity picked up at supply 5 is exactly 60 tonnes; 40 tonnes during the first visit and 20 during the second. In this way we note that the supply constraints are fulfilled and that the solution obtained by solving SUB2 might not be a path in the network.

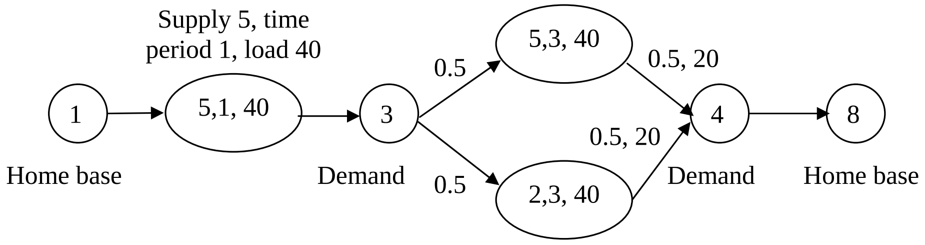


Figure 22: example of a solution obtained by solving SUB2

Let V(LP2) be the optimal objective value obtained by applying D-W decomposition and solving SUB2 as a subproblem.

2.3.3 Procedure P3: solving [GM] by column generation

Here we assume that the whole set of feasible routes is available and stored in a pool. As mentioned earlier the linear relaxation of [GM] has the same structure as the master problems obtained in P1 and P2. Since we do not add all feasible routes to the model we actually work with a restricted version of [GM]. We define the column generation subproblem, SUB3, in P3 as the problem of pricing all the routes in the pool and returning a route with (most) negative reduced cost.

Relation between P1, P2 and P3

From the definitions of P1, P2 and P3 we conclude the following:

- The restricted master problems in P1, P2 and P3 contain the same sets of constraints, but their sets of variables differ.
- The sets of feasible solutions to the subproblems SUB1, SUB2 and SUB3 differ.
- In order to explain the difference between the subproblems we use Figure 23, where the sets of feasible solutions related to three subproblems are illustrated.

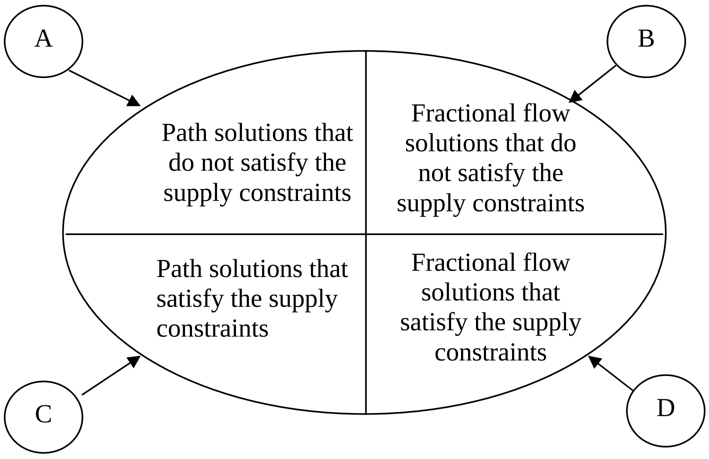

A
B
Path solutions that do not satisfy the supply constraints
Fractional flow solutions that do not satisfy the supply constraints
Path solutions that satisfy the supply constraints
Fractional flow solutions that satisfy the supply constraints
C
D

Figure 23: sets of feasible solutions related to the three subproblems

We can conclude the following:

- The set of feasible solutions to SUB1: $S1 = A \cup B \cup C \cup D$
- The set of feasible solutions to SUB2: $S2 = C \cup D$
- The set of feasible solutions to SUB3: $S3 = C$

Note that solutions to SUB1 are never (automatically) in the sets B and D, since SUB1 has the integrality property. But when we form SUB2 by adding constraints to SUB1, the integrality property is destroyed, and we can get fractional solutions from the set D, (which makes P2 of no practical interest for us, since the corresponding variables in the RMP2 can not then be associated with routes for vehicles).

Also note that $S3 \subseteq S2 \subseteq S1$, since S2 is derived from S1 by adding a set of supply constraints, and S3 is derived from S2 by adding integrality constraints. Since the set of constraints added to P2 is redundant in [TM] we conclude that
V(LP3) >= V(LP2) = V(LP1). Further, the example above shows that
V(LP3) > V(LP2) might hold.

We may conclude from this analysis that the generic model of the LTSP is in a sense stronger than the traditional model. In addition, the generic model includes exact route costs, without introducing nonlinearities, and it is also more flexible with respect to variations in the LTSP.

2.4 Earlier work on operative forestry transportation planning

As already mentioned, the everyday forestry transportation problem has been studied in a number of countries such as Chile, New Zealand and Finland. Since 1990 a computerised simulation system (ASICAM) has been in use in several forestry firms in Chile. A description of this system is given in Weintraub et al. (1996). The simulation process produces a complete working schedule for one whole day for problems involving about 200 trucks, 40 supplies and 15 demands. The process is a heuristic and it works as follows:

- In the first step, the time component is initialised to 0.
- Evaluate all possible trips for all trucks that are available at some time within an hour from the starting time t_0.
- Assign the trips that have been scheduled until time t_0 + 15 minutes.
- If any customer is still open, move the time component by 15 minutes and repeat from the second step, otherwise, end.

The evaluation of possible trips that is made in the second step is based on a set of heuristic rules. The actual costs of each trip are taken into consideration and a congestion penalty that depends, among other factors, on the number of trucks that may load at the same time at a certain supply point is added to the cost. This penalty is not a real cost, but is used only to evaluate the possible trips at a certain time of the process. After computing the cost of each trip, the trips are chosen according to a priority list.

The first priority is given to trips that include a customer with urgent demands. The second priority is given to demands that are late in their schedule, and the third is assigned to supply points with a high stock level. In the third step, the algorithm starts by choosing the trips with the lowest costs that are included in the first priority category. If no such trips exist, the process chooses between the trips that satisfy the second priority requirement, and then the third, and finally it chooses between the remaining trips. Note that in each iteration, the state of each demand and supply is updated and the priorities are recomputed.

The use of ASICAM has led to a number of improvements, among these:

- Less queuing
- Transportation costs reductions

- Reduction of the number of trucks and loaders
- More regular deliveries to customers

The ASICAM system has been the subject of interest in other countries where forestry is important. Cossens (1993) checked the suitability of this system for conditions in New Zealand and concluded that it could be used after a few modifications. However, some New Zealand companies preferred a real-time despatching system instead of having the entire daily plan made in advance. Real-time despatching plans are less sensitive to queues, break downs or any other sudden changes that might occur during the day. This shows that the needs of the forestry companies can differ between countries and companies.

Rönnqvist and Ryan (1995) describe a real-time despatching approach at one of the major forestry companies in New Zealand. Here, the entire daily schedule is not required, and instead, the objective is to generate one trip at a time for each truck that is available. The truck drivers call the transport centre once their deliveries have been completed and ask for a new trips. The solution method must therefore be able to find solutions quickly.

The mathematical model used is similar to the generic model used in the thesis. It contains mainly one type of variables (slack and surplus variables are also included) and is a set partitioning type formulation. The columns represent trip sequences, or routes, that could be use by the trucks and the objective is to assign one trip sequence to each truck in order to meet all constraints. Since this approach aims at solving the problem in real time, the method starts by choosing the "best" single trips for the trucks in a greedy fashion. The trips found at this stage are stored in a look-up table that can be accessed by the planner. During a second stage, the method designs a trip sequence that includes the next trip and all the trips needed to cover the remainder of the planning period. The first trip in each trip sequence is added to the look-up table.

The trip sequences created in the second stage are used in an LP relaxation of the mathematical model, which is then solved by column generation. The column generator is a heuristic that produces "good" columns quickly. The branch-and-bound procedure applied concentrates on finding integer solutions for the trucks that are expected to be calling in within a short time. Again, the trips constructed in this optimisation phase are used to update the look-up table. After this phase, the whole process is repeated in order to assign new trips to the trucks available at the moment. One of the advantages of this system is that queues can be taken into account. The first two stages of the algorithm ensure the presence of good trips quickly in case any truck calls in, and if time allows, the optimisation phase aims at producing better trips.

Despatching is also used by Sahlin (1998) and Carlsson et al. (1998). Their solution process consists of three phases. In the first phase, a heuristic simply enumerates all trips of a certain character. During this phase, all supplies and demands that are available at the time when the procedure is run are considered. During the second stage the costs of the trips are calculated, and finally, in the third phase, a trip is assigned to each available truck, if possible. This three-phase procedure is repeated until the scheduling period is over.
During the first phase of the algorithm, six types of trips are generated, where one is the wait option. The other trips consist of one or two pickups followed by a delivery

to a customer or a drive back to the home base. Each time a trip is finalised, the truck driver calls the despatcher and gets a new trip according to the trip assignment. If no feasible trips were assigned to the truck, it must wait until a feasible trip is generated. The cost of a trip depends of the current position of the truck, and also other factors such as the load of the truck and the priority of the picked up product. This despatching method was tested on problems based on data from a Swedish forestry company, Sydved, which is the same data described in Chapter 4. The tests made show that solving the trip assignment problem optimally produces better solutions than by using a heuristic, and that the number of trucks used was reduced.

The combination of both heuristic and exact methods has also been applied successfully in Finland. A system called EPO has been in use at a large Finnish company, Enso-Gutzeit, since 1993. In 1995, when Linnainmaa, Savola and Jokinen presented their paper in Montreal describing this system, Enso was in charge of 250 trucks that carried out the transportation to the factories and sawmills. Among many challenges that the Finnish forestry sector was facing, the authors mention that the government forbade by law the storage of logs in the forests for more than two weeks during summer time. This law, together with a number of other requirements, made it even more urgent to organise the transportation by developing a system that could handle the transportation planning.

EPO is a system that deals with all the planning stages from strategic to operative. The input data is collected on-line directly from the forests and the main output is a weekly schedule for each truck. The solution approach combines both heuristics and optimisation, and consists of three phases:

1. Determine which supply is reserved for each factory
2. Produce possible schedules for the trucks
3. A postprocessing phase that is done in part manually.

The main objective is to minimise the total distance driven, even if the minimisation of empty driving is very important too. The system has been in use at despatch centres that are in charge of 20 trucks each. The annual savings were estimated to several millions US Dollars. The company also planned to develop the system further so that it could handle train and ship transportation as well.

A special case of the log truck scheduling problem is considered in Jorgensen (1999), where the supply point for each demand is defined beforehand. This means that the number of possible trips from supply points to demand points is much more limited compared to the case where the destinations are not specified in advance. This kind of application arises in a Swedish transport company, Svenska Skogsåkarna, which is located in the northern part of Sweden. This company is in charge of a large number of vehicles of different capacities, and they assign trips to vehicles in order to meet all the requirements of their customers. In this case, customers are either forestry companies or industries, and a trip-order is defined as a certain quantity of logs that has to be transported from one specific location to another within a specific time horizon.

Svenska Skogsåkarna have been testing an optimisation-based computer system that has been developed in Denmark. The solution method used in this system is described in Jorgensen (1999) and consists of solving an assignment problem where the remaining quantity in each trip-order is assigned to trucks. The cost matrix for the assignment problem is computed in a way that takes into consideration several constraints (like time windows).

The algorithm suggested works as follows: in the first step the different orders are checked to see if the quantity in any order is less than the capacity of the smallest available truck. If this is the case, there is a possibility to fill the truck from a different trip-order with the same pickup point, if such a trip-order exists. In the second step, all assignment costs are computed. A heuristic called “best-next best” is used in the third stage to solve the assignment problem. In each solution of the assignment problem orders are assigned to trucks. After that, the assignment costs are recalculated for each truck that was given an order in the previous iteration. The remaining quantities in each order are also updated, and the whole process is repeated until all orders are transported. The heuristic used to solve the assignment problem results in a short running time, which allows interactive use of the system.

3 Routing problems related to the log truck scheduling problem

Vehicle routing problems constitute an important and well-studied class of optimisation problems within Operations Research, since a huge number of applications are related to vehicle routing. Examples are school bus routing, airline and railway fleet scheduling, fuel oil delivery, urban transit, and many others.

In this chapter, we describe various routing problems and discuss the relations between them. In particular, we show that the log truck scheduling problem is a very general routing problem. For this purpose, we start by describing the most basic routing problems and proceed by adding and removing constraints until the log truck scheduling problem is derived.

3.1 The travelling salesman problem

The most basic routing problem is the Travelling Salesman Problem (TSP), a problem that has been studied for over 50 years. The travelling salesman problem is still an important subject for research, and the methods used for solving it can be applied (indirectly) to vehicle routing problems. The TSP corresponds to routing a *single* vehicle (or salesman) through a number of points, and the route must end at the same point as where it started. Each city should be visited *exactly once* and the objective is to decide the order in which all the cities are visited, that minimises the total travel cost (or distance).

Although the TSP is easy to state, it is an NP-hard problem (see, e.g., the proof in Papadimitriou, 1977), and no polynomial algorithms for solving it can be found (unless P = NP). The TSP has been studied extensively, and today, many methods based on heuristics and optimisation theory are used successfully to solve quite large instances; these methods include genetic algorithms, simulated annealing, cutting planes, branch-and-bound, dynamic programming, et cetera. The largest TSP instance that has been solved to optimality is the route (or tour) among 24,978 towns and villages in Sweden. This instance has been attacked by several research groups using various methods, since 2001. The method that has succeeded with the latest improvements on the Sweden instance in March 2003 is the Lin-Kernighan heuristic described in Helsgaun (2000).

3.2 The vehicle routing problem

In the basic Vehicle Routing Problem (VRP), (Bodin et al. 1983), a number of vehicles with a given capacity, q, must be routed to service n demand points, or customers, and all routes must start and end at a depot. Customer i has a specific delivery requirement, d_i, that must be met, and the cost of travelling between customers i and j is c_{ij}.

In other words, the VRP consists of finding a set of minimal cost routes, one for each vehicle (empty routes where the vehicles stay at the depot are also allowed), in order to satisfy the demands of the customers. The constraints on the routes are the following:

- Each route must start and end at the depot.
- Each customer must be visited *exactly once*.
- The total demand from the customers that are served by each vehicle must not exceed the capacity of the vehicle.

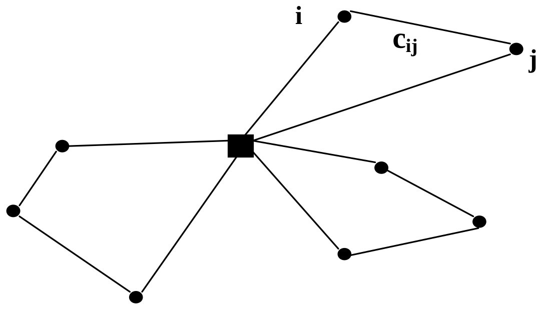


Figure 24: A VRP solution with 3 vehicles

There are of course a large number of variations on this basic VRP. One example (of this) is to minimise the number of vehicles used instead of minimising the total routing cost. The basic VRP generalises the TSP; the TSP is obtained by considering a VRP with a single vehicle of unlimited capacity. Thus, the TSP is a special case of the VRP, which is therefore NP-hard.

The VRP can be formulated in various ways, for example as a so-called *set partitioning problem*. This model for the VRP is

$$\begin{array}{llllr} \textit{min} & \sum_{j \in P} c_j x_j & & & (3.1.1) \\ \textit{subject to} & \sum_{j \in P} a_{ij} x_j & = 1 & \forall i \in C & (3.1.2) \\ & \sum_{j \in P} x_j & \le n & & (3.1.3) \\ & x_j \in \{0,1\} & & \forall j \in P & (3.1.4) \end{array}$$

Here each variable, x_j, corresponds to a given route *j* for a vehicle, and it takes the value 1 if route *j* is chosen, and 0 otherwise. Further, *n* is the number of vehicles, *C* is the set of customers, *P* the set of all feasible routes, c_j the cost of route *j*, and, finally, a_{ij} is a constant with value 1 if route *j* visits customer *i* and 0 otherwise. For each route $j \in P$, these constants are such that $\sum_{i \in C} a_{ij} d_i \le q$. Constraint (3.1.2) forces all customers to be visited exactly once, and constraints (3.1.3) ensure that at most *n* vehicles are used.

An important variation of the VRP is obtained by specifying the time interval within which the service at each customer can be completed; this specific problem is known as the vehicle routing problem with time windows.

3.3 The vehicle routing problem with time windows

In the Vehicle Routing Problem with Time Window VRPTW (see e.g. Solomon, 1987 and Solomon and Desrosiers, 1988) a time component is included in the VRP, and the problem is then referred to as a scheduling problem. As before, each customer must be visited exactly once, but now the time of arrival to and departure from each customer should also be registered, and these times should lie within a time interval, or time window, associated with the customer. The time window of a customer *i* is represented by an interval $[a_i, b_i]$ where a_i is the earliest time when the service can start at node *i* and b_i is the latest. In real life situations, routing problems where the time aspect is important are common, while pure routing problems, where the geographical component is the dominating factor, are less common. It is important to note that the VRPTW is a generalisation of the VRP; if the time constraints are not binding (this happens if the time intervals are wide enough), the VRPTW reduces into a VRP.

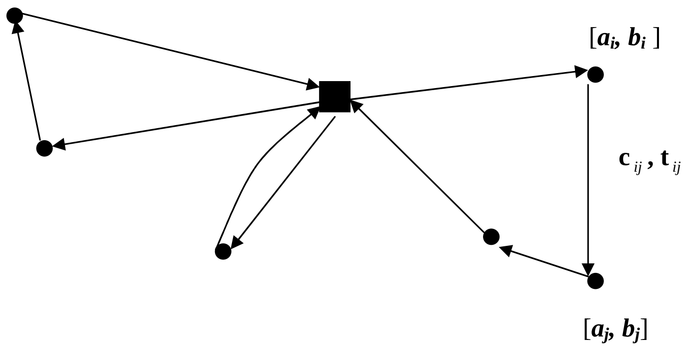


Figure 25: An example of a VRPTW solution

The VRPTW is easily stated. The problem consists of designing a set of routes, one for each vehicle, in such a way that the total travel cost is minimised and where each route satisfies the following constraints:

- The route must start and end at the depot.
- Each customer must be serviced exactly once.
- The total demand at the customers serviced by each vehicle should not exceed the vehicle capacity.
- The time window of each customer must be respected.

The number of vehicles in the VRPTW can be fixed or unlimited. In the first case, a constraint that limits the number of vehicles used is needed. Figure 25 shows an example of a VRPTW where the solution consists of three routes. Two parameters are associated with each arc; c_{ij} is the cost of travelling between node *i* and *j*, and t_{ij} is the time it takes to travel this distance.

There are different ways of modelling the VRPTW. A model where three types of variables are used is described in Desrosiers et al. (1995). These are flow variables, time variables, and load variables that are used to express the total load of a vehicle after visiting a certain customer. The formulation used for example in Kohl (1995) contains two types of variables. A flow variable x_{ijk} is 1 if vehicle *k* travels directly from node *i* to node *j*, and 0 otherwise. Further s_{ik} is a time variable that denotes the time when vehicle *k* starts its service at customer *i*. The problem is formulated as follows.

Minimise $$\sum_{k\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk} \qquad (3.2.1)$$

subject to

$$\sum_{k\in V}\sum_{j\in N} x_{ijk} = 1 \qquad \forall i \in C \qquad (3.2.2)$$

$$\sum_{i\in C} d_i \sum_{j\in N} x_{ijk} \le q \qquad \forall k \in V \qquad (3.2.3)$$

$$\sum_{j\in N} x_{0jk} = 1 \qquad \forall k \in V \qquad (3.2.4)$$

$$\sum_{i\in N} x_{ihk} - \sum_{j\in N} x_{hjk} = 0 \qquad \forall h \in C, \forall k \in V \qquad (3.2.5)$$

$$\sum_{i\in N} x_{i,n+1,k} = 1 \qquad \forall k \in V \qquad (3.2.6)$$

$$s_{ik} + t_{ij} - K(1 - x_{ijk}) \le s_{jk} \qquad \forall i \in N, \forall j \in N, \forall k \in V \qquad (3.2.7)$$

$$a_i \le s_{ik} \le b_i \qquad \forall i \in N, \forall k \in V \qquad (3.2.8)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in N, \forall j \in N, \forall k \in V \qquad (3.2.9)$$

Here the customers 0 and n+1 both represent the depot, d_i is the demand of customer i, c_{ij} is the cost of travelling from customer *i* to *j*, and, finally, t_{ij} is the time duration of that trip. Further, *V* is the set of all vehicles and *N* is the set of all nodes. The first constraint forces each customer in *C* to be visited exactly once and by exactly one vehicle. Constraint (3.2.3) forbids the vehicle to serve more customers than its capacity *q* allows. The flow conservation constraints are expressed in (3.2.4) to (3.2.6). Constraint (3.2.7) handles the travel time requirements between any pair of customers and constraint (3.2.8) ensures that the time windows are respected. Here *K* is a sufficiently large constant. (In this formulation, all vehicles are used, although some of them can be used on empty routes.)

Applying Dantzig-Wolfe decomposition to the VRPTW, with the constraints (3.2.2) and (3.2.4) leads to a set partitioning master problem. It contains one type of variables, namely structured route flow variables. The master problem obtained from the decomposition of Kohl's model is similar to the set partitioning model of the VRP, see model (3.1.1) to (3.1.4). The difference lies in the D-W subproblem, since a feasible route in the VRPTW has to satisfy a larger number of constraints than a feasible route in the VRP. If the entire set of feasible routes is considered, the complete master problem becomes an alternative way of formulating the full problem.

The VRPTW is of course far from unique in being expressible by different mathematical formulations. In fact, all vehicle routing problems have this in common.

Unfortunately, the usefulness of the model given above (and its relatives for similar routing problems) is limited, and it cannot be used directly to solve the VRPTW. The reason is that the size of this model grows rapidly, which makes it impractical. However, this model and its relatives are still useful since they can be decomposed into smaller problems with structures that can be exploited.

Both the mathematical model (3.2.1) to (3.2.9) and the one which is based on set partitioning can handle a number of variations of the VRPTW, for example they can be used for a heterogeneous fleet, deal with either pick-ups or deliveries, and multiple depots as well as just one. The VRPTW can also be extended to a more general problem involving both pick-up and delivery operations. The problem then obtained is known as the pick-up and delivery problem with time windows.

3.4 Pickup and delivery problem with time windows

The Pickup and Delivery Problem with Time Windows (PDPTW), see, e.g, Savelsbergh and Sol (1995), is similar to the VRPTW, except that now both pickups and deliveries are included. In PDPTW applications, each customer has a *request* that has to be satisfied. This request consists of a pickup operation at an origin node and a delivery at a destination node. The pickup and delivery operations have to be performed on time, and therefore a time window is associated with each pickup and delivery point. The pickup operation must of course precede the delivery. The PDPTW arises in a number of applications, like the dial-a-ride problem or in the transportation of disabled persons.

The additional issues lead to new constraints that must be added to the VRPTW constraints. Thus, besides the constraints of the VRPTW discussed earlier, two constraints that are directly connected to the pickup and delivery operations have to be met in the PDPTW. Again, the problem consists of finding optimal routes in such a way that a number of requirements are satisfied, namely:

- Each operation point (pickup or delivery) must be visited exactly once
- Each route starts and ends at the depot
- Time window constraints
- Capacity constraints
- Constraint on the number of vehicles available
- Precedence constraints
- Pairing constraints

The VRPTW and the PDPTW have the first five constraints in common. The precedence constraints express that each pickup must be made before the corresponding delivery, and, finally, the pairing constraints make sure that once a pickup point is visited, its corresponding delivery point has to be visited by the same vehicle.

Savelsbergh and Sol, distinguish between the single vehicle PDPTW and the multiple vehicle cases. The former is a constrained TSPTW, while the latter is a constrained VRPTW. Here again, a number of variations on the problem is of course possible, depending on whether it is people or goods that are to be transported. Both the single and multiple vehicles PDPTW can be formulated mathematically using vehicle flow variables, time variables, and load variables.

In Dumas, Desrosiers and Soumis (1991), the multiple vehicle PDPTW is formulated as a set covering problem, where each column represents an admissible route, that is, a route fulfilling all constraints in the problem. The set covering model is obtained by replacing the equality constraint (3.1.2) in the model (3.1.1) to (3.1.4) by the constraint

$$\sum_{j \in P} a_{ij} x_j \qquad \geq 1 \qquad \forall i \in C$$

which enforces each customer being included in at least one of the routes that are used.

It is interesting to note that the idea to use a set covering type model and column generation for a pickup and delivery problem that arises in ship routing, was studied already in the late sixties by Appelgren (1969 and 1971). The column generation subproblem can be formulated as a longest path problem in an acyclic network and is solved by dynamic programming. He also combines column generation with branch-and-bound and cutting plane methods, in order to solve the integer problem.

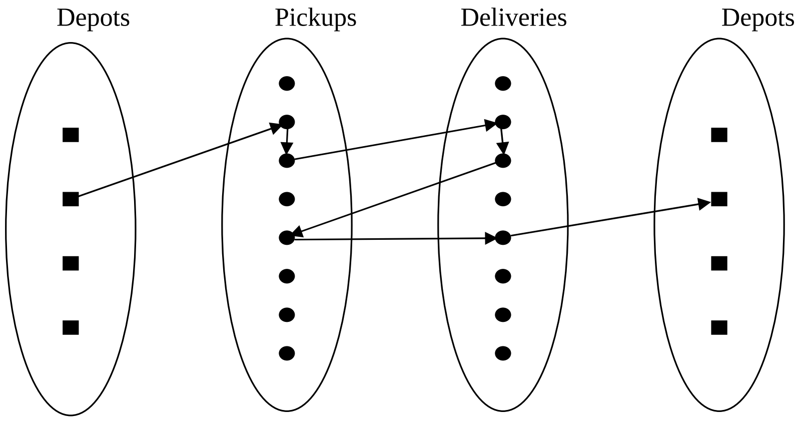


Figure 26: A one-vehicle route in the PDPTW

Figure 26 shows a network presentation of a one-vehicle route starting and ending at the same depot. Here we differentiate between three types of nodes: the depots, the pickup nodes, and the delivery nodes. The last set of depots is identical to the first one, thus each depot is represented by two nodes in the network, the first node being used as a start node and the second as the end node. In this example the vehicle is free to combine pickups and deliveries as long as all constraints are satisfied.

3.5 Routing problems including split pickups or deliveries

The routing problems that are presented above have one common feature: the pickup and delivery locations are visited *exactly once*. Many practical routing problems however involve several visits to the customers. Problems where locations must be visited several times are referred to as Split Pickups and (or) Split Deliveries Problems. The basic vehicle routing problem with split delivery was introduced by Dror and Trudeau in 1989. Split pickups or deliveries are of course of interest only when the transported goods can be physically divided.

Normally, there are two reasons for splitting the pickups (deliveries); firstly, in the case where some supplies (demands) of goods are larger than the vehicle capacity, and secondly, when some supplies (deliveries) are less than the vehicle capacity. The latter case may not seem natural, but it can be more cost efficient to allow split pickups (deliveries). Dror and Trudeau discussed this issue in the case of split deliveries, and we illustrate their result using an example (Figure 27). Suppose there are three vehicles with capacity 30 volume units, and suppose there are 3 customers with demands 25, 10, 25 respectively. Suppose also that the distances in the network satisfy the triangle inequality. Then it is easy to note that it is more cost efficient to use two vehicles and split the deliveries instead of not splitting and using three vehicles.

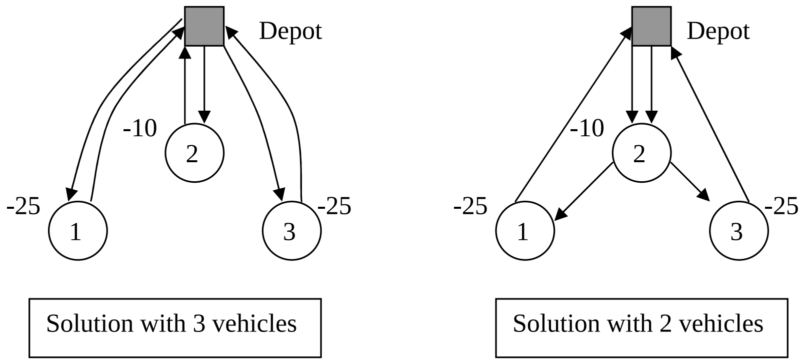


Figure 27: Example of efficient split deliveries

Ho and Haugland (2004) considered the vehicle routing problem with time windows and split deliveries. The split pickup and delivery problem without time windows has been studied, for example, in De Meulemeester et al. (1997) or Dror et al. (1998). In all the routing problems discussed above (VRP, VRPTW, PDPTW), customer demand is satisfied automatically by ensuring that each customer is visited exactly once. In the case where split deliveries are allowed, a customer might be visited more than once, and therefore constraints ensuring that the customer demands are satisfied replace the requirement of exactly one visit. The constraints in the split delivery problem with time windows (see e.g. Ho and Haugland, 2002) are the following:

- Each route must start and end at the depot
- Capacity constraints
- Time windows
- The customers’ demands must be fulfilled
- Each customer must be visited by at least one vehicle (in fact redundant).

3.6 The log truck scheduling problem

The log truck scheduling problem is closely related to vehicle routing and scheduling problems. It has many similarities to the pickup and delivery problem with time windows in particular. The major differences between the log truck scheduling problem and the PDPTW are the large number of pairing (pickups and deliveries) constraints combinations and the possibility of visiting the same location several times during the same day, by the same or by several trucks. A truck could actually even drive back and forth between a certain pickup point and a certain delivery point along the whole route before returning to the depot.

The log truck scheduling problem arises in the daily operative planning task. The objective is to find a set of feasible routes, one for each vehicle, in order to meet all customer demands. Most of the log truck scheduling constraints are common to the PDPTW and VRPTW. Nevertheless, there is one major difference between these vehicle routing problems and the log truck scheduling problem, namely that both split pickups and split deliveries are not only allowed, but are even necessary. In all vehicle routing problems that do not allow splits, a task (pickup or delivery) is totally accomplished once the corresponding location is visited. This is not the case in the log truck scheduling problem, since a customer could be visited several times and its demand might still not be satisfied. More specifically, the constraints in PDPTW and VRPTW that ensure that each pickup and delivery task is performed exactly once, is replaced in the log truck scheduling problem by explicit constraints on the total load quantities that are picked up or delivered at supply or demand points, respectively. These constraints are similar to the supply and demand constraints of the usual transportation problem.

This means that the main decision in the LTSP is to determine both what supply and demand points each truck should visit, and how many tonnes it should pickup and deliver at these locations. In particular, the quantities picked up at each supply point are not known in advance. In fact, these quantities are dependent on the capacity of the vehicle, the load of the truck when it arrives at the supply point, and the quantity left at the supply point. The quantities delivered at a demand point are handled in a different way; it is up to the planner to decide whether (or not) only fully loaded trucks are allowed to deliver at the demand points. Moreover, the planner decides the fraction of the load that must be dropped at each demand point.

The objective of the log truck scheduling problem is to develop a set of minimal costs routes such that

- Total deliveries to demand points (customers) are as requested.
- The supplies at the pickup points (harvesting locations) are not exceeded.
- Each vehicle starts and ends at its depot (or home base).
- The time windows at each location are respected.
- The capacity of the vehicle is not exceeded.
- A delivery is preceded by one or several pickups.
- The correct assortments are delivered to the customers.

The first two constraints replace the requirement in the PDPTW that each location is visited exactly once. The last constraint is a more general form of the pairing constraint. In the PDPTW, each pickup is followed by a delivery at a point that is specific for the pickup that was first made. In the log truck scheduling problem, different assortments of logs can usually be picked up at several supply points, and can, in principle, be delivered to any customer requesting those assortments.

The possibility of split pickups and split deliveries makes the LTSP more complex than the PDPTW. Also other practical factors such as a heterogeneous fleet, multiple depots (home bases), wide time windows, and several breaks during the day, add to the complexity of the problem. If we consider the special case of the log truck scheduling problem where each demand and each supply is less than the vehicle capacity, and all quantities can be picked up only if the available vehicle capacity allows it, or, in other words, only one truck is allowed to visit each point, and thus the log truck scheduling problem turns into a multiple vehicle PDPTW. Thus, the log truck scheduling problem is more general than the multiple vehicle PDPTW, which is in its turn a generalisation of the vehicle routing problem with time windows and consequently, a generalisation of the classical vehicle routing problem. These observations prove that the log truck scheduling problem is NP-hard.

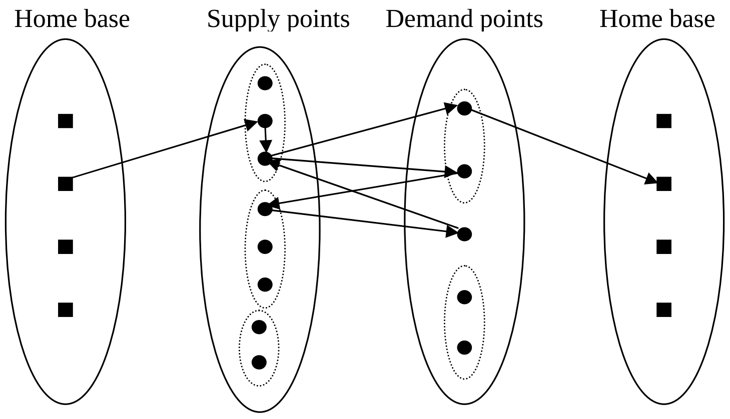


Figure 28: An example of a route in the log truck scheduling problem

Figure 28 shows a route travelled by one truck during a day. The set of supply points is composed by a number of subsets, each of which consists of the supply points that offer the same assortment. The customers are also grouped according to which assortment they require. A vehicle is allowed to pick up at several supply points included in the same subset before a delivery, and the delivery has to match with the assortment of the subset.

There are of course several variations of the log truck scheduling problem; the following requirements, and their opposites, define important cases of the problem.

- Vehicles must be fully loaded before any delivery.
- Vehicles should not mix different assortments in one load.
- Vehicles must always pick up the maximum quantity that is possible.
- Vehicles must be emptied at a customer location point.

- Vehicles must be empty at the start and at the end of the day.

If vehicles are allowed to mix several assortments in one load, or to visit several customers in a row, the problem becomes even more complex, since the number of possible routes then increases considerably. In real life situations however, it is most practical and realistic to unload completely at a single demand point and to pickup as much as possible at each supply point. In the thesis, we consider all the requirements that are listed above.

We finally remark that a problem similar to the log truck scheduling problem arises in ship routing, see Christiansen and Nygreen (1998). Their problem includes an inventory issue, which is not the case for the log truck scheduling problem.

4 Column generation by a cluster-first-route-second heuristic

4.1 Introduction

As already described, the log truck scheduling problem is one of the most complex routing problems since both split pickups and deliveries are included. It consists in finding one feasible route for each vehicle in order to satisfy the demands of the customers and in such a way that the total transport cost is minimised. In this chapter we present a method to solve the LTSP based on the generation of a pool of routes, followed by composite pricing of columns and a branch-and-bound method.

The generic model presented in Chapter 2 acts as the master problem in the method. We generate a priori a set of feasible routes that define potential columns or variables in the restricted master problem. This generation is based on the optimal flow solution to a transportation problem. This flow solution to the transportation problem helps to build clusters, which are then used to create the routes; this route generation scheme is thus a type of cluster-first-route-second heuristic. All feasible routes generated are stored in a pool. This set of feasible routes is then used to solve the LP relaxed master problem by applying a composite pricing algorithm, which mainly consists of pricing the pool of columns and maintaining an active set of these. To obtain an integer solution to the LTSP, we employ a branch-and-price approach in which we apply composite pricing to allow new columns to enter the restricted master. The branching strategy used is constraint branching. We apply this method to two case studies from Swedish forestry companies.

4.2 General framework

The columns needed in the generic model [GM] can be obtained in different ways, some of which are illustrated in Figure 29. One way is to enumerate the whole set of feasible routes, that is, columns, and construct the complete generic problem by adding all the corresponding columns. In most cases, this approach is not practical due to the large number of feasible routes.

Another approach is to generate a subset of all feasible routes. If the number of routes in this subset, or pool, is very large, a pricing procedure can be applied in order to select the most profitable columns and add them to a restricted problem. This restricted problem includes a limited number of active columns, and the pricing procedure is only used to verify optimality or to generate new active columns. This approach does not guarantee a global optimal solution to the original problem. However, it does guarantee an optimal solution to the problem considered i.e. within the generated pool of columns. Solving a subproblem according to the Dantzig-Wolfe scheme is another way of generating columns, see e.g. Lasdon (1970).

We have chosen to generate a large number of columns with a heuristic algorithm and store them in a pool of columns. By pricing all columns in this pool, new columns with negative reduced costs can be chosen and added to the restricted master problem. This approach is referred to as *composite pricing* since we do not actually use a proper column generator to produce the columns.

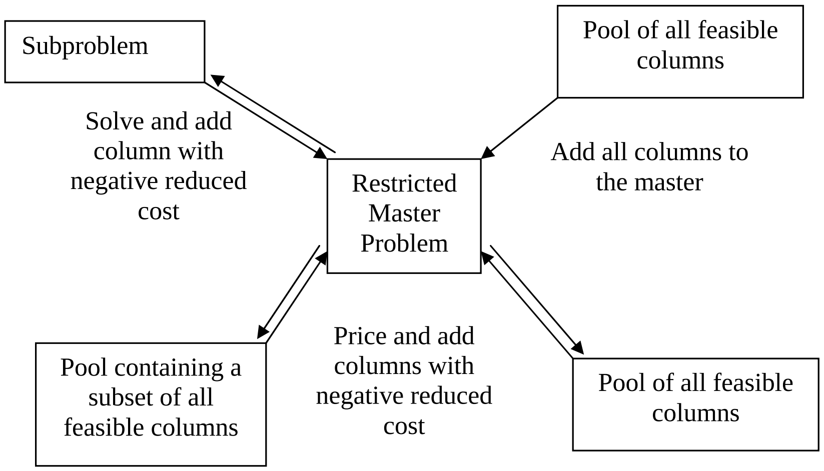

Subproblem
Pool of all feasible columns
Solve and add column with negative reduced cost
Restricted Master Problem
Add all columns to the master
Pool containing a subset of all feasible columns
Price and add columns with negative reduced cost
Pool of all feasible columns

Figure 29: Different sources of columns.

Composite pricing involves two types of pricing iterations. On a major iteration, all columns in the pool are priced and, based on the value of the reduced costs, we select a subset of variables (usually, but not necessarily, just those with negative reduced costs) which are added to the previously found subset of active variables. On a subsequent sequence of minor iterations, only the variables in the active subset are priced to find the entering variable in the standard simplex method. When no entering variables exist in the active subset, another major iteration is performed to create an expanded active subset. When no new variables, with negative reduced cost, can be found in a major iteration, we have determined an optimal solution to the LP relaxed generic problem which includes the whole a priori generated pool of columns. If we replace the composite pricing of all columns in the pool by a proper subproblem defined through an optimisation problem, then we obtain a standard column generation scheme.

The purpose of this chapter is to propose a fast heuristic method for the LTSP based on optimisation methods. In the next chapter we create a proper column generator by solving a subproblem which is a large constrained shortest path problem, that is NP-hard and more time consuming.

By considering the subset of all feasible columns that are stored in the pool, we work with a restricted version of the complete model [GM]. Our aim is to solve the problem within this subset of feasible columns. For this reason we apply a method that consists of two phases

1. The first phase consists of solving the linear relaxation of the restricted problem.
2. In the second phase an integer solution is obtained by first applying branch-and-price and afterwards branch-and-bound.

Each of these phases is described in detail in the following sections.

4.3 Solving the linear relaxation

The first step in this phase is to relax the integrality constraints (4) in the model [GM]. The idea is then to apply composite pricing to a pool of routes, that is, we use the dual prices obtained from solving the linear relaxation of the restricted master problem to compute the reduced costs of all routes in the pool. In order to initialise the dual prices we solve the restricted master with a set of initial columns; this initial problem is denoted LRRMP0. The initial columns represent *empty routes*, the routes where each truck stays at home (with zero cost) during the whole day, together with slack and surplus variables. The initial optimal LP values is the sum of all penalties resulting from not satisfying any demand. Once the dual prices are obtained we use them in a composite pricing approach to solve the LRRMP within the set of routes in the pool, which are generated according to the cluster-first-route-second heuristic described below.

4.4 Designing feasible routes

Our strategy is to enumerate a subset of the feasible routes and to limit the number of routes in this subset as much as possible without omitting the routes with high quality. Instead of enumerating all possible routes by considering visits to all possible supply points and to all the possible combinations of demand points, we restrict the number of demand points that could be "reached" from each supply point. This means that we only allow the truck to visit a certain customer out of a small number after a visit to a certain supply point.

We start by solving a transportation problem that gives us a possible flow of logs between the supply and demand points. This is performed for each of the assortments, and the transportation problem obtained for a certain assortment is defined as follows. Consider *m* supply nodes with supply s_i and *n* demand nodes with demand d_j. Further, we introduce arcs from the supply nodes to the demand nodes. Every arc (*i, j*) has a cost associated with a unit flow. The aim is to decide a flow w_{ij} on each arc in the network in order to satisfy all demand at a minimal transportation cost and without exceeding the supply. The general formulation of this transportation problem is

$$
\begin{array}{llll}
\min & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} w_{ij} & & \\
s.t. & \sum_{j=1}^{n} w_{ij} & \leq s_i & \text{for } i = 1, 2, \ldots, m \\
 & \sum_{i=1}^{m} w_{ij} & \geq d_j & \text{for } j = 1, 2, \ldots, n \\
 & w_{ij} & \geq 0 & \text{for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n
\end{array}
$$

The solution obtained from the transportation problem is used to decide which demand points that could be reached from each supply point. In this way we make sure that the demands of the customers can be satisfied at the same time as the number of generated routes becomes relatively small. In a similar way, we limit the number of supply points that can be reached from the demand points; for each demand point we choose the two supply points with the closest distance to the demand point. This route generation heuristic can be thought of as an extension of the cluster-first-route-second heuristics that are well-known within vehicle routing. These usually consist of grouping several locations that are geographically close and building routes within the resulting groups, or clusters.

The rules that we have implemented for enumerating all possible routes within the clusters are as follows.

1. Trucks are only allowed to pick up one assortment during a trip.
2. Trucks continue the pick-ups until they are fully loaded.
3. The delivery is made to only one customer. The truck is thus empty after a visit to a customer.
4. The amount loaded at a certain point is decided by minimum of the remaining truck capacity and the quantity remaining at the supply point.

Rules 3 and 4 reflect real-life practice. Most customers have a large demand and it is natural to empty the truck there. It is in real life possible to pick up different assortments, but we have chosen to restrict our heuristic to rule 1 in order to reduce the number of possible feasible routes. When it comes to rule 2, we do allow a route where the truck is only almost fully loaded. If the load of the truck equals the capacity of the truck minus a specified constant, the route is admitted in the pool.

The route generation algorithm proceeds in the following way: it starts at the driver’s home base and visits any one of the supply points to load the truck. The quantity that the truck picks up depends on the amount available at the supply point and the free capacity of the truck. Since the same truck can visit a certain supply point several times during the route, with deliveries to demand points in between, the supply available at that point has to be updated after each visit. The algorithm always loads the truck to the maximum. This means that the quantity loaded at the supply point *i* equals the minimum of the remaining supply at point *i* and the truck’s free capacity, that is

$$\text{load}(i) = \min(\text{ free capacity}, \text{residual supply}(i)).$$

After loading at the supply point there are two possible outcomes, either the truck is fully loaded or it is not. If it is fully loaded the next step is to find a customer that can receive a delivery from this supply point, according to the clustering, and to deliver the load to that customer. During the routing process, the time windows are checked each time a new location is visited and only time-feasible trips are registered. Since the clusters are constructed in such a way that all supply points and demand points that are included in the same cluster agree on the assortment type, there is no need to take that issue into account while designing the trips. If the truck is not fully loaded, the algorithm searches for non-empty supplies in the same cluster, in order to fill the truck. This step is repeated until the truck is full or almost full, after which a customer is visited for unloading.

4.5 Obtaining integer solutions

The optimal solution obtained after solving the LP relaxed restricted master is typically fractional. In order to solve the original integer problem it is therefore necessary to enforce integrality. Branch-and-bound is a classical technique to accomplish this, and the most common way of solving integer problems.

The basic idea behind a branch-and-bound procedure is to partition the set of feasible solutions into smaller sets and search for integer solutions in the subsets instead of in the original set. Each time such a partition is performed, a range of fractional values is excluded from the set of possible solutions. In this chapter, we apply a branch-and-price technique (B&P) that allows new columns to enter the problem within the branch-and-bound tree. For more details about branch-and-price, see for example Barnhart et al. (1998).

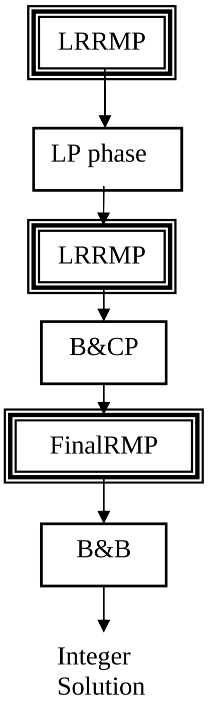


Figure 30: Illustration of the complete algorithm.

Briefly, branch-and-price consists of a *branching* stage and a *pricing* stage. Each time a branching is applied new constraints are added to the problem, giving rise to (two) subproblems. The role of the pricing stage is to price the potential columns and return the column with the least reduced cost, and if this reduced cost is negative the column is added to the problem. In our implementation the pricing stage is performed in the pool of columns and not by solving a column generation problem. We will call our approach *branch-and-composite-price (B&CP).* In practice, new constraints are not added to the master problem. Instead, the set of active columns is reduced by inactivating the columns that do not satisfy the new constraints. This simplification can be made because the routing variables are binary.

An illustration of the complete algorithm is given in Figure 30. The initial relaxed restricted master, LRRMP0, contains only the initial columns mentioned earlier. The LP phase that consists of repeatedly pricing all columns in the pool and adding new columns to the problem, results in the LRRMP1. The final restricted master, FRMP, is obtained after a B&CP phase where a number of columns are added to LRRMP1. Early tests showed that a large number of columns were added in the beginning of the B&CP phase but after a while, the algorithm spent a long time on pricing the columns without adding any to the restricted master. For this reason we have chosen to stop the B&CP scheme after a fixed number of nodes in the branch-and-bound tree. We then apply standard branch-and-bound to the final restricted master for obtaining an integer solution.

The reason for using both B&CP and traditional branch-and-bound is as follows. Applying branch-and-bound directly to LRRMP1 typically gives very poor integer solutions, or even no integer feasible solution at all (in the sense that the demand is not entirely satisfied), because this problem contains too few columns. We thus need the B&CP to find a relatively large set of columns and a good integer solution. We stop including more columns and solve the remaining problem using standard branch-and-bound, because running the B&CP scheme to optimality is much too time consuming. This combination turned out to be the most efficient approach. The problem FRMP generally, in our tests, includes a good set of columns in order to provide a good integer solution. This integer solution is typically not the optimal solution within the subset of routes in the pool.

Several issues have to be considered before running a branch-and-price procedure, the most essential being the branching strategy and the order in which the subproblems are to be inspected and solved. There are several branching strategies for partitioning the set of feasible solutions. As an example of this, Kohl (1995) describes five different branching strategies for the VRPTW. The classical branching strategy when an LP relaxation is considered is variable branching. The idea is to choose a fractional variable, that in our case is required to be binary, and to fix it to be zero on one side of the branch and one on the other. If there are several variables with a fractional value, the variable with the largest value, for example, can be chosen to branch on.

One disadvantage of this branching strategy is that in problems containing binary variables and convexity constraints, like constraints (1) in [GM], this strategy can lead to an unbalanced tree. This means that the branching affects the solution space very little on one side, while the other side is highly effectual. For example, if we branch on a binary variable and choose the side where this variable is set to zero, we would gain little. If we have a large number of variables, this variable (which is set to zero) will most likely be replaced by a variable belonging to a similar column and the lower bound of the objective value would change very little. On the other hand, if we choose the side with variable value one, many constraints would become satisfied and many variables would be forced to take the value zero, leading to little or no remaining flexibility. Difficulties caused by unbalanced branch-and-bound trees are noted in several large scale applications, see e.g. Ryan and Foster (1981) and Barnhart et al. (1998).

4.5.1 Constraint branching

Instead of choosing one single fractional variable and imposing the values zero or one in the branches, constraint branching forces the *sum* of some variables to be zero or one. This branching strategy was suggested by Ryan and Foster in 1981, and the authors' motivation for the strategy is precisely the unbalance that has been described in the previous section. By forcing the sum of a group of variables to be zero or one, the tree tends to become more balanced. Ryan and Foster apply this strategy to the set-partitioning problem [SPP]

$$
\begin{array}{llllll}
\textit{[SPP]} & \textit{min} & \sum_{j\in P} c_j y_j & & & (4.1)\\
 & \textit{subject to} & \sum_{j\in P} a_{ij} y_j & = 1 & \forall i \in C & (4.2)\\
 & & y_j \in \{0,1\} & & \forall j \in P & (4.3)
\end{array}
$$

where the right hand side of each constraint is equal to one and a_{ij} is equal to one or zero. Consequently their theory is based on the fact that any sum of variables covering a pair of constraints satisfies the following inequality:

$0 \leq \sum_{j\in J} y_j \leq 1$ where *J* is the set of indexes connected to variables covering a certain pair of constraints.

The set *J* of variable indexes differs with the pair of constraints considered. In order to choose which set of variables and which sum to branch on, the following procedure can be applied:

1. For each pair of constraints *u* and *v*, define the set of variables that belong to both constraints at the same time. The set J_{uv} of the variable indexes is defined as
$$J_{uv} = \{ j : a_{uj} \neq 0 \text{ and } a_{vj} \neq 0 \}$$
2. Compute the sum of the variables corresponding to each set.
3. Choose from among all sums computed the one that is closest to 1 (but is strictly less than 1). In this step we compute

$\max_{u,v\in I: u\neq v} \sum_{j\in J_{uv}} y_j$ *where I is the set of constraints (4.2).*

In the next one-branch, the sum chosen in step 3 is forced to be one. This means that the pair of constraints corresponding to that set must be covered by the same variables at the same time.

For example if we chose the pair of constraints *a* and *b* for which the sum of the fractional variables is closest to one, then the set of variables y_j for all *j* can be partitioned into two sets. The first set J_{ab} includes the indexes of the variables that cover both constraints at the same time, this set will be called *J*. The set *K* is defined as

$$K = \{ j : a_{uj} \neq 0, a_{vj} = 0 \text{ or } a_{vj} \neq 0, a_{uj} = 0 \}$$

Clearly, the sets *J* and *K* are disjunct and their union forms the set of all variables appearing in constraints *a* and *b*.

Then, the algorithm forces $\sum_{j \in J} y_j$ to be one, which means that $\sum_{j \in K} y_j$ automatically becomes equal to zero. Consequently, constraint branching forces the sum of a group of variables to become one (or zero) instead of fixing the value of *one* variable to one (or zero). In other words the variable branching strategy is a special case of constraint branching. Figure 31 shows the difference between variable branching and constraint branching.

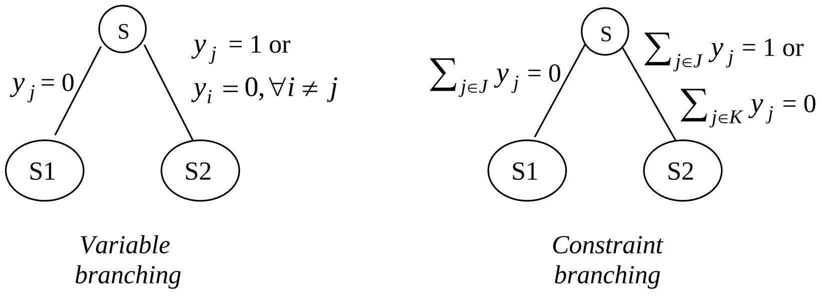


Figure 31: The relation between variable branching and constraint branching.

4.5.2 Constraint branching applied to the LTSP

We note that in the generic model [GM] of Chapter 2, the right hand side constants are only equal to one in the convexity constraints. Since the convexity constraints do not have any variables in common (they include different trucks), we choose to apply this branching strategy to all pairs of constraints containing exactly one convexity constraint. Then the sum of the variables that cover both constraints is less or equal to one, so that the constraint branching strategy can be used.

In practice, a pair of constraints means that a vehicle and a location are chosen to branch on (location is in our problem either a pickup point or a delivery one). For example if the constraints pair corresponds to vehicle *k* and pick-up point *l*, imposing that the sum of the variables covering these constraints should be one is equivalent to forcing vehicle *k* to visit the pickup point *l*. The one-branch does not specify the number of times the pickup point must be visited. It is however visited at least once.

In contrast to variable branching, both branches here lead to significant changes in the problem since the zero-branch forbids the vehicle *k* to visit the location *l*.

Imposing a certain branch is like adding an extra constraint to the problem and in this case, each subproblem is solved after fixing a number of variables to zero. We define the set *J* as the set of all columns corresponding to vehicle *k* that visit the pick-up point *l*.

- In the zero-branch, fix all variables in the set *J* to zero. This forbids the vehicle to visit the location.
- In the one-branch, find the complement set J'. This set contains all the variables corresponding to the vehicle *k* that do not visit location *l*. Fix then all variables included in the complement set J' to zero.

There is an interesting interpretation of constraint branching in general, and specifically in the case of the log truck scheduling problem. Since the sum of the variables is between zero and one, the value of a variable can be viewed as the chance of a truck visiting a specific location. When we seek the constraint pair for which the sum of the variables is closest to one, we then seek the truck-location pair with the highest probability that is strictly less than one. If it is highly likely that the truck visits a location, we hope then that it will do so even in the final solution. This justifies the criteria for choosing the group of variables that has been mentioned earlier. The example in Figure 32 shows how a vehicle/location pair is chosen. In the example, truck A and Demand 1 sum to 0.7 and are therefore chosen to branch on.

Variables value	0.15	0.15	0.25	0.45	$\sum x_j$
Truck A	1	1	1	1	
Supply 1		25	25		0.4
Supply 2	40	15	10		0.55
Supply 3		5	5		0.4
Supply 4	80	75	40	120	1
Demand 1			40	40	0.7
Demand 2		40		40	0.6
Demand 3		80		40	0.6
Demand 4	120		40		0.4

Figure 32: Illustration of constraint branching in the log truck scheduling problem.

Note that it could happen that the constraint branching strategy fails in the log truck scheduling problem. This can occur only if the actual fractional solution consists of columns where exactly the same locations are visited by the same truck. In such a case, which is easy to detect, we switch to variable branching that guarantees the creation of new branches. Variable branching can be seen as a special case of constraint branching. Therefore, it is natural to complement constraint branching by variable branching in case of a failure of the former. Here, it is important to mention that this special case has never appeared during our computational tests.

4.6 Numerical results

The method described in this chapter was tested on problem cases from the two Swedish companies Södra and Sydved.

The first case (CASE 1a) is taken from Sydved, who controls large forestry areas, buys logs from independent forestry owners, and is in charge of the log transportation from different origins to customers. Since Sydved owns a large forestry area, they have divided the area into a number of smaller districts where the transportation planning is done separately. The set of data used in CASE 1a is obtained from one of those districts.

This case is characterised by a large number of supply points that are scattered around the district and in this specific case study the number of supplies is 266 while the number of demands is only 15. The total supply is several times larger than the total demand, but the supply points differ a lot by their available quantities, since some independent lot owners want to sell as little as 3 tonnes of logs while larger supplies can offer up to 800 tonnes. There are 4 assortments and the total quantity is 3239 tonnes. Twenty-eight trucks are available for the transportation. The working days are divided into two shifts, each of which is 8 hours long. The start and the end nodes of the shifts are in this case at the home base, which means that the drivers meet at a home base to change shifts.

Since the supply in CASE 1a is so much larger than the total demand, we created a new case (CASE 1b) by reducing the number of supply points, and consequently the total supply. In this way, the original case is transformed into a more realistic daily planning problem. CASE 1b has a total supply that is approximately the double of the total demand. The results in Table 3 are for CASE 1b.

In the second main case (CASE 2), that is taken from Södra, 6 trucks are used to transport wood chips from 16 supply points to 13 demand points. The total quantity available and required is 1482 tonnes, and the number of wood chips assortments is 5. Moreover, all trucks have the same capacity, namely 38 tonnes. This means that the total number of trucks-loads (or trips required) is equal to 39, since all quantity available at the supply point can be expressed as an integer multiple of the truck capacity. The transportation problem in this case is trivial since all destinations (supply and demand combinations) are fixed in advance, or predefined.

All algorithms are written in C, and CPLEX 6 is used to solve all linear programs and the final integer program. The computer used is a PC with Pentium 4, 1.6 GHz processor. First we study CASE 1a. We chose to interrupt the whole B&CP procedure after a certain number of nodes in the search tree. This is because we noticed that, when approximately 5000 nodes were searched, most pricing operations did not return any new columns and were very time consuming. When 5,000 nodes were searched we turned to branch-and-bound. We have tested to use both an own branch-and-bound implementation and the standard CPLEX MIP solver. The latter was faster for CASE 1a as well as for CASE 1b.

In CASE 1a, Sydved had used the whole fleet of 28 trucks in order to satisfy the total demand. This number could be reduced to 19 trucks in the solution obtained. This reduction is due to the nature of the routes in the solution, within which more trips were performed. In Sahlin (1998) results were reported for the same case, and the number of trucks used in Sahlins solution was 23.

In order to test the quality of the clusters suggested in the algorithm, we designed other clusters and routes, and solved the problem using the new sets of columns. In this test the clusters were designed according to simple rules. These were: each supply point is connected to the two closest demand points requiring the correct assortment, and each demand point is connected to the two closest supply points and a number of randomly chosen ones, with correct assortment. Table 1 shows that the LP solution improved with the number of the columns in the pool. We varied the number of randomly chosen supply points in order to change the number of columns in the pool. The improvement achieved with the number of columns in the pool was not enough to satisfy all demands, not even for the case where the number of columns generated exceeded 3 millions.

# of columns in the pool	**LP solution**	**Satisfied demand?**
174,373	1,204,554	No
365,373	752,972	No
433,412	661,273	No
3,154,877	562,739	No

Table 1: Testing the quality of the routes for CASE 1a.

These results can be compared to Table 2, where an LP solution where all demands are satisfied was found without having to generate more than 134,636 columns. The columns in this test were generated according to the heuristic described in Section 4.4.

# of columns in the pool	**LP solution**	**Satisfied demand?**
134,636	397,377	Yes
275,351	357,757	Yes
1,335,577	355,521	Yes
2,572,408	351,915	Yes
2,884,531	351,911	Yes

Table 2: Testing the effect of the number of routes generated in CASE 1a.
The clusters do not change from one test to another. In order to modify the number of columns in the pool we varied the number of supply points that are possible to visit

directly after leaving a demand point. Table 2 shows that the LP solution does not improve much with the number of columns in the pool. The whole algorithm was applied to CASE 1b. The integer solutions registered in the third column were found by applying CPLEX MIP solver to the restricted master created after the LP phase and B&CP. We tried to use the CPLEX MIP solver without running B&CP first, but in that case no integer feasible solutions (where all demands are fulfilled) were found. The running time given in the fifth column is the total running time for the whole algorithm including both B&CP and the CPLEX MIP solver. The integer solutions given by CPLEX are optimal within the set of columns included in the final restricted problems.

# of columns in the pool	**LP solution**	**IP solution B&P and MIP solver**	**# of added columns in B&P**	**CPU time in sec.**	**CPU time for pool generation**
160,363	993,225	1,160,927	656	174	2
353,416	947,599	1,007,894	840	237	7
975,561	943,106	987,980	1,521	1,002	23
1,876,773	913,018	987,127	1,114	1,671	57

Table 3: Results for the whole algorithm applied to the second case CASE 1b.

The corresponding results for CASE 2 are as follows. The number of routes generated is 2.9 millions. The results obtained are compared to the routes that were actually used in real-life. In order to make this comparison possible we computed the total distance driven by each truck. Each route consists of loaded trips, unloaded ones, and trips originating or terminating at the home base. It is important to note that the actual routes are in this case well structured and carefully planned. In fact, the whole set of data for the third case is constructed using the actual routes as a basis. This was done by observing and registering the trips made and the locations visited, and then including these locations in the data and excluding all others. In this way this set of data favours the manual solution, since a larger degree of freedom or a larger number of supply points usually lead to better solutions. The results are shown in Table 4 below, where substantial savings were can be observed.

	Actual routes	Scheduling solution	Difference between the two solutions	
Total driven distance	7,984 km	7,314 km	-670 km	-8.4%
Loaded distance	4,456 km	4,157 km	-299 km	-6.7 %
Unloaded distance	2,599 km	2,452 km	-147 km	-5.6 %
Home journeys	929 km	705 km	-224 km	-24 %
Total loaded percentage	55.8 %	56.8 %		+1 %

Table 4: Results of CASE 2.

4.7 Conclusion

The log truck scheduling problem involves a huge number of potential routes. In this chapter we proposed a cluster-first-route-second heuristic to design feasible routes based on building clusters that are derived from the solution of a transportation problem. This clustering technique has shown to be critical in order to get high quality integer solutions.

As a solution method, we suggested the use of a branch-and-composite-price scheme, which is based on branch-and-price but where the column generation is replaced by composite pricing. As branching rule we apply constraint branching. The interpretation of this branching is to force a logging truck to visit a particular pickup and delivery point. This approach has not been used in LTSP earlier. The solution times are acceptable for daily planning purposes, as shown by studies of cases from major Swedish forest companies. Compared to manual solutions and a simpler heuristic, the method studied in this chapter can construct and select efficient routes and substantially reduce the number of trucks required.

5 Near-exact column generation by constrained shortest paths

5.1 Introduction

The aim of this chapter is to present a near-exact method for solving the LTSP. We use the generic model [GM], the column-based model where each column represents one feasible route for one truck. A feasible route is a route originating and ending at the driver's home base and that satisfies a number of restrictions. Since the number of feasible routes is large we apply a column generation algorithm and solve a constrained shortest path problem in order to generate new columns that can be added to the model. The constrained shortest path problem, or CSPP, is solved exactly by a k-shortest path enumeration algorithm. When the value of the constant "k" is sufficiently large, k-shortest path guarantees that the CSPP is solved exactly by finding a feasible route with the smallest reduced cost.

Branch-and-price is applied in order to obtain an integer solution. Since solving the CSPP exactly is time consuming, we choose to use a restricted number of columns in the pricing procedure of branch-and-price. This is one of the reasons the method is referred to as near-exact. This chapter focuses on the modelling and solution of the CSPP that arises in the LTSP. Numerical results from a case study are presented.

5.2 General framework

As for all routing and scheduling problems, there exist a wide range of heuristic and exact methods for the PDPTW and its variants. We refer to a survey by Cordeau and Laporte (2003) on the dial-a-ride problem, which is a variant of the PDP that is concerned with picking up people instead of goods. Particularly we mention the traditional heuristic cluster-first-and-route-second that involves grouping requests that can be serviced by the same vehicle before deciding upon the route of the vehicle. Clustering is usually based on geographical closeness among locations.

Amongst the exact methods, column generation seems to be a popular approach for pickup and delivery problems. Dumas, Desrosiers and Soumis (1991), Sigurd, Pisinger and Sig (2004), and Xu et al. (2001) are examples of papers where column generation is used. In the first of these papers the column generation subproblem, which is a constrained shortest path problem, is solved exactly by dynamic programming, while in the second and third, heuristic and exact approaches are presented.

This chapter presents a column generation approach for the log truck scheduling problem. The solution approach consists of two main phases:

Phase 1: The LP relaxation of [GM] is solved by column generation.
Phase 2: Integer solutions within the set of columns generated in the first phase are found. This is achieved by applying branch-and-price, which allows new columns to enter the restricted master problem during the branch-and-bound procedure.

5.3 Phase 1: The algorithm for solving the LP relaxed problem

The method for solving the LP relaxation of [GM] is the following:

1. Initialisation.
2. Design an initial set of feasible routes.
3. Solve the LP relaxation of the model within the set of routes generated in step 2.
4. Repeat for each truck:
   - Construct the k-shortest path network. Use the dual variables obtained in step 3 to compute the arc costs in the network.
   - Solve the subproblem for a certain *k* and return feasible columns with negative reduced costs. If no such columns exist go to the next truck.
   - Add the columns obtained to the restricted master problem, LRRMP, and reoptimise.
5. Terminate according to some stopping criteria.

Step 1: The initial model

The purpose of the initial model is to provide a set of dual values that can be used to price the initial set of columns. The initial model basically consists of one column for each truck, expressing that the truck is staying at home for the whole day. The solution to this initial problem is "feasible" and the resulting cost equals the sum of all penalties that are due to not transporting any logs. Most important of all, the solution also gives a set of dual variables values that are necessary for step 3.

Step 2: Designing an initial set of feasible routes

The initial set of feasible routes is designed a priori and independently of step 1. Here, the heuristic described in Chapter 4 is applied. The main idea is to limit the number of locations that can be visited or reached from each supply or demand point by creating clusters. Clusters are determined by both solving the classical transportation problem and using the geographical closeness between different points. These clusters result in two types of sequences:

1. list of supply points that are within a certain distance from each demand point
2. list of demand points that can receive logs from each supply point.

Once the clusters are determined, an enumeration process starting at the home base of each truck generates routes while assuring that all constraints are satisfied. The process starts by letting each truck visit each of the supply points, then the second list is used to decide the demand point that can be visited next. Supply points that can be visited after a certain demand point are found in the first list. Each time a new destination is reached, the arrival time is computed and the time window constraint is checked. The truck always picks up the largest amount possible; this amount depends on the supply available at the time of the visit and on the free capacity of the truck. In this way the capacity constraint and the supply constraints are respected.

The number of feasible routes that can be generated in this phase can be adjusted since, in practice it depends on the length of the first list. The situation in most forestry case studies is that the number of supply points is typically much larger than the number of demand points. Finally, the set of feasible routes and its corresponding set of columns are stored in a pool.

Step 3: The pricing procedure

The goal of the pricing procedure is to choose columns from the pool to be added to the restricted master. To begin with, the dual variables obtained from the solution of the initial model are used to price all columns in the pool. By pricing we mean to compute the reduced cost of a column. The reduced cost for each column is generally given by the expression

$$\bar{c}_j = c_j - \pi^T a_j$$

where c_j is the cost of the column, π the dual values obtained from the LP solution, and a_j the constraint coefficients of the column. In our case a_j represents the amounts loaded or unloaded at any location during the route. The reduced cost of a route can be interpreted as the value we gain by visiting the supply or demand points that each has its reward expressed through the dual values. Each time the vehicle visits a supply or demand point a price that is expressed by the dual value corresponding to that point and the amount picked up or delivered is collected. A negative reduced cost corresponds to a total gain and a positive to a loss. The pricing procedure works as follows:

1. Use the dual values to compute the reduced costs of all columns in the pool and find the column with the least reduced cost.
2. If the smallest reduced cost is negative, add the column to the restricted master and reoptimise it, update the dual values and go to step 1. Otherwise, Stop: the optimal LP solution within the set of columns in the pool has been found.

In fact, in Step 2 it is more efficient to add a larger number of columns with negative reduced cost to the restricted master problem instead of only one.

5.4 The column generation subproblem

The subproblem aims at generating feasible routes that can be added to the restricted problem LRRMP. There is one subproblem for each truck even if the fleet is homogeneous. This is due to the fact that different trucks have different home bases and the distances in the first and last trips depend on these locations. The objective function of each column generation subproblem is formed by the dual variables obtained from solving the LRRMP.

The purpose of the subproblem is to find a minimal reduced cost route that respects a number of rules. We consider the network illustrated in Figure 33, where nodes *0* and *n+1* represent the home base of the truck. Finding a feasible minimal reduced cost corresponds to finding a constrained shortest path from node *0* to node *n+1* in the network, as described below. The constrained shortest path problem is NP-hard and it is usually solved by algorithms based on dynamic programming. The constrained shortest path problem includes problems like the shortest path with time windows or the resource constrained shortest path problem. The subproblem of the LTSP can be modelled as a resource constrained shortest path problem.

The constraints that must be satisfied for obtaining feasible routes are as follows:

- If the truck visits a certain location it has to leave it after the visit.
- Time windows: the start and end time of service at each location.
- Precedence constraints: the truck can not deliver before picking up the logs.
- Start and end location at the home base.
- Capacity constraints: the truck's capacity can not be exceeded.
- Coupling constraints: deliver the correct assortment that is demanded at the demand points.
- Different assortments should not be mixed in the different blocks of the truck.
- The total amount picked up at a supply point should not exceed the amount available there.

Most of these constraints can be handled by defining a specific network for each truck. This network, that is illustrated in Figure 33, is similar to the one defined in Chapter 2, Section 2.3.1, except that it is here defined for one truck only.

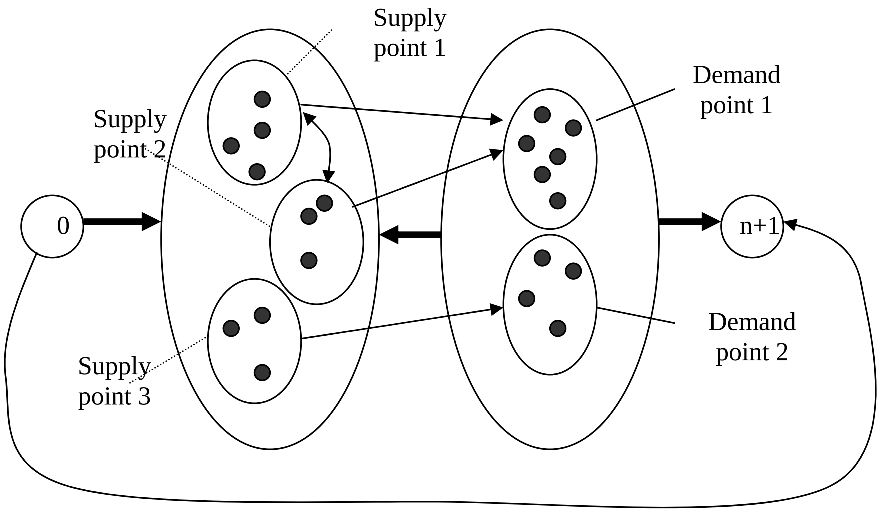


Figure 33: Illustration of the subproblem for one truck.

The single-truck network makes it possible to take into account a number of constraints implicitly. For example the arcs originating in node 0 are only connected to supply nodes or node *n+1*. In this way, vehicles are not allowed to drive directly to a customer before loading at any supply node. Another example is how the coupling constraint is imposed in the network; arcs from a supply point with a specific assortment connect to a demand that requires the same assortment type. In this way most constraints become implicitly included in the network. (The subproblem can also be made much more general by adding arrows between different delivery points, allowing the truck to deliver a part of its load to one customer and the remaining part to a new customer.)

In order to model the subproblem we need the following notations:

S	=	set of the original supply points
S*i*	=	set of all supply nodes representing supply point $i \in S$
AS	=	set of all supply nodes $= \bigcup S_i$
D	=	set of the original demand points
D*j*	=	set of all demand nodes representing demand point $j \in D$
AD	=	set of all demand nodes $= \bigcup D_j$
A	=	set of all arcs in the network
N	=	$AS \bigcup AD$

The variables used in the model are as follows:

- x_{ij} = 1 if the arc (*i,j*) is used, 0 otherwise
- y_i = quantity picked up at supply point *i*, for $i \in AS$
- q_{ij} = load of the vehicle when traversing arc (*i j*)
- s_i = arrival time at node $i \in N \bigcup \{n+1\}$

The additional notations and constants used in the model

- t_{ij} = the time it takes to travel on arc (*i*, *j*)
- c_{ij} = travel cost from node *i* to *j*. This cost depends on the load of the truck.
- [a_i, b_i] = time window for each node $i \in N \bigcup \{n+1\}$
- *p* = capacity of the vehicle
- y_i = quantity picked up at supply node *i*, for $i \in$ **AS**

In this model we assume that a truck *only* visits a copy of a supply point if it is able to pickup the whole amount available at that copy. For this reason the copies of a supply point are associated with the quantities y_i .that represent units of tonnes. We also assume that only fully loaded trucks are allowed to visit the demand points. Let v_i, w_k, and z_p denote the dual variables associated with the three sets of constraints in the LP relaxation of [GM]. The column generation subproblem [SUBk] of the log truck scheduling problem for a given truck *k* can be formulated as follows

minimise

$$\sum_{(i,j)\in A} c_{ij}x_{ij} - \sum_{i\in AS} w_i y_i - \sum_{(i,j)\in A: j\in AD} z_j q_{ij} - v_k$$

subject to

$$\sum_{j:(0,j)\in A} x_{0,j} = 1 \qquad (5.1)$$

$$\sum_{l:(l,i)\in A} x_{li} - \sum_{j:(i,j)\in A} x_{ij} = 0 \qquad \forall\, i \in \mathrm{N} \tag{5.2}$$

$$\sum_{i:(i,n+1)\in A} x_{i,n+1} = 1 \tag{5.3}$$

$$q_{0,j} = 0 \qquad \forall\, (0\,,j) \in \mathrm{A} \tag{5.4}$$

$$q_{i,n+1} = 0 \qquad \forall\, (i\,,\, n{+}1) \in \mathrm{A} \tag{5.5}$$

$$\sum_{j:(i,j)\in A} q_{ij} = 0 \qquad \forall\, i \in \mathrm{AD} \tag{5.6}$$

$$x_{ij}\Big(\sum_{l:(l,i)\in A} q_{li} + y_i \Big) = q_{ij} \qquad \forall\, j \in \mathrm{N}, \forall\, i \in \mathrm{AS} \tag{5.7}$$

$$q_{ij} \leq p x_{ij} \qquad \forall\, (i\,,j) \in \mathrm{A} \tag{5.8}$$

$$\sum_{i\in S_j} y_i \leq r_j \qquad \forall\, j \in \mathrm{S} \tag{5.9}$$

$$s_0 = a_0 \tag{5.10}$$

$$x_{ij}\left(s_i + t_{ij}\right) \leq s_j \qquad \forall\, (\,i\,,j\,) \in \mathrm{A} \tag{5.11}$$

$$a_i \leq s_i \leq b_i \qquad \forall\, i \in \mathrm{N} \cup \{n+1\} \tag{5.12}$$

$$s_i\,, y_i \geq 0 \qquad \forall\, i \in \mathrm{N} \tag{5.13}$$

$$q_{ij} \geq 0 \qquad \forall\, (\,i\,,j\,) \in \mathrm{A} \tag{5.14}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, (\,i\,,j\,) \in \mathrm{A} \tag{5.15}$$

Constraint (5.1) forces the vehicle to leave its home-base and (5.3) forces it to return home at the end of the route. If the vehicle ever visits any node in the network, then it must also leave that node. This is expressed in constraint (5.2). The load variables are initialised to zero in constraint (5.4) for all arcs originating at the home base, since the vehicle is empty when it leaves its home base. In constraint (5.5) the loads are forced to zero for the trip home. We assume that a truck is totally emptied at the demand point. This assumption agrees with real life situations and it is expressed in constraint (5.6). The only time the load of the truck is non-zero is after leaving a supply point. The load then depends on the load of the truck prior to the visit and on the quantity associated with the supply point visited. In constraint (5.7), the load of the truck is determined by the load of the truck directly before it reaches node *i* and the constant quantity picked up at node *i*.

Constraint (5.8) forces the truckloads not to exceed the truck capacity. The total amount picked up at all supply nodes representing supply point *j* should not exceed the existing amount (constraint (5.9)).The time variable s_i is decided in constraints (5.10) to (5.12). The start time for the vehicle is initialised in (5.10). The arrival time at each node is determined in (5.11) and finally, constraint (5.12) assures that the time windows are respected for each node in $\mathrm{N} \cup \{n+1\}$.

The model above makes it possible to load any quantity as long as the supply and the free capacity of the truck permit it. As previously mentioned, precedence and coupling constraints are dealt with in the network. The problem can be made more general by allowing the truck to load several assortments, but we have chosen to focus on the case where only one assortment is permitted. The number of copies needed for each supply or demand point depends on the case the model is used for.

It is important to note that the model is not linear. The objective is to minimise the reduced cost and this cost is not linear since it depends of q_{ij}. The model also contains non-linear constraints, (5.7) and (5.11), that can however be made linear. The subproblem is resolved after solving each master problem with the existing columns. For this reason, it is important to solve the subproblem rather quickly. Dynamic programming is today one of the most popular techniques for solving the constrained shortest problem see e.g. Ioachim et al. (1998). This problem has appeared as a subproblem in a number of vehicle routing problems such as the VRPTW and PDPTW. Even though the constraints here differ somewhat from those considered in earlier applications, the dynamic programming method is applicable also in our case, in principle.

Nevertheless dynamic programming is in our case impractical. This is due to firstly the load decisions and secondly the time windows that must be respected. Since it is possible to pick up any quantity between 0 and 40 tonnes (assuming that the truck capacity is 40 tonnes), the number of possible states becomes too large. Moreover, the number of states gets even larger when taking the time aspect into account: a truck can arrive to a location at any time within the time window. Besides that, the constraints assuring that the amounts picked up at the supply points do not exceed the total supplies lead to a large number of additional states (equal to the number of supply points).

5.5 Solving the subproblem

In this section we describe step 4 in the general algorithm of Section 5.3.

5.5.1 Constructing the network

In order to solve the constrained shortest path problem we start by constructing a network where most of the complicating constraints are embedded. We define the following network:

- The start node and end node are the driver's home base and are represented by node *0* and node *n+1* (where *n* is the total number of supply and demand nodes in the network).
- Each supply point is represented by a number of copies, each of which can be visited at most once. Each copy of a supply point is characterised by a specific quantity and a specific time window.
- Each demand point is also represented by a number of demand copies that can be visited at most once. Each copy of a demand point is characterised by a specific time window.
- One arc connecting the home base node 0 to *n+1*. This arc allows the vehicle to stay at home.
- Arcs connecting the home base 0 to all supply nodes in the network that are reachable within their time window.
- Arcs connecting each supply node to all compatible supply nodes that are reachable within their time window.

- Arcs connecting each supply node to all compatible demand nodes that are reachable within their time window.
- Arcs connecting each demand node to all supply nodes that can be reached on time.
- Arcs connecting all demand nodes to node $n+1$ if the time window allows it.

Two nodes are compatible if they offer/require the same assortment type. The copies of each supply point are created by firstly discretising the quantities that can be picked up at that node and secondly discretising the time window.

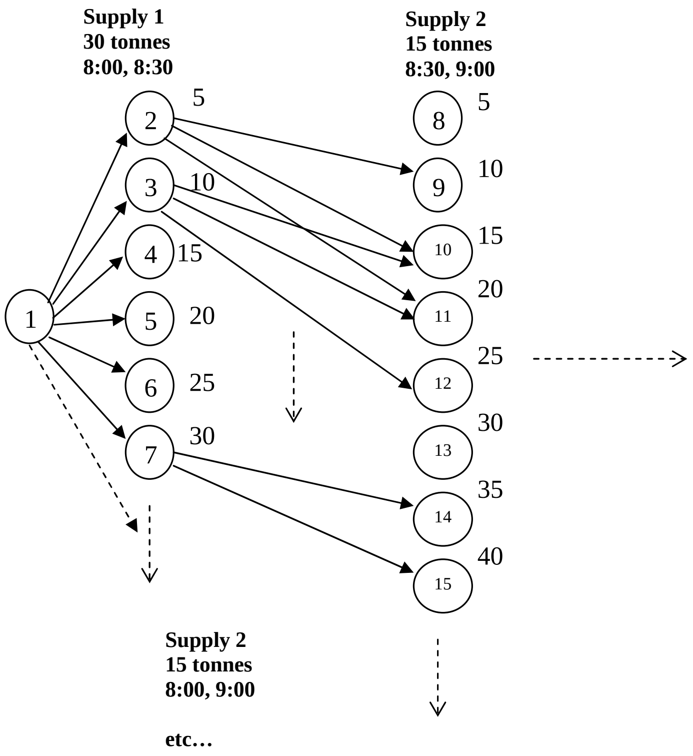


Figure 34: The network used to compute the *K*-shortest paths

This is achieved by dividing each time window in time intervals of appropriate lengths, for example, 30 minutes, and by deciding the possible loads that the truck can carry at different positions in the network. For example if the capacity of the truck is 40 tonnes, we restrict the quantities that can be picked up at a certain supply point to 5, 10, 15, up to 40 tonnes. We also assume that the truck is empty after visiting a demand point. Thus, each copy of a supply point is associated with a tight time window and to a number that describes the truckload after visiting that node. This number is equal to 5, 10, 15 up to 40 since the truck capacity is 40. In this way the truck capacity constraint is respected.

The network structure described above is illustrated in the example in Figure 34. This structure also assures that the time window constraints are satisfied. This is done by computing the latest arrival time from each node *i* to each node *j*. If this time lies within the time window (or before the opening of the time window) of node *j* then the arc is feasible and it is added to the network, otherwise it is not. We conclude that a shortest path found in the resulting network will automatically satisfy all constraints mentioned earlier except one, namely the constraint ensuring that the total amount picked up during the route (or path) at a certain supply point does not exceed the supply. This restriction will be taken into account by applying a *K*-shortest paths algorithm.

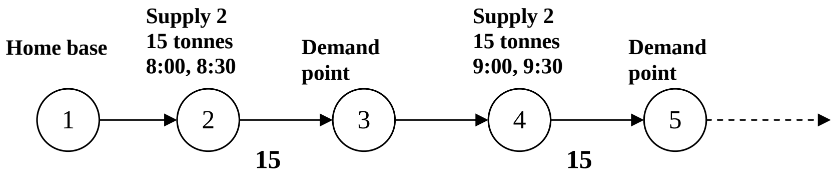


Figure 35: Example of an infeasible path.

The example in Figure 35 shows a situation where the latter constraint is violated. In this example, both nodes 2 and 4 represent the same supply point, and the load of the truck is 15 tonnes after visiting any of those nodes. In this path the truck picks up 15 tonnes between 8:00 and 8:30 and again between 9:00 and 9:30, which results in an infeasible route since the total amount available at supply point 2 is 15 tonnes only.

In order to solve the constrained shortest path problem we define the arc costs in such a way that the total cost of a path (or route) is equal to the reduced cost of the route. For this reason we define the cost of arc (*i*, *j*) as the following: $\tilde{\tau}_{ij} = \tau_{ij} - \pi_j a_j$, where τ_{ij} is the actual cost of travelling between nodes *i* and *j*, π_j is the dual value corresponding to node *j*, and a_j is the amount picked up or delivered at node *j*. The reduced cost in step 4 is computed as the sum of the costs $\tilde{\tau}_{ij}$ of all arcs belonging to the path, minus the dual value v_k for the truck under consideration.

The *k*-shortest paths algorithm finds not only the shortest path, but also the second shortest, third, etc..., up to the k^{th} shortest path in the network, where *k* is a given number. If the shortest path is not feasible, then there is a chance that the second or third shortest is. In this context it is important to note that the network described above does not contain cycles.

5.5.2 The *K*-shortest paths problem

As its name indicates, the *K*-shortest paths problem is closely related to the well-known shortest path problem ; given a digraph with *n* nodes (where *s* is the start node and *t* the terminal node), *m* arcs and a cost associated to each arc, the problem consists in finding the shortest path from node *s* to *t*. Finding the *K*-shortest paths in the same network involves identifying the second, the third, up to the K^{th} shortest path from *s* to *t* (where *k* is a given constant). The *K*-shortest paths problem was studied already in the late sixties (Dreyfus 1969) and it is of great interest in many practical applications, see, for example, Yen (1971), Brander and Sinclair (1995), Eppstein (1998), and Katoh, Ibaraki, and Mine (1982). There are various reasons for this.

- In some applications, the shortest path in the network must satisfy a number of additional constraints. By computing a number of shortest paths we can find a shortest path that meets these requirements.
- Sometimes a K-shortest paths algorithm is used to find a solution to the subproblem in a column generation procedure. This is advantageous since such algorithms find many feasible columns with negative reduced costs at once (by solving one subproblem). Adding many columns in each iteration might reduce the number of times the column generator needs to be invoked, before an optimal solution is found.
- In some situations, computing several shortest paths creates alternative (near-) optimal solutions, which gives the user the possibility of choosing between several good solutions.

There are two types of K-shortest paths problem: the unconstrained and the constrained. In the constrained problem, the shortest paths have to satisfy some constraints, for example no nodes are repeated. This particular problem is known as the - *shortest elementary path problem*. In this thesis it is the unconstrained problem and the existing algorithms for solving it which are of interest.

5.5.3 Overview of existing algorithms for *K*-shortest paths

There are two major classes of K-shortest paths algorithms, namely the class based on the *Optimality Principle* and the class based on the construction of the *K-shortest paths tree*. In this section, a short explanation of the optimality principle and the K-shortest paths tree is given together with the algorithms using these two concepts.

Optimality Principle

The optimality principle for the shortest path problem is true in graphs with no negative cycles. It is easily stated for the single shortest path problem (that is for K=1); *optimal paths must have optimal subpaths*. When K>1 a more general form of the optimality principle is used: *any K^{th} shortest path includes subpaths from the j^{th} shortest path, for some $j \leq K$*. The proof of the optimality principle is given in Martins et al. (1998).

The algorithms based on the optimality principle are of two types: *labelling* algorithms and *deletion* algorithms. The first type is a generalisation of the labelling algorithms for the shortest path problem. As in the case for the shortest path problem, literature about the K-shortest paths distinguishes between the label setting and label correcting approaches. These two approaches have the use of label lists in common. Each label can involve one or several components that are associated with each node, where each component is used to save some path information. Another way is to enlarge the set of nodes in order to assign a single component label to each node. In this latter case, the label represents the cost or length of a certain path from node s to the node in question.

The difference between the label setting and the label correcting algorithms consists in the choice of the nodes that are to be examined. The label setting algorithm for the

K-shortest paths problem is a generalisation of Dijkstra's algorithm for the shortest path problem, see for example Martins et al. (1998) for a description of the algorithm. The label setting algorithm always picks the node with the label of least value (cheapest cost or shortest length), and in this way a node is labelled permanently just as in Dijkstra's algorithm.

In label correcting algorithms, the node chosen does not have to be permanently labelled. There are several ways of choosing the next node from the set of candidate nodes. One way is to pick the first element in a FIFO list. This results in the general form of the Bellman-Ford-Moore method, see Cherkassky, Goldberg and Radzik (1996). It is important to note that in this case, the set of the *K*-shortest paths is determined first after the execution of the whole algorithm. In contrast to that, the label setting algorithm builds up this set during the execution of the algorithm.

The second type of algorithms based on the optimality principle is deletion algorithms. The idea here is to use the efficient single shortest path algorithms, to find the *K*-shortest paths. This is achieved by firstly solving the single shortest path problem in the original graph and secondly creating a new enlarged graph by modifying the original one in the following way.

1. The shortest path is deleted from the modified graph.
2. No new paths are added.
3. No other paths belonging to the original graph are deleted.

In this way, a new graph, G_2, is created and the shortest path in this new graph is the second shortest path in the original graph. This procedure is repeated and graphs G_3 ,G_4 ,…, G_K are constructed. The single shortest path solution in G_K is the *K*th shortest path of the original graph and the algorithm terminates. Martins (1984) first suggested the deletion algorithm. The disadvantage of this algorithm is its space requirements, since the graphs are enlarged from one step to another. Martins and Santos (2000) improved this algorithm by reducing the number of arcs in the modified graph, and further improvements were made in Jimenez and Marzal (1999).

K-shortest paths tree

We start by illustrating the basic result that the algorithms included in this class are based on; the proof is found in for example Martins, Pascoal and Santos (1998). This result is illustrated in Figure 36 below, where a network and the corresponding *K*-shortest path tree for *K*=3 are given. The first shortest path is

$p_1 = \{1 \cdot 2 \cdot 4 \cdot 6\}$, the second $p_2 = \{1 \cdot 2 \cdot 3 \cdot 4 \cdot 6\}$ and the third $p_3 = \{1 \cdot 3 \cdot 4 \cdot 6\}$.

Node 2 in the first shortest path is called the *deviation node* for the second shortest path. Similarly, node 1 is the deviation node for the third shortest path. As one can note, the nodes could be repeated within the tree. The *deviation path* of the second shortest path is the path from node 2 to 6. Note that the deviation path is the first shortest path from the deviation node 2 to the final node 6 that does not use any arc of the first shortest path as the first arc.

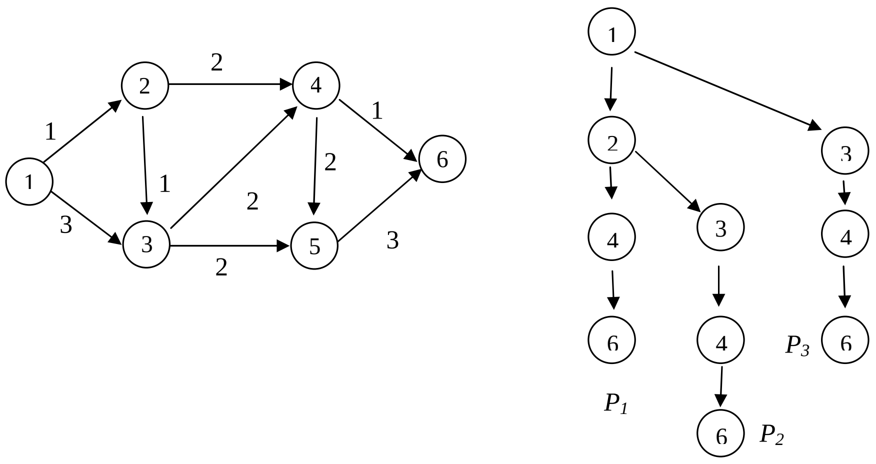


Figure 36: Example of the K-shortest paths tree

The *deviation algorithm*, which is the basic algorithm in this class, is based on the *K*-shortest tree and the deviation node concepts. The idea is to construct the *K*-shortest path tree, starting by computing the shortest path from every node in the graph to *t*. Then, the first shortest path is found and a list of deviation paths that are candidates for the second shortest path is determined. The following steps are repeated for all $k < K$

- For each deviation node *v* belonging to p_k (the *kth* shortest path from *s* to *t*), determine the shortest path from *v* to *t* such that this path does not use an arc of the actual *k* shortest path tree as its first arc.
- Add all deviation paths to the list of candidates.
- $k = k + 1$
- The shortest path among the alternatives in the candidate list is the *kth* shortest path.

Note that in each step, the algorithm only considers the nodes included in the deviation part of the *kth* shortest path, since the candidates of the first, second, …, (*k*-1)*th* shortest paths have already been computed and added to the list.

There are several variations of the deviation algorithm. The one described above examines all possible deviations from each analysed node. Another possibility is to consider only the shortest deviation path for each node. Eppstein (1999) proposed a variation that constructs a graph representing all possible deviations from the shortest path. This deviation algorithm is considered to be among the most efficient *K*-shortest paths algorithms.

The algorithm that has been used in our work is called the *recursive enumeration algorithm*, or REA. REA belongs to the first type of *K*-shortest paths algorithms, that is REA is based on the optimality principle.

The recursive enumeration algorithm

The theory behind the recursive enumeration algorithm is the generalisation of Bellman's equations for the single shortest path problem. This theory makes use of a set of recursive equations that makes it possible to compute the *K*-shortest paths from *s* to every node *v* in the graph. The following notations are used in the equations and in the example shown at the end of this subsection:

- $L^k(v)$ = length of the *k*th shortest path from *s* to *v*.
- $\pi^k(v)$ = the *k*th shortest path from *s* to *v*.
- $L(\pi)$ = the length of path π.
- $C^k(v)$ = the set of all paths that are possible candidates to the *k*th shortest path.

The generalisation of Bellman's equations is stated and proved in Jimenez and Marzal (1999). These can be stated as follows.

The following equations are true for all nodes v of the graph,

$$L^k(v) = \begin{cases} 0 & \text{if } k=1 \text{ and } v=s; \\ \min_{\pi\in C^k(v)} L(\pi) & \text{otherwise}; \end{cases}$$

$$\pi^k(v) = \begin{cases} s & \text{if } k=1 \text{ and } v=s; \\ \arg\min_{\pi\in C^k(v)} L(\pi) & \text{otherwise}; \end{cases}$$

where $C^k(v) = \{\ \pi^1(u).v : u \in \text{the set of all arcs terminating at } v\ \}$ *if k = 1 and* $v \neq s$*, or k = 2 and v = s, otherwise*

$$C^k(v) = \left(C^{k-1}(v) - \{\pi^{k'}(u)\cdot v\}\right) \cup \{\pi^{k'+1}(u)\cdot v\},$$

where u and k' *are the node and index that form the (k-1)th shortest path to v, i.e.* $\pi^{k-1}(v) = \pi^{k'}(u)\cdot v$*, if this path exists.*

The algorithm developed by Jimenez and Marzal (1999) utilises a recursive procedure, *NextPath*, which identifies the set of candidates to the next shortest path. The REA is illustrated using the example in Figure 36.

Initialisation step: The shortest paths from node 1 to every other node in the graph is computed.

$\pi^1(2)$ = 1.2 (Sequence of nodes 1 and 2 and arc (1,2))
$\pi^1(3)$ = 1.2.3
$\pi^1(4)$ = 1.2.4
$\pi^1(5)$ = 1.2.3.5

π^1 *(6)* = 1.2.4.6

Iteration 1: k = 2

C^2 (6) = $\{\pi^1(5)\cdot 6, \pi^1(4)\cdot 6\}$, but π^1 *(4)*.6 is the first shortest path from node 1 to 6, and therefore it is excluded from the set of candidate paths and the second shortest path from node 1 to 4 and 4 to 6 is added to the set. This implies

$$C^2(6) = \{\pi^1(5)\cdot 6, \pi^2(4)\cdot 6\}$$

Since π^2 (4) is not computed, *NextPath(4,2)* is called and π^2 (4) = 1.2.3.4 is found. The shortest path in *C^2 (6)* is π^2 (4).6 .

Iteration 2: k = 3

C^3 (6) = (*C^2 (6)* - π^2 (4).6) ∪ $\{\pi^3(4)\cdot 6\}$ = $\{\pi^1(5)\cdot 6, \pi^3(4)\cdot 6\}$

π^3 *(4)* is computed by calling *NextPath(4,3).* The path 1.3.4 is returned. The third shortest path from node 1 to 6 is the shortest path in *C^3* (6) . π^3 *(4)*.6 is the shortest.

k = *K, terminate.*

Experimental tests have shown that the REA is particularly suitable for networks where the shortest paths are composed of a small fraction of the existing nodes, which is the case in our application, due to the property of the network. In other words, REA is especially suitable for our case and we use the recursive enumeration algorithm developed and implemented by Jimenez and Marzall (1999).

The *K*-shortest paths algorithm is invoked for each truck and the generated paths are checked for feasibility. A certain number of the feasible paths with negative reduced costs is added to the restricted master, which is resolved with the new columns. The resulting dual variables are sent to the following column generation subproblem and its corresponding network.

Lower bound and stopping criteria

When the subproblems are solved in the order described in the general algorithm no lower bounds are produced. In order to compute a lower bound we need to solve one column generation subproblem for each truck using *the same dual values* as input for all trucks. The optimal value of each subproblem, which is the reduced cost of the first feasible route, is used to compute the lower bound. This is done by adding the sum of the optimal values of all the subproblems [SUBk] to the current LP optimal value of the restricted master problem, (e.g. Vanderbeck and Wolsey, 1996).

Since in our case, the subproblem requires a rather long running time, and reaching LP optimality is time-consuming, we choose to interrupt the LP phase after a fixed number of iterations. (We mean by iteration, solving the *k*-shortest path problem once for each truck and reoptimising the LP in between.) Given the non-optimal LP solution, we solve the subproblems, one for each truck, and compute a lower bound adding the reduced costs to the LP solution.

Here it is also possible to choose among different stopping criteria depending on the time factor: in order to prove LP optimality a long running time is required. We choose to interrupt the LP phase after a predefined number of iterations.

5.5.4 Phase 2: Obtaining integer solutions

During the first phase a large number of columns is generated: the columns generated a priori and the ones generated by the column generator. Only a small number of these columns needs to be added to the restricted master for obtaining the LP solution. If branch-and-bound is applied directly to the restricted master problem, there is a large risk that the integer solution found is far away from the optimal one. It is even possible that it is not feasible, in the respect that all constraints are satisfied without using the penalised slack variables.

In order to find a near-optimal integer solution, we choose to apply branch-and-price, which allows new variables or columns to enter the problem within the branch-and-bound tree. For this reason we store a number of routes generated by the *K*-shortest paths algorithm in a pool together with the columns generated a priori. This pool is then used in the branch-and-price method to produce integer solutions. The general framework of branch-and-price is described in Figure 37. Here, $\tilde{c}_{ij}$ is the reduced cost of a column. The subproblem referred to in the-branch-and-price context is slightly different from the one discussed earlier, since each time a branching is performed new constraints are added to the problem and the branching strategy used in this chapter is constraint branching that is described in Chapter 4.

Exact branch-and-price requires pricing of every node of the tree. Since this is time consuming, we decide to price the columns in the pool only when the LP solution increases, between one node of the tree to another, more than a predefined percentage, for example, 0.2%. The pricing procedure resembles the one described earlier, the only difference is that we have to check the ‘feasibility’ of the columns with respect to the branching constraints as well. An earlier feasible column might become infeasible during the branch-and-price tree, since new constraints are being added.

When a certain number of columns has been added to the restricted master problem during branch-and-price we interrupt the procedure. Then, branch-and-bound is applied to the resulting restricted master problem and the integer solution found there is the one registered in the table of results.

5.6 Tests and results

5.6.1 Case study

The data of the case studied comes from a major Swedish company, Sydved, which co-ordinates the transportation in the southern part of Sweden. The south of Sweden is typically characterised by a large number of small supply points that are scattered around the area. The industries are evenly spread in the area. This is in contrast to the northern part of Sweden which is characterized by large supplies and industries located along the coastline.

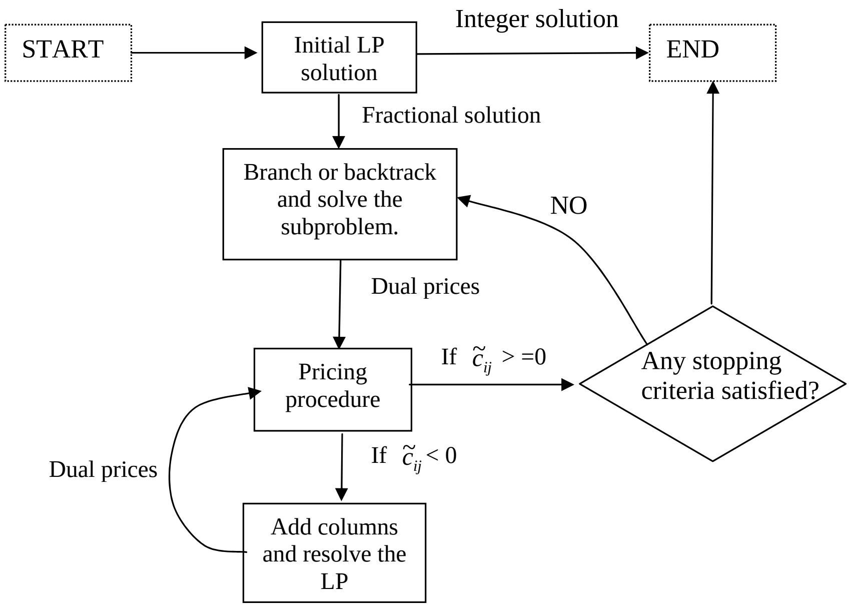


Figure 37: The general framework of the branch and price procedure

In this case the number of supply points is 187 whereas the number of demand points is 15. There are 4 assortments and the total quantity demanded is 3239 tonnes. There are 28 trucks are available for the transportation. The working days are divided into two shifts, each consisting of 8 hours of work. The first shift starts at 5 a.m. in the morning and stops at about 1 p.m., after that a new driver takes over and the second shift starts ten minutes later to last until 9 p.m. The start and end nodes of the shifts are the home base, which means that the drivers meet at this location to exchange shifts.

The algorithms are implemented in the C language and Cplex is used to solve the linear programs and the final restricted master problem created by using column generation and branch-and -price. The computer used is a PC with Pentium 4, 1.6 GHz processor.

5.6.2 Numerical results

The results are presented so that we study one part of the algorithm at a time. First we study the impact of using the constrained shortest path algorithm for solving the LP relaxation. This is compared to the case where we use only a pool of columns, as in previous studies (Chapter 4). We then study the impact of using the branch-and-bound algorithm for solving the final restricted master problem.

Solving the LP relaxation of the problem

In the study of this problem in Chapter 4 we used a pool of columns, and the columns generated were not affected by the dual values. In Table 5, we use the heuristic proposed in Chapter 4 to enumerate a priori a set of routes. Here, we vary the number of enumerated routes according to the heuristic and the LP values are registered. We may note that the solution is highly dependent on the number of columns in the pool. This in particular if we have less than one million columns.

# of routes in the pool	LP value	CPU time in sec.
89,720	463,190	11
571,887	350,177	78
1,080,112	341,762	140
2,836,273	334,981	351
4,167,880	334,981	574

Table 5: The LP value obtained by using the set of columns generated a priori.

5.6.3 Using the *K*-shortest paths column generator

The initial LP solution corresponding to solving the LRRMP0, is 3,257,822, which is the sum of all penalties in the cost function. At first we tested using the dual values obtained from the solution of the initial problem to build the first k-shortest path network. The number of nodes in this network is 46,810 and the number of arcs is approximately 4.1 million. The value of *K* used is 300,000. The time it takes to solve one *K*-shortest paths problem for this value of *K* is around 25 seconds. The number of feasible routes created varies from one iteration to another, and in the worst case this number is zero. In such cases the algorithm continues with the next truck. No initial set of columns is used in this test.

# of iterations	LP value	CPU time in sec.
1	1,599,560	1,356
2	417,065	2,600
3	343,718	3,956
4	335,816	5,414
5	332,583	6,783
6	331,877	9,127
7	331,512	10,452
8	331,485	11,800

Table 6: The LP value obtained after each iteration where the *K*-shortest paths algorithm is used to solve the subproblem.

The results of this test are shown in Table 6. Each *K*-shortest paths iteration corresponds to solving a *K*-shortest paths problem for each truck and resolving the restricted LP in between. We added at most 5 columns to the restricted master problem after each *K*-shortest paths problem. Note the characteristic behaviour of column generation: the LP value drops quickly in the beginning, and after a number of iterations only a small reduction in this value is achieved. In Table 7, we used an initial pool containing 571,887 (see Table 5) and the LP value obtained from this pool as a start solution, then *K*-shortest paths was applied for 10 iterations.

# of iterations	LP value
1	333,920
2	329,768
3	329,059
4	328,544
5	328,369
6	328,369
7	328,365
8	328,365
9	328,365
10	328,365

Table 7: The LP value after each iteration where k-shortest paths is used to solve the constrained shortest path problem.

Some of the feasible routes found by the *K*-shortest paths algorithm are iteratively stored in the pool. When the number of columns in the pool becomes too large we get memory problems since all columns must be kept stored during the whole procedure. Using the heuristic mentioned above to produce an initial LP value saves us important computational time as long as the number of columns in the pool is moderate. This test also shows the slow final convergence of column generation. Using the bounding discussed in Section 5.5.3, we find that the lower bound is 2% away from the LP value registered in iteration 10.

Solving the IP problem

Table 8 shows the integer values obtained by using the routes generated heuristically, according to the method in Chapter 4. Thereafter, in Table 9, we start by solving the LP relaxation within the set of columns in a number of pools. Then, only one *K*-shortest path iteration is applied during the LP phase and, some columns are added to the restricted master problem. Note that no columns are added to the pool. We also note that the LP value is much improved as compared to Table 5, and the reason for this is the possibility to use the dual information for generating new columns.

# of routes in the pool	Integer value	CPU time in sec	# of columns added in B&P
89,720	640,244	733	1,456
571,887	395,741	463	1,673
1,080,112	371,945	2,606	1,656
2,836,273	355,087	4,821	1,656

Table 8: The integer value obtained by using the method in Chapter 4.

In these tests the branch-and-price procedure is interrupted after 10,000 tree nodes. The restricted problem that is created during the LP and the branch-and-price phases is afterwards solved to optimality by the Cplex MIP solver.

We can further improve the quality of the integer solutions found by adding columns to the pool obtained from solving the constrained shortest path problem. These columns can then be used during branch-and-price. In Table 10, we show the results obtained by adding up to 500 columns to the initial pool from each *K*-shortest path problem. As we have 28 trucks we add at most 14,000 columns. By using these additional columns we can find much better solutions.

# of routes in the pool	LP value	Integer value	CPU time in sec	# of columns added in B&P
89,720	340,548	368,635	1,996	1,102
571,887	333,920	361,147	2,140	1,358
1,080,112	328,971	352,327	6,143	2,251
2,836,273	328,729	353,383	3,196	1,661

Table 9: The integer value obtained by using the initial pool of columns and one *K*-shortest paths iteration.

# of routes in the pool	# of routes in the pool after K-shortest paths	Integer value	CPU time in sec	# of columns added in B&P
89,720	101,345	368,004	4,254	1,054
571,887	582,115	356,299	8,841	1,458
1,080,112	1,090,860	350,023	3,533	1,911

Table 10: The integer value obtained by using the enlarged pool of columns during branch and price.

5.7 Conclusion

In this chapter we have described a near-exact column generation algorithm for the log truck scheduling problem. Our computational experiments show that exact column generation can be used to solve the LTSP in reasonable time. Exact column generation improves the solution obtained from the heuristic cluster-first-route second column generation approach and provides lower bounds that are a measure of the solution quality. Using a *K*-shortest paths algorithm to solve the constrained shortest path problem is in our case very efficient, since it returns a large number of feasible routes that can be used in the integer phase.

6 Column generation by repeated clustering and route enumeration

6.1 Introduction

The method proposed in this chapter combines ideas from the methods presented in Chapters 4 and 5. Similarly to the method in Chapter 5, we make use of the information provided by the dual variables in the route generation (phase), and as in Chapter 4 we use the transportation problem to build clusters which are used in the route generation. In this way we avoid the time-consuming constrained shortest path problem at the same time as we take advantage of the dual information.

Similarly to all scheduling problems, any feasible solution to the LTSP takes into account three aspects, namely clustering, routing and scheduling (Desrosiers et al. 1995). Algorithms for solving scheduling problems differ in the order they consider these three aspects, which can be handled either simultaneously or one at a time.

These three aspects apply to the LTSP as follows:

1. Clustering consists of deciding which supply points should deliver to each customer, and which vehicle should service each location.
2. Routing consists of finding the order in which several locations are to be visited within each route.
3. Scheduling is to decide the time when the vehicle should start and the time schedule for the whole route.

The constrained shortest path problem used in Chapter 5 takes into consideration these three aspects simultaneously. This problem is, however, very time-consuming to solve. For this reason we choose to separate the three decisions of the LTSP, and to deal with clustering first and routing and scheduling second. We achieve this separation by replacing the constrained shortest path problem by a two-phase procedure: the first phase consists of solving a transportation problem with reduced costs, and the second phase involves generating feasible routes. The transportation solution of the first phase is used to build the clusters within which routes are enumerated. In this way the clustering aspect is considered prior to routing and scheduling, which are thereafter regarded simultaneously when enumerating the routes. The two-phase procedure is repeated, each time with new dual values. Figure 38 illustrates the difference between the method in Chapter 5 and the repeated clustering and route generation method.

Like the method in Chapter 5, the repeated clustering and route enumeration method takes advantage of the information provided by the dual variables in order to generate new routes. This is accomplished by using the values of the dual variable to define the reduced costs in the transportation problem. The basic idea is to repeatedly use the current values of the dual variables in order to reduce the costs in the transportation problem, and thereby generate new clusters and new routes that can be added to the master problem.

In this chapter we describe the repeated clustering and route enumeration method, give the reduced transportation problem used, and relate it to the LTSP. We also apply the repeated clustering and route enumeration method to a case from a Swedish forestry company and present numerical results.

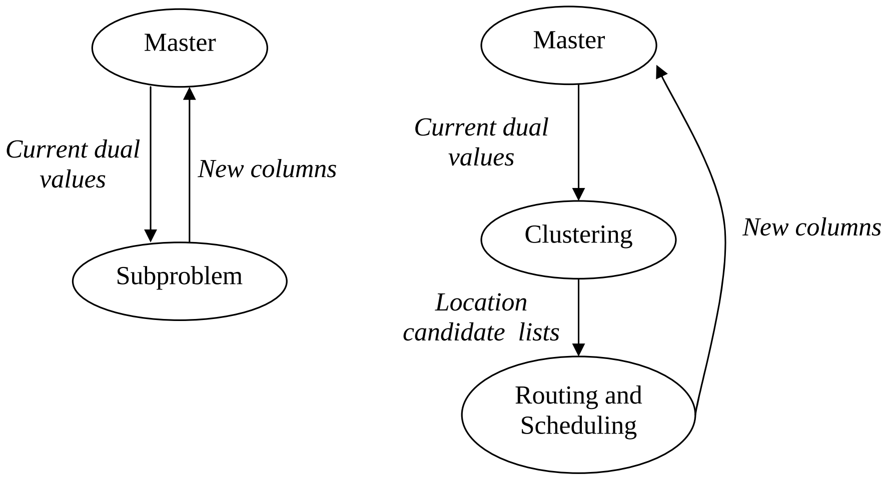


Figure 38: Comparison between the methods in Chapter 5 and in Chapter 6

6.2 General framework

The overall method, in which the repeated clustering and route enumeration is used, involves two stages: an LP stage where the linear relaxation of the master problem is solved approximately and an IP stage where branch-and-price is applied. The LP stage starts by solving the linear relaxation of the initial restricted master problem (LRRMP0), as in the methods of both Chapters 4 and 5. The optimal values of the dual variables for the supply and demand constraints in this initial problem are used to define the reduced arc costs of the first transportation problem. The LP stage evolves by solving this transportation problem for finding clusters that are used to generate feasible routes. These routes are generated by an enumeration process applied within the clusters, and all feasible routes generated are added to a pool. Then, the columns corresponding to the feasible routes are priced and the column with the most negative reduced cost is added to the LRRMP, which is resolved. This pricing procedure is repeated until no more columns with negative reduced cost are found in the pool, after which a new clustering and route generation is performed, and so on.

The general framework for solving the LP relaxation of [GM] is the following:

1. Initialisation.
2. Design an initial set of feasible routes.
3. Solve the LP relaxation of the restricted master problem within the set of routes created in step 2.
4. Repeat for a predefined number of iterations:

5. For each truck *k*, do:

    a. Construct the reduced transportation problem, using the dual variables obtained in step 5.d (or step 3) to compute the arc costs.
    b. Solve the transportation problem and construct candidate lists.
    c. Generate routes for truck *k* and store the feasible routes in the pool.
    d. Optimise the LRRMP over the columns currently in the pool, by repeatedly pricing the columns in the pool and reoptimising the LRRMP.

The cyclic course of action that takes place in the LP stage is illustrated in Figure 39. We refer to it as *sequential* column generation, and distinguish it from classical column generation in which the subproblems are solved in a parallel manner, that is, using the same dual values for all subproblems. In this figure the pool is ignored.

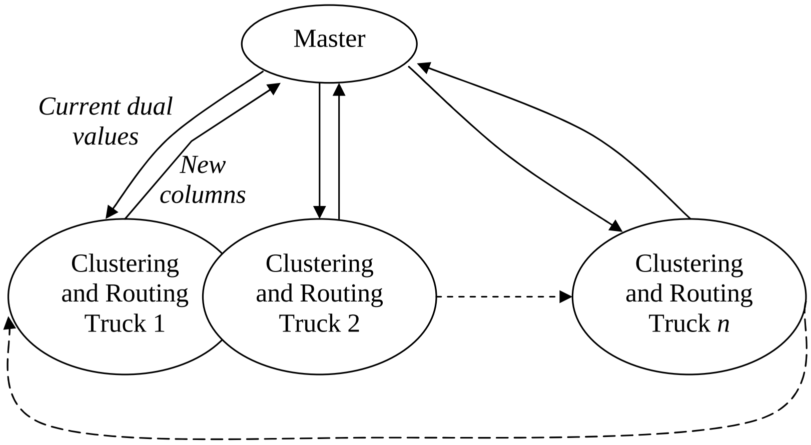

Master
Current dual values
New columns
Clustering and Routing Truck 1
Clustering and Routing Truck 2
Clustering and Routing Truck n

Figure 39: The cyclic procedure in the LP stage

As this general framework shows, the reduced transportation problem, clustering and route generation is truck specific. This is necessary since all trucks can sometimes not visit all supply points due to local agreements, truck type, or, driver preferences.

The goal of the IP stage, where branch-and-price is applied, is of course to find integer solutions to the LTSP. As in the two methods proposed earlier, constraint branching is applied. At some of the nodes of the branch-and-bound tree we generate new feasible routes; the current dual values are used to reduce the arc costs of the transportation problem and the whole clustering and route enumeration procedure of the LP stage is repeated. In other words, we solve the transportation problem, generate new clusters and then routes that are added to the pool. In contrast to the LP stage, the pricing performed during the IP phase includes checking the feasibility of the columns with respect to all additional constraints added to the problem through the branchings made in order to reach the current node. The columns that satisfy the branching constraints and that have negative reduced costs are candidates to enter the LRRMP.

The enumeration of the routes is made by firstly building candidate lists that define the locations that are *reachable* from each home base, supply point and demand point, respectively. The candidate lists are based on the clusters generated by solving several transportation problems, and, for example, a demand point is considered to be reachable from a supply point if flow is sent between these two locations in the solution to the reduced transportation problem . The purpose of these lists is of course to limit the number of possible continuations of a partial route, which in its turn limits the number of routes enumerated. Our goal in this method is to use dual information to generate high quality clusters, that lead to high quality candidate lists and enumerated routes.

The enumeration procedure always starts at the home base of the truck, visits all supply points reachable according the candidate list, and continues by visiting more supply points, whenever the truck is not fully loaded, or demand points, whenever the truck is full. This is performed while keeping track of the time windows, the breaks during the day, the load of the truck, and the hour when exchanges of drivers must be made. The enumeration procedure also makes sure that the mixture of assortments picked up matches the demand required. We will in this chapter describe the various candidate lists in detail and show how these are derived from the transportation problem.

Since the repeated clustering and route enumeration method depends on the information given by the dual variables for the linear relaxation of the restricted master problem, LRRMP, we start by examining this problem and its dual in more detail.

6.3 The linear relaxation of the master problem and its dual

The LRRMP is stated mathematically as:

$$
\begin{array}{lll}
\textit{min} & \sum_{i=1}^{n}\sum_{j=1}^{m_i} c_{ij}\, x_{ij} + \sum_{k=1}^{n_s} p_k y_k + \sum_{p=1}^{n_d} g_p u_p + \sum_{p=1}^{n_d} g'_p u'_p & \\
\textit{subject to} & \sum_{j=1}^{m_i} x_{ij} = 1 & i = 1,2,\ldots,n \\
 & \sum_{i=1}^{n}\sum_{j=1}^{m_i} a_{ijk} . x_{ij} + y_k = s_k & k = 1,2,\ldots,n_s \\
 & \sum_{i=1}^{n}\sum_{j=1}^{m_i} b_{ijp} x_{ij} + u_p - u'_p = d_p & p = 1,2,\ldots,n_d \\
 & x_{ij} \geq 0 & \forall i, j \\
 & y_k, u_p, u'_p \geq 0 & \forall k, p
\end{array}
$$

Let v_i, w_k, and z_p denote the dual variables associated with the three sets of constraints respectively. The dual problem of the LRRMP then becomes:

$$\begin{array}{lllll}
\textit{max} & \sum_{i=1}^{n} v_i + \sum_{k=1}^{n_s} s_k w_k + \sum_{p=1}^{n_d} d_p z_p & & & \\
\text{subject to} & v_i + \sum_{k=1}^{n_s} a_{ijk} w_k + \sum_{p=1}^{n_d} b_{ijp} z_p & \le c_{ij} & \forall i, j \\
& w_k & \le p_k & \forall k \\
& z_p & \le g_p & \forall p \\
& z_p & \ge -g'_p & \forall p
\end{array}$$

To discuss the impact of the dual variables we make use of the following complementarity conditions.

$$\begin{aligned}
& y_k (w_k - p_k) = 0 \\
& u_p (z_p - g_p) = 0 \\
& u'_p (z_p + g'_p) = 0
\end{aligned}$$

These conditions define the values that the dual variables associated with the supply and demand constraints might obtain. Note, in particular, the following:

- $w_k = p_k$ holds if $y_k > 0$, that is when the supply is not emptied
- $z_p = g_p$ holds if $u_p > 0$, that is when the demand is not fulfilled
- $z_p = -g'_p$ holds if $u'_p > 0$, that is when the demand is exceeded

We observe that the dual variables clearly indicate when a demand is not satisfied or, on the other hand, when it is exceeded. They also indicate if there are any logs left at the supply points.

Proper column generation, as solving the constrained shortest path problem in the method of Chapter 5, automatically takes advantage of the information hidden in the values of the dual variables, like the information described above. The purpose of the repeated clustering and route enumeration method is to make use of the dual information without applying the expensive column generation approach of Chapter 5. This is done by defining a specific reduced transportation problem.

6.4 The transportation problem with reduced costs

The generic transportation problem used in this chapter is defined as follows.

- The supply nodes correspond to the supply points of the LTSP.
- The demand nodes correspond to the demand points of the LTSP.
- The arcs of the network connect all supply nodes to all demand nodes.

In order to define the arc costs of the transportation problem we need to introduce the *compatibility* aspect. A pair of supply and demand nodes is referred to as *compatible* when the following criteria are satisfied.

- The assortment at the supply point matches the product at the demand point.
- The truck considered in the transportation problem is allowed to visit the supply point.
- The supply point is allowed to deliver to the demand point.

If one of these criteria is not satisfied then a supply-demand pair is *incompatible*. Whenever a supply-demand pair is incompatible, the cost of the corresponding arc is set to a large number. As mentioned earlier, the repeated clustering and route enumeration method considers one transportation problem for each vehicle, and these transportation problems differ from one truck to another since each truck may visit only a subset of all supply points.

Apart from the compatibility issue, the arc costs in the reduced transportation problem depend on the transport prices per km in the LTSP. These prices are usually divided into four categories:

- The price per km for driving an empty truck during normal working hours.
- The price per km for driving a full truck during normal working hours.
- The price per km for driving an empty truck after normal working hours.
- The price per km for driving a full truck after normal working hours.

We define $P_{\min}$ as the *minimum price per km and tonne* for transporting logs by truck, or, in other words, the cheapest possible way of transporting one tonne of logs one kilometre. Clearly, $P_{\min}$ is the price per km for driving a full truck during normal hours divided by the capacity of the truck. Letting d_{kp} denote the shortest distance from supply point *k* to demand point *p*, the arc costs of the reduced transportation problem then are the following:

- $P_{\min} d_{kp} - w_k - z_p$ for all arcs connecting *compatible* pairs of supply and demand nodes.
- M (a large positive number) for all arcs connecting *incompatible* pairs of supply and demand nodes.

A rationale behind incorporating the dual values in the arc costs is that we thereby prioritise transport of logs to demand points with unfulfilled demands ($z_p = g_p$), or from supply points that are not emptied ($w_k = p_k$), and penalise transport to demand points with excess of logs ($z_p = -g'_p$). The use of the reduced transportation for clustering can also be motivated by its relation to the LTSP, as described below.

6.4.1 Relation between the LTSP and the reduced transportation problem

Here we consider the case where all $p_k = 0$ and assume that the transportation problem always has a feasible solution that does not use arcs belonging to incompatible pairs of supply and demand points. We argue that the reduced transportation problem defined as above is then a relaxation of the LTSP. (In practice, the values of p_k and g'_p are typically small, compared to the value of g_p, or even zero.) These are the arguments for our statement:

- Every feasible solution to the LTSP is associated with a feasible solution to the reduced transportation problem.
- The objective function of the reduced transportation problem underestimates the objective function of the LTSP.

It is easy realised that every feasible solution of the LTSP also satisfy the transportation constraints, that is, that the total supply picked up at a certain supply point can not exceed the supply available, and that the demand required by the customer must be satisfied. Figure 40 shows a feasible route and its corresponding transportation problem solution.

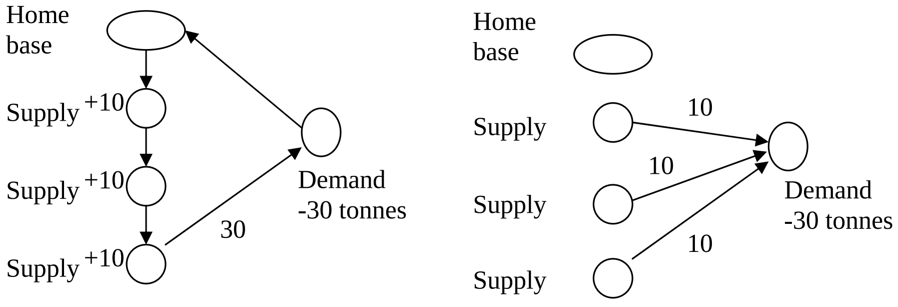


Figure 40: An LTSP solution and its corresponding flow solution

In order to establish our second argument, let us first consider the transportation problem with arc costs $P_{\min} d_{kp}$. The following observations can be made:

- Each quantity of logs is transported in the shortest possible way, since the arc costs are defined by shortest distances
- Each quantity is transported according to the cheapest price per km and tonne (we assumed that no arcs with cost M are needed).
- The solution of the transportation problem does not include driving from the home base in the beginning of the route and back to the home base in the end of the route.
- The solution of the transportation problem does not include driving from demand points to supply points.

We illustrate the first observation in Figure 41, where distances have been introduced in the example in Figure 40. Note that a quantity of logs at Supply 1 will traverse a total distance of 11 km in the LTSP solution, while the traversed distance in the transportation problem is only 9 km. The second observation follows directly from the definition of P_{min}. The third and the fourth are obvious.

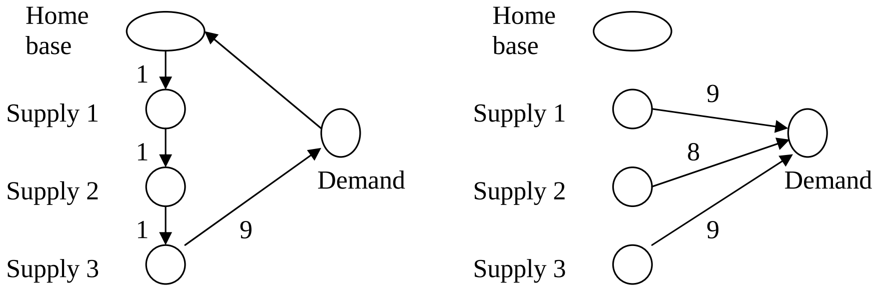


Figure 41: An LTSP solution and its corresponding flow solution

We conclude that the transportation problem with arc costs $P_{min}\ d_{kp}$ is a relaxation of the LTSP, which means that the optimal value of the former is a lower bound to that of the latter. We now turn to the reduced transportation problem. It is then easily realised that it can be obtained from the transportation problem with arc costs $P_{min}\ d_{kp}$ by subtracting multiples w_k and z_p of the supply and demand constraints, respectively, from the objective function, which then becomes

$$\sum_{(k,p\in A)} P_{\min} d_{kp} x_{kp} - \sum_{k=1}^{m} w_k \left(\sum_{p=1}^{n} x_{kp} - s_k \right) - \sum_{p=1}^{n} z_p \left(\sum_{k=1}^{m} x_{kp} - d_p \right)$$

For any feasible solution to the transportation problem, the value of this function is less or equal to the value of that of the original transportation problem, since the second term is non-positive (since $w_k \le p_k = 0$) and the third is zero. This function can be rewritten as

$$\sum_{k=1}^{m} \sum_{p=1}^{n} \left(P_{\min} d_{kp} - w_k - z_p \right) x_{kp} + \sum_{k=1}^{m} w_k s_k + \sum_{p=1}^{n} z_p d_p$$

and by dropping the constant term the reduced transportation problem, with arc costs $P_{min}\ d_{kp} - w_k - z_k$, is obtained. The reduced transportation problem is hence a relaxation of original transportation problem, and therefore also of the LTSP.

The four observations stated above can be considered as drawbacks of the transportation problem, since they lead to an underestimation of the LTSP. In what follows we introduce variants of the basic transportation problem where these drawbacks are dealt with.

6.4.2 Extensions of the reduced transportation problem

Transportation problem with backhaul variables

In the extension with *backhaul variables* that enforce the vehicle to return back to a supply point are added. The introduction of backhaul variables is therefore a means for making a transportation solution resemble more closely a solution to the LTSP.

This extension can be stated mathematically as follows.

$$
\begin{array}{llcll}
\textit{min} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} w_{ij} + \sum_{j=1}^{n}\sum_{i=1}^{m} b_{ji} v_{ji} & & & \\
\textit{s.t.} & \sum_{j=1}^{n} w_{ij} & \leq & s_i & \text{for } i = 1, 2, \ldots, m \\
 & \sum_{i=1}^{m} w_{ij} & = & d_j & \text{for } j = 1, 2, \ldots, n \\
 & \sum_{j=1}^{n} (w_{ij} - v_{ji}) & = & 0 & \text{for } i = 1, 2, \ldots, m \\
 & \sum_{i=1}^{m} v_{ji} & = & d_j & \text{for } j = 1, 2, \ldots, n \\
 & w_{ij}, v_{ji} & \geq & 0 & \text{for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n
\end{array}
$$

The first two sets of constraints are the traditional transportation constraints. The third set of constraints guarantees the flow balance at the supply points. The fourth set of constraints ensures that all flow transported by the backhaul variables from any demand point equals the total demand.

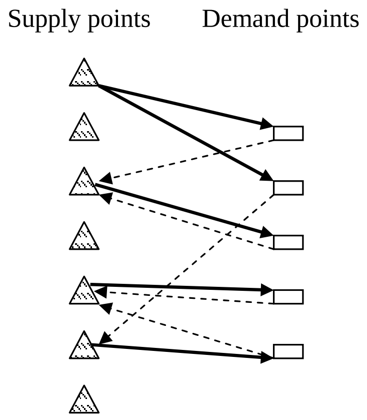


Figure 42: A solution of the transportation problem with backhaul

The objective function is to minimise the cost of transporting flow from the supply to the demand points, *and* the cost of transporting the same flow back to the supply points. This means that the solution to this extension of the transportation problem decides the flow from supplies to demands by taking into consideration the fact that each truck is driving back to some supply point after the delivery, instead of staying at the demand point. The example in Figure 42 shows a solution to the transportation problem with backhaul.

Transportation problem with home distances added to the arc costs

In each iteration, the repeated clustering and route enumeration method solves a new transportation problem for each truck. Each transportation problem is defined for a specific truck and the compatibility criteria define the arc costs of the transportation problem. One of the drawbacks mentioned earlier is that the transportation problem does not consider the home bases of the drivers. In order to deal with this drawback we define a transportation problem where the arc costs are as illustrated in Figure 43.

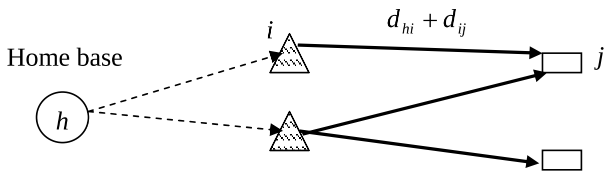


Figure 43: Transportation problem where home distances are added to the arc costs

The arcs shown in the example correspond to compatible pairs of supply and demand points. The distances d_{hi} from the home base *h* to the supply point *i* are added to the original distances in the arc costs of the transportation problem. If the truck is not allowed to visit a certain supply point then d_{hi} is set to a large positive number M, so that all the arcs originating from that supply will have the cost M+d_{ij}. This extension of the transportation problem is used to design the candidate list of the supply points that can be visited from a certain home base only in the beginning of a route.

Transportation problem where small supplies are penalised

The first and second drawbacks of the transportation problem is that it underestimates the total distance travelled and the cost of travelling, since many quantities of logs are not transported the shortest distance from a supply point to a demand point, or on a fully loaded truck. An attempt to deal with these two issues is made by penalising all *small* supplies. A small supply is a quantity less than the truck capacity. The difficulty with small supplies is that they usually lead to split pickups, in the sense that the truck must visit several pickup points to load. If the truck instead becomes loaded after only one visit to a supply point, the distance travelled in the solution to the transportation problem is a better estimate of the true distance travelled in a feasible route of the LTSP. By adding a penalty cost to all arcs in the transportation problem connecting small supplies to demand points, we prioritise solutions where large supply points are chosen, as we should since large supplies in fact cost less to transport per kilometre and tonne, than small supplies do. Here it is important to note that most transportation problems derived from the LTSP are not balanced; the total supply usually exceeds the demand. This means that we have a degree of freedom when choosing which supply points should be used to satisfy the demands. We have experimented by associating differently large penalty costs to different supplies, according to how large the quantity available is.

6.5 Enumerating feasible routes

As mentioned earlier, the enumeration of routes depends on a set of candidate lists that define the nodes reachable from any given location (home base, supply point or demand point). We use four types of candidate lists:

- Home-Base-Supply-List

- Supply-Demand-List
- Demand-Supply-List
- Demand-All-Supplies-List

The *Home-Base-Supply-List* defines all supply points that are reachable from a certain home base and is constructed according to the solution to the extension of the transportation problem where the distances between the home base and the supply points are added to the arc costs. Here, the issue of compatibility is implicitly taken into account in the transportation problem. This means that all supply points in this list are allowed to be visited by the truck.

The second list, the *Supply-Demand-List*, is constructed according to the forward flows in the transportation problem with backhaul, and it includes all demand points that are reachable from each supply point. Since the transportation problem is not balanced, some of the supply points might not be included in the transportation solution. We choose to construct the Supply-Demand-List for such supplies by allowing the truck to continue to the two closest demand points that are compatible with the supply.

The third list, the *Demand-Supply-List*, determines all supply points that are reachable from each demand point. This list is generated according to the backhaul flows in the transportation problem with backhaul.

Finally, the fourth list, the *Demand-All-Supplies-List,* is determined according to the forward flow variables in the solution to the transportation problem with backhaul. It includes all supply points that deliver to the same demand point. In the case when a truck visits a supply point with a quantity less than the truck capacity, the truck needs to visit another supply point for another pickup operation. This issue is however complicated, due to the various assortments that are allowed to be mixed on the truck.

The purpose of the Demand-All-Supplies-List is to help the route generator to find a candidate supply point to visit whenever split pickups are needed.We explain this by the example below. The product required at demand point 3 is A, and the assortments matching this product are 2, 5 and 8. A truck delivering to demand point 3 is therefore allowed to freely mix the assortments 2, 5 and 8. Let the solution to the transportation problem be according to Figure 44. If the truck starts by visiting supply point 7 and picking up 25 tonnes, then it must proceed (assuming that the truck capacity is 40 tonnes) by visiting a second supply point, with a matching assortment, to continue loading. The Demand-All-Supply-List is then used to find a supply point that is compatible with the demand point which supply point 7 is supposed to deliver to, according to the flow solution.

The process of enumerating feasible routes always starts at the home base of a truck. The truck then visits the first supply point according to the Home-Base-Supply-List, load the truck as much the truck capacity and the quantity available at the supply point allow. If the truck is fully loaded, it continues to a demand point, according to the Supply-Demand-List, otherwise it drives to another supply point to continue loading according to the Demand-All-Supply-List. After the truck has unloaded at a demand point it continues from there by visiting a supply point from the Demand-Supply-List, and the process of loading and unloading is repeated. Each time the truck reaches a new location, the time window constraints are checked. In this way, the construction of the feasible route proceeds until the working day ends and the truck returns to the home base.

The enumeration procedure loops over all trucks and all supply points in the candidate lists. It constructs each route, trip by trip until the end of the day is reached. The number of feasible routes enumerated can be controlled by varying the length of the candidate lists (in particular the length of the Home-Base-Supply-List).

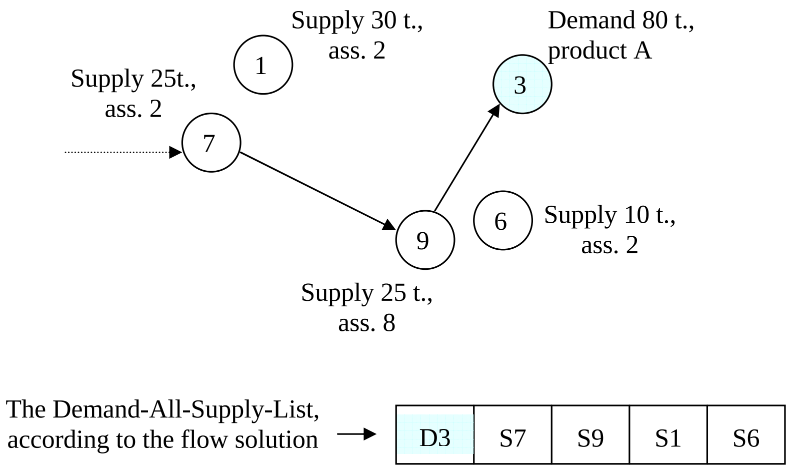


Figure 44: An illustration of the use of the Demand-All-Supply-List

6.6 Numerical results

Description of the case

The case studied in this chapter is from a large forestry company, Holmen Skog, and this study is done in cooperation with the Norrköping region. The Norrköping region manages 80,000 hectares of Holmen's forests and the company also deals with almost 4,000 private forest owners in that same region. Besides managing the forests, Holmen Skog also supplies its customers with raw material from its own forests or from the private forest owners.

The trucks used for the transportation of logs from forests to customers belong to independent drivers or to small transportation companies. Holmen Skog arranges the contracts between sellers of logs and buyers, and by doing that they form so called *transport orders* in which the specifications of the orders are described. A transport order contains the place for picking up the logs, the assortment of the logs, the quantity in tonnes, the place where the logs must be delivered, and the date for the delivery. This means that in this case study, the origin and destination of the logs are more or less specified in advance.

The drivers working for Holmen Skog usually load their truck on their way back to their home bases, which means that many trucks start daily routes by immediately visiting a demand point for unloading. In our case study, the trucks are assumed to be empty at the beginning of the day, but the enumeration algorithm could easily be modified to take into account this additional restriction. The case studied in this chapter involves 12 trucks of capacity 37 tonnes, 13 demand points, 386 supply points, 17 groups of assortments, or products, 17 assortments, and 77 full loads that need to be transported from supply points to customers. The trucks are used in two or three shifts and 24 hours a day. We allow in our algorithm up to three stops per day, where a stop is defined by a certain location and a certain time interval within which the truck must be present at that specific location.

Description of the parameters

The parameters used in the tests below are the following:

1. *Number of iterations in the LP stage*. Tests have shown that the LP solution is improved, with respect to objective value and satisfaction of customer demand, mainly during the early iterations in the LP stage.
2. *Length of the Home-Base-List*, that is, the number of supply points reached from the home base. This number controls the number of feasible routes generated for each truck. This parameter is referred to as *"# supplies in iter."* in the tables of numerical results.
3. *Maximum number of routes enumerated for each vehicle*. Tests have shown that the number of feasible routes generated might differ widely from one truck to another. In order to keep this difference small we introduced a parameter in the enumeration process that interrupts the route enumeration if the number of routes exceeds a certain number. This parameter is referred to as "*Max routes*" in the tables of numerical results.
4. *Alpha, beta, and gamma*. These are three parameters used to penalise the supply points with quantities less than the truck capacity. Alpha penalises supplies less than 10 tonnes, beta penalises supplies between 10 and 20 tonnes, and, finally gamma is used for supplies between 20 and 37 tonnes, which is the capacity of the truck.
5. *Number of columns added while pricing during B&P*. This parameter controls the maximal number of columns with negative reduced costs that can be added to the restricted master during branch-and-bound. Each pricing procedure computes the reduced cost of all routes in the pool but, the number of routes with negative reduced cost may not exceed this parameter. It is referred to as *"# col. B&P"*.

Numerical results

In Table 11 we compare sequential column generation, where the transportation problems are solved sequentially using the current dual prices (Ssub), see Figure 39, with traditional column generation in which the transportation problems are solved in a parallel manner (Psub), as illustrated in Figure 45. The parameter values in Table 11 are as follows: alpha, beta, and gamma are equal to zero, the length of the Home-Base-List is 10, and finally, branch-and-price is not used. Recall that the mathematical model allows customer shortage, and this shortage is expressed here as the total number of full loads not delivered to the customers. The LP values and the integer values, obtained from branch-and-bound, are compared. These values do not include the penalties for deficit and surplus at demand and supply points, respectively. Hence, the values correspond to the real costs of the routes in the solution.

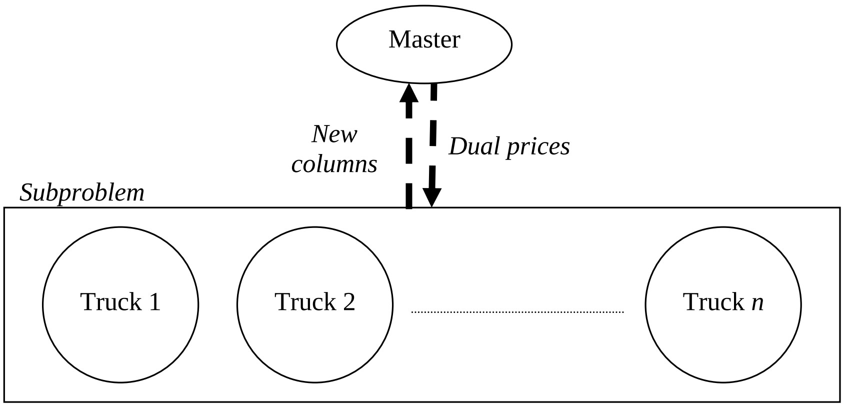


Figure 45: The traditional column generation process

TP type	Max routes	#LP iter.	LP solution	LP shortage	B&B: IP solution	Total # routes	Customers shortage
Psub	2,000	2	39,071.72	12	33,938	315,427	14
Ssub	2,000	2	41,013.24	7	38,473	291,500	8
Psub	2,000	3	39,117.86	8	36,002	474,039	9
Ssub	2,000	3	39,271.57	6	37,734	449,440	8
Psub	2,000	4	39,587.34	6	37,906	619,516	7
Ssub	2,000	4	40,857.94	6	36,797	598,059	8

Table 11: Comparison between traditional and sequential column generation

As the results in Table 11 show, sequential column generation gives better solutions when the number of iterations is kept low, since the dual prices used are always updated and thus closer to the optimal dual prices. However, after only a few iterations, the LP value is stabilised together with the dual prices, a fact that is shown by the similar LP results for both the traditional and sequential column generation.

#LP itera.	Max routes	LP solution	LP short	B&B: IP solution	Total # routes	# col in FRMP	Cust. Short.	CPU min
2	2,000	40,121.3	12	38,748	143,104	555	14	2:37
2	5,000	38,674.8	7	34,519	320,622	561	9	4:48
3	2,000	41,459.2	9	39,298	215,606	619	11	3:53
3	5,000	39,905.7	6	35,570	477,734	588	8	6:54
4	2,000	38,709.7	8	35,737	284,308	634	10	4:42
4	5,000	37,755.8	6	35,570	626,681	593	8	8:38

Table 12: Integer solutions obtained from branch-and-bound

The purpose of Tables 12 and 13 is to compare the integer solutions obtained from branch-and-bound alone with the integer solutions obtained from branch-and-price. Here we apply only sequential column generation and we choose the length of the Home-Base-List to be 5. We register the number of columns in the final restricted master problem (FRMP) obtained after the addition of columns during the LP stage (Table 12), and the LP stage and branch-and-price (Table 13), respectively. The values of the parameters alpha, beta, and gamma are set to zero.

#LP itera.	Max routes	#col. B&P	LP solution	B&P: IP solution	Total # routes	# col in FRMP	Cust. Short.	CPU time min
2	2,000	5	40,121.3	37,965	263,899	830	13	5:56
2	5,000	5	38,674.8	34,260	628,989	971	8	15:48
3	2,000	5	41,459.2	36,456	334,665	899	7	8:10
3	5,000	5	39,905.7	38,186	789,442	728	7	9:23
4	2,000	5	38,709.7	37,406	402,534	889	8	9:42
4	5,000	5	37,755.8	35,369	943,202	938	7	19:12
2	2,000	20	40,121.3	37,965	263,899	1,215	13	5:42
2	5,000	20	38,674.8	34,260	628,989	1,541	8	14:12
3	2,000	20	41,459.2	36,445	334,665	1,259	7	7:26
3	5,000	20	39,905.7	35,570	789,442	988	7	10:46
4	2,000	20	38,709.7	38,496	402,534	1,214	8	8:27
4	5,000	20	37,755.8	35,717	943,202	1,493	7	17:13

Table 13: Integer solutions obtained from branch-and-price

The results in Tables 12 and 13 show, as expected, that the integer solutions obtained after branch-and-price cover the total demand better than the solutions obtained from branch-and-bound alone. The lowest customer shortage in these results is 7 loads, which is approximately 10% of the total demand.

Max rou,tes	Alpha	Beta	gamma	B&P: IP solution	Total # routes	# col in FRMP	Cust. Short.	CPU time min
3,000	0	0	0	37,558	472,889	823	7	10:04
5,000	0	0	0	38,186	789,442	728	7	11:53
7,000	0	0	0	36,225	1,117,642	889	7	22:57
10,000	0	0	0	35,273	1,510,536	896	7	28:23
5,000	30	30	30	34,778	700,123	1,275	4	28:33
5,000	50	50	50	35,164	680,024	1,132	3	24:51
5,000	100	100	100	34,889	634,965	901	4	15:15
3,000	100	50	30	35,805	389,226	853	4	11:07
5,000	100	50	30	34,615	598,779	1,223	2	22:03
7,000	100	50	30	36,299	801,351	1,008	4	20:21

Table 14: Effect of alpha, beta, and gamma

In the next experiment we study the effect of the values of the parameters alpha, beta and gamma. The idea is to choose values that are similar to the distances used in the transportation problem so that the penalties get the right effect; if, for example, the penalties are chosen to an equal value that is too large, then all small supplies will almost be excluded from the problem (unless they are mch needed to satisfy the demand). We have experimented by choosing various values according to the table below. In these tests, the length of the Home-Base-List is 5 , the maximal number of columns returned by branch-and-price is also 5, and we run 3 iterations in the LP stage.

The results in Table 14 show that using the parameters alpha, beta, and gamma helps reducing the customer shortage with up to 70%. The vehicles then are encouraged to visit large supply points, where they can be fully loaded after only one or maybe two visits. In other words, the vehicles will typically visit a larger number of supply points during the day and in this way satisfy more customer demand.

In Table 15 we compare the results obtained by using the reduced transportation problem (RTP) in order to build clusters, instead of the traditional transportation problem (TTP).(Recall that the traditional transportation problem was used in the method presented in Chapter 4.) Since the arc costs in the traditional transportation problem are constants, there is no need of running several LP iterations. However, it is possible to vary the number of enumerated routes in each iteration by varying the two parameters *Max routes* and *#supplies in iter*. When the reduced transportation problem is used, several LP iterations are run and the sequential column generation as described at the beginning of Chapter 6 is applied. Note that small supplies are penalised in Table 15, with alpha = 100, beta = 50, and gamma = 30.

The results in Table 15 show that using the reduced transportation problem leads to better quality routes which in is turn gives better integer solutions and less customer shortage. This shows that the information carried by the dual prices is essential for the enumeration of high quality routes.

TP type	Max routes	#LP iter.	# supplies in iter.	B&P:IP solution	Total # routes	Customers shortage/L
TTP	2,000	1	5	37,761	62,687	10
TTP	2,000	1	10	37,936	127,325	9
TTP	2,000	1	20	40,265	278,022	8
RTP	2,000	1	5	37,761	153,637	6
RTP	2,000	1	10	33,457	215,975	3
RTP	2,000	1	20	36,160	353,178	5
RTP	2,000	2	5,5	34,216	215,153	6
RTP	2,000	3	5,5,5	37,031	272,750	4
TTP	5,000	1	5	37,433	141,738	10
TTP	5,000	1	10	37,756	300,007	9
TTP	5,000	1	20	38,044	644,854	9
RTP	5,000	1	5	35,030	330,249	5
RTP	5,000	1	10	36,485	482,362	4
RTP	5,000	1	20	35,400	812,528	1
RTP	5,000	2	5,5	36,070	496,959	4
TTP	10,000	1	5	38,372	274,412	9
TTP	10,000	1	10	36,712	562,539	10
TTP	10,000	1	20	40,225	1,227,759	7
RTP	10,000	1	5	34,648	575,653	3

RTP	10,000	1	10	36,797	928,482	1
RTP	10,000	1	20	35,749	1,545,974	3
RTP	10,000	2	5,5	34,403	855,264	2

Table 15: Comparison between using the traditional (TTP) and the reduced transportation problem (RTP)

As described earlier in this chapter, the dual prices contain valuable information about the supply and demand points; for example, they indicate whether a supply point has been emptied or not. The real-life case studied here involves an excess of supply which implies that the dual variables corresponding to the supply constraints will mostly be equal to p_k, as shown in Section 6.3. The difficulty that arises then is that the dual prices do not indicate when the supplies are *almost* emptied. For this reason we choose to experiment by creating another extension of a transportation problem, namely the transportation problem with *residual supplies*.

The idea is to modify the supplies in the transportation problem by setting them to exactly the slack values in the supply constraints of LRRMP (that is, the values of y_k). In Table 16 we have used three variants of this residual transportation problem:

- TTP: traditional transportation problem with original supplies and demands
- ResTP: residual transportation problem where both supplies and demands are modified both in the transportation problem with backhaul and in the transportation problem where the home distances are taken into account
- SResTP: residual transportation problem where only supplies are modified in both transportation problems mentioned above
- HomeSResTP: residual transportation problem where the supplies are modified only in the extension of the reduced transportation problem where home distances are taken into account.

In Table 16, we choose alpha = 100, beta = 50, gamma = 30 and we show the length of the Home-Base-List in each iteration. The values of the parameters Max routes is 5,000, and the number of columns added during B&P is 10.

# LP iter.	# supplies in iter.	Transp. Problem type	B&P: IP solution	Total # routes	Cust. Short.	CPU time min
2	5,5	TTP	35,137	496,959	4	10:53
2	2,6	TTP	34,466	430,810	3	10:47
3	2,5,5	TTP	34,510	532,934	1	19:02
2	5,5	ResTP	36,263	119,261	6	2:33
2	2,6	ResTP	36,809	116,890	6	5:07
3	2,5,5	ResTP	38,153	99,218	6	2:47
2	5,5	SResTP	34,327	559,991	4	24:02
2	2,6	SResTP	36,082	512,420	6	15:58
3	2,5,5	SResTP	36,843	626,264	7	10:22

2	5,5	HSResTP	35,589	518,423	3	10:27
2	2,6	HSResTP	36,086	418,901	1	11:05
3	2,5,5	HSResTP	37,370	534,924	2	13:17

Table 16: Comparison of residual transportation problems

The results for the variant of the residual transportation problem where only the supplies are modified, and only in the extension of the transportation problem where the home distances are taken into account (that is the HSResTP),are shown to be satisfactory. In the HSResTP the vehicles start by visiting supply points with large *residual* supplies (since small supplies are penalised).

We can conclude from the tests made in this chapter that the reduced transportation problem improves the solutions obtained from the traditional transportation problem. We also showed that it is essential to penalise small supplies and to use branch.and.price in order to get better integer solutions. Sequential column generation seems to be better than the traditional column generation. Note that all the final restricted master problems in this chapter are solved to optimality by Cplex. Note also that the computational time in these tests is reasonable, which makes the method in Chapter 6 practical for real life problems. It is easy to observe that, in general, the results vary for different parameters value. In order to be able to use the method in Chapter 6 in a computer aided system these parameters need to be assigned good values that are more or less robust. This might be a problem since the method is meant to handle several variants of the LTSP as discussed in Chapter 7.

6.7 Conclusion

The method described in this chapter is an optimisation-based heuristic for the log truck scheduling problem. It involves heuristic column generation using several steps that are performed exactly: the solution of the reduced transportation problem, the pricing of the columns in the pool, and the solution of the final restricted master problem. The results obtained show that this method gives better solutions than the method presented in Chapter 4, which in fact can be seen as a simplification of the method presented in this chapter. The running times for real life instances show that this method is more practically useful than the method in Chapter 5. We also conclude that the idea of penalising small supplies can be essential for finding efficient routes, in the case where the supply available exceeds the demand and the number of small supplies is relatively large. In Chapter 7 we discuss the generality of this method in comparison with, for example, the method in Chapter 4.

7 Discussion and conclusion

The method presented in Chapter 6 has been adapted to a system that allows us to test it on real world data, and specifically, real world distances. In this chapter we briefly describe this system, which is called RuttOpt, and present the data flow between the system and the optimisation method. The method in Chapter 6 is general enough to be applied to the several variants of the LTSP that are encountered at different forestry companies within Sweden. We start by comparing the three case studies and discuss their differences.

The methods in the thesis can probably be improved and made more efficient by adding a heuristic construction algorithm that constructs a feasible or near-feasible solution to the integer problem. In Chapter 7 we discuss some ideas for improving the methods of the thesis and we give some conclusions.

7.1 Comparison between the variants of the LTSP

In the thesis we have worked with cases from three forestry companies, Sydved, Södra, and Holmen Skog, and all three cases are taken from the southern parts of Sweden. As mentioned earlier, the geographical location in Sweden influences the properties of the LTSP, since the southern parts are characterised by a larger number of small supply points, which leads to more splits and thus makes the routing problem more complex. We compare the three cases with regard to the freedom of combining trips and assortments, the vehicles' freedom of visiting supply points, and the amounts available at the supplies.

7.1.1 The Sydved study

This case offers a large degree of freedom concerning the aspects mentioned above:

- All vehicles are allowed to visit all the supply points in the area where the study is made.
- Logs can be transported from any supply point to any demand point with a matching assortment.
- There is a considerable total excess at the supply points (the total supply is four times as large as the total demand) and the number of small supplies is large.
- The working days are divided into two shifts, each consisting of 8 hours of work.
- One break during the day is required in order to switch drivers.

The fact that any vehicle is allowed to visit any supply point makes it possible to use one transportation problem instead of one for each truck.

7.1.2 The Södra study

In the Södra study, wood ships of several assortments are to be transported from supply to demand points. The situation in this case is as follows:

- The vehicles are allowed to visit any supply point in the whole area of the study.
- The quantities available at the supply points can be expressed as a number of trucks-loads, and split pickups are therefore never needed.

- The quantities required at the demand points are in the same way given as numbers of trucks-loads.
- The sum of all supplies is equal to the sum of demands, so that the problem is balanced.
- Each supply point is allowed to send wood ships to a specific customer only. The trips between the supply and demand points are determined by the company in advance, and can therefore not be chosen by the algorithm.
- Wide time windows. In fact, the trucks can be routing 24 hours a day since both supply and demand points can be serviced at any hour of the day.

Hence, this case differs from the previous one in several respects. Note that since the trips between supply and demand points are predefined, it is possible to treat such pairs as nodes in a network and to solve the resulting problem as vehicle routing problem with time windows. Naturally, predefined trips decrease the degree of freedom while enumerating the routes. In this Södra case we could in fact enumerate *all* possible routes of a certain length.

7.1.3 The Holmen Skog study

The degree of freedom in this case lies between the Södra case and the Sydved case. Here each vehicle operates within a given region of supply points which the vehicle is allowed to visit. The characteristics of the Holmen Skog study are the following:

- Each vehicle is allowed to visit only predefined supply points.
- Each supply point is allowed to deliver to a predefined set of customers.
- Each demand is defined by a group of assortments, and not only one assortment as it is in the Sydved case.
- Each vehicle is allowed to have up to three breaks a day.
- Trucks can be routing 24 hours a day.
- The case contains both small supplies and a large excess in the total supply.

In order to deal with the first of these characteristics, the method in Chapter 6 involves solving one transportation problem for each truck.

The variants occuring in the three cases motivate the need for a generic method that can handle many variants of the LTSP. The methods in both Chapters 5 and 6 are able to handle these variants; it is easy to eliminate arcs from the subproblem in Chapter 5, and in this way, enforce additional constraints, if needed. The repeated clustering and route enumeration method can also handle those three variants of the LTSP, although it might not be as efficient for the Södra case as a vehicle routing algorithm, which should therefore be preferable.

Below we describe the RuttOpt system, its advantages and the data flow between the system and the repeated clustering and route enumeration algorithm.

7.2 The RuttOpt system

7.2.1 RuttOpt - description of the system

In 1998, a prototype of a user-friendly optimisation system for log truck scheduling was first developed by Soliton Elektronik AB in cooperation with the Linköping University and SkogForsk. In 2003, a new version was developed by Optimal Solutions. This newer system contains a module that utilises the national data road database and computes the shortest path between any two nodes in the network. The new version of the system is primarily intended to be used for scenario analyses, and SkogForsk has planned for further development of the system.

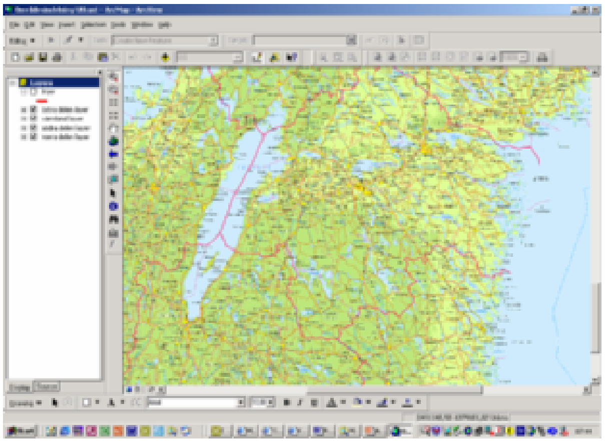


Figure 46: The maps in RuttOpt

RuttOpt has a graphical interface, based on the geographic information system ArcView, that simplifies the planner's work. All input data necessary for generating truck schedules can easily be stored in the system. This data is used to generate maps, to represent the locations of the supply and demand points, and to create the files needed for the optimisation process and the final reports about the schedules generated and other results.

The national road data base includes a large amount of information. In order to create the data concerning the paths and routes in the network, a reduction of the network is necessary. Then, all shortest paths are computed while taking into account several road characteristics, such as: the width of the road segments, the maximum weight that might be transported on the road, the road quality, the speed allowed, et cetera. Once all shortest paths between any pair of given locations have been computed, the information is sent to the optimisation algorithm. In the following section we describe the input and output data that are used to communicate between RuttOpt and the optimisation algorithm.

7.2.2 Input and output data

The input data used in the repeated clustering and route enumeration method consists of 14 files, which contain information about the following:

1. Nodes
2. Supply points
3. Demand points
4. Distances between locations
5. Time to travel between locations
6. Vehicles
7. Time windows
8. Transportation costs
9. Capacities of the vehicles
10. Which supply points that are allowed to be visited by each truck
11. Which supply points that are allowed to deliver to each customer
12. Assortments
13. Products (groups of assortments)
14. Number of planning days

There are four types of nodes: home bases, supply points, demand points, and intermediate nodes where drivers can exchange work shift. In real-life situations, different type of nodes can be situated at the same geographical location, but the system treats them as separate nodes. The trucks are allowed to have up to three stops per day, and each stop is to be made within a time interval. The costs are given in Swedish kronor per kilometre and they are divided into four categories: cost of an empty truck during normal working hours, cost of a fully loaded truck during normal working hours, cost of an empty truck after normal working hours, and finally cost of a fully loaded truck after normal working hours. A truck can stop either at an intermediate node or at any other node type.

The output of the optimisation algorithm is the complete information about the routes chosen in the integer solution; the various locations visited, the order in which they are visited, and the operation performed at each of these locations. The optimisation algorithm also computes the total distance travelled by the trucks (in kilometres).

In the three instances of the LTSP that are studied in the thesis, all trucks used have a crane and are self-loading. RuttOpt handles also the case where (some of) the trucks need a separate crane, that has to be located at the supply points. This is done by defining new supply point, with narrower time windows that depend of the actual presence of this crane.

The RuttOpt system (and the method in Chapter 6) can therefore take all the crucial practical aspects of the log truck scheduling problem into account, and is therefore applicable to real-life instances of the problem.

7.3 Conclusions

In the thesis we consider the log truck scheduling problem as it arises in Swedish forestry. We establish that optimisation-based methods which are derived from the column generation principle are viable approaches to this complex pickup and delivery type problem. We have demonstrated that the number of possible feasible routes is huge and we concluded that it is not possible to enumerate all feasible ones.

The generic model has proven to be practically useful; the number of constraints is low and the number of variables, and thus the size of the problem, can be controlled. However, this generic model, although compact and simple, implicitly includes many complicating constraints. Furthermore, the success of the generic model and the methods based on column generation, depends entirely on the procedure used for column generation and the resulting quality of the routes generated.

We demonstrated how the solutions obtained from the optimisation method were better than the manual planning solutions in both the Sydved and Södra cases. It is impossible to compare the results obtained to the optimal ones since these are not known. The near exact method in Chapter 5 however allows us to compute the gap between the integer solution and the lower bound to the LP relaxation, which gives an idea of the quality of the routes generated. Here, the role of exact column generation is important. For the Sydved case, the near-exact method in Chapter 5 produces an integer solution that is at most a few percent away from an optimal integer solution, as indicated by the lower bound to the LP relaxation. Further, the method in Chapter 4 produces an integer solution of similar quality, and which is therefore also near-optimal. Thereby we can estimate the quality of the solutions obtained from the repeated clustering method in Chapter 6 since it can be seen as an extension of the method in Chapter 4.

The repeated clustering and route enumeration method differs from the cluster-first-route-second method in Chapter 4 by the fact that new clusters are generated iteratively, and that those make use of the dual information from the restricted master problem. The results in Chapter 6 clearly demonstrate the success of this method in comparison with the mere cluster-first-route-second method of Chapter 4.

The LTSP cases studied in the thesis have significantly different properties. As discussed earlier, some instances of the LTSP may in fact be described by a simpler model, namely as a vehicle routing problem with time windows. Even though all variants of the LTSP can be approached by the same generic algorithm, this might not be efficient. We can here observe the classical conflicting objectives that are often encountered in operations research, namely generality of a solution approach opposing its overall efficiency. It is therefore of course essential to decide on the degree of generality required by an optimisation system and how well its methods suit the user of the system.

The column generation principle has during the last decade shown to be a useful computational tool within routing and scheduling. However, column generation is a pure linear programming technique; it aims at solving the LP relaxed problem to optimality rather than solving the original integer problem. This fact affects the quality of the columns generated negatively from the integer programming viewpoint. The experimental results from Chapter 6 have shown that it is hard to even find *any* feasible integer solution, in the sense that all customer demand is fulfilled.

This serious drawback of the column generation principle necessitates the use of branch-and-price, that is, the necessity of adding new columns during the branch-and-bound tree. It also motivates the need of generating a relatively large number of routes instead of only a few; many routes are not efficient from an LP point of view, but can be promising for the integer problem. The pool strategy allows us to store all generated routes and make over-generation of columns practically feasible. But, as seen from the results in Chapter 6, not even this strategy guarantees the finding of integer feasible solutions. We believe that it is necessary to take into consideration the integrality aspect of the problem in the column generation, and in such a way increase the likelihood of finding feasible integer solutions of high quality.

Experiments made on the three cases presented in the thesis show that the methods that are based on the solution of transportation problems, followed by route enumeration, are not robust. This is so since the enumeration process is highly sensitive with respect to parameter values, and needs careful tuning when new instances are being solved. We find the method in Chapter 5 to be more robust than the other two methods. However, one big drawback of the latter is that it is not practical enough to be used as an operative optimisation tool, since the subproblems give rise to huge networks. One way of dealing with this drawback is to reduce the size of the k-shortest paths network. Below we propose ways for achieving this goal.

7.4 Opportunities for future work

7.4.1 Reducing the k-shortest paths network

The size of the k-shortest paths network depends on the number of supply and demand points, and it also depends on the amounts available at the supply points, since these quantities are discretised.

There is typically much more supply points than demand points, and it is therefore the number of the former that is most crucial for the size of the k-shortest paths network. In the instances where the number of supply points that are allowed to be visited by a vehicle is much smaller than the total number of supplies, the k-shortest paths network is automatically reduced. Otherwise, the number of supply points can be decreased artificially by using the information given by the solution of the transportation problem, for example, by including only the supplies that have been chosen in the flow solution.

Regarding the large number of copies of each supply point which is due to the discretisation of the quantities, we start by observing that this discretisation is made to allow different combinations of splits. In the Södra case, where all supplies and demands are expressed as an integer multiple of full truck loads, no splits are necessary and the k-shortest paths network is automatically reduced. In order to reach the same effect in other cases, we suggest aggregating the small supply points so that new nodes are formed. These new nodes have supplies equal to integer multiples of full truck loads. The papers by Laporte and Palekar (2002) and Oppen and Lokketangen (2004), discuss several aspects of clustering and node aggregation.

In order to aggregate several supply points into one we need to consider the assortments available at these points and the distances between them. The creation of one larger node implicitly assumes a route between the original supply points that are contained in the larger node. Figure 47 illustrates the idea of aggregating several supply points.

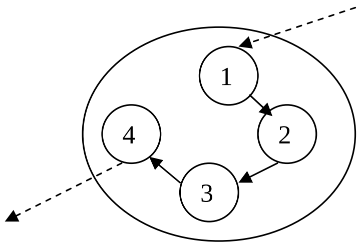


Figure 47: Reducing the number of supply points by aggregation

By aggregating the supply points we not only reduce the number of supplies in the network, but also avoid the need for split pickups (the split type that involves visiting several supply points in order to fill the truck) and thus the need for quantity discretisation.

The reduction of the k-shortest paths network implies much fewer possible paths, which means that the constant *k* can, most probably, also be decreased.

7.4.2 Construction method

A drawback of the methods based on clustering and route enumeration is that we experienced difficulties finding integer solutions, or even LP solutions, that satisfy all the demands of the customers; in these methods we simply enumerate a large amount of feasible routes and hope that there exists a combination of routes, one for each vehicle, that cover all demands. The routes are constructed separately from each other and the only information given to the transportation problem is, in the method in Chapter 6, through the dual variables, which gives no guarantee that the required combination of routes does exist.

One way of dealing with this drawback is to use a construction method that aims at building a feasible (or near-feasible) integer solution, in the sense that it satisfies all demand (see e.g. Kohl and Karisch 2004). This type of construction method to create a set of initial columns is quite common when using the column generation principle. This initial set of columns usually forms the initial restricted master problem. One simple construction method that builds the solution from scratch would, for example, be to build the routes in a greedy fashion in the reduced network proposed earlier. The idea is then to start at the home base of the first truck, visit the closest supply point (which we have arranged to have an integer number of full loads available), load the truck, drive to the closest matching demand point, unload, and continue building the route sequentially while checking the time windows and updating both demands and supplies. Then proceed by constructing the route of the next truck in the updated network. In this way we make sure that the set of routes generated together satisfy all of, or most of, the customers' demands.

As mentioned above, the construction method can be used with the aim to give a set of routes that can be used to create the initial restricted master problem. This initial set of routes clearly leads to better dual prices than those obtained from the solution of the LP relaxation of the initial restricted problem with empty routes only. (Here, we remind the reader that, even the empty solution, is feasible from an optimisation point of view. But the solution based on the corresponding set of dual prices is of course quite useless). After this initiation, the algorithm in the repeated clustering and route enumeration method can proceed as usual. The use of a non trivial initial restricted master problem should of course increase the chance of finding a combination of routes that satisfies all demands in the final restricted master.

Note that a small variation in the approach using the residual transportation problem can give us the initiation method we describe here. The residual supplies and demands in the residual transportation problem are computed based on the solution to the relaxed restricted master problem. If these residuals are instead computed from an integer solution, we automatically obtain a variant of a construction algorithm that uses the transportation problem to decide the candidate list, instead of routing in a greedy fashion.

7.4.3 Advantages of hybrid methods

The repeated clustering and route generation method combines the column generation principle and branch-and-price, which are exact methods, with heuristic route enumeration. Combinations of column generation, and (meta-) heuristics has recently become very popular, since (meta-) heuristics work with integer (feasible) solutions while column generation solves the linear relaxation of the integer problem to optimality, and they therefore, in a sense, complement each other.

An interesting subject for future work is to apply hybrid methods, combining column generation with metaheuristics, to the LTSP. Here, tabu search, which has proven to be one of the best metaheuristics for routing problems, can be applied to generate integer solutions, modify and improve them and, then the routes obtained from the tabu search can all be added to the pool of routes. The goal of such a method is to obtain good integer solutions without using time-consuming branch-and-price, but by only applying branch-and-bound to the routes in the pool.

References:

Appelgren, L. H., “A column generation algorithm for a ship scheduling problem”. *Transportation Science,* 3:53-68, 1969.

Appelgren, L. H., “Integer programming methods for a vessel scheduling problem”. *Transportation Science,* 5:64-78, 1971.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H., “Branch-And-Price: Column generation for solving huge integer programs”. *Operations Research*, 46, 1998.

Bodin, L., Golden, B., Assad, A. and Ball, M., “The state of the art in the routing and scheduling of vehicles and crews”. *Computers and Operations Research,* 10 (2):63-211, 1983.

Brander, A., and Sinclair, M., “A comparative study of *K*-shortest path algorithms”. In *Proceedings of 11th UK Performance Engineering Workshop,* 370-379, 1995.

Carlsson, D., Rönnqvist, M., “Wood flow problems in the Swedish forestry”. Dept of Mathematics, Linköpings Universitet, *LiTH-MAT-R-1998-16.* 1998.

Carlsson, D., Rönnqvist, M. and Sahlin, H., “Operative planning and dispatching of forestry transportation”. Dept of Mathematics, Linköpings Universitet, *LiTH-MAT-R-1998-18.* 1998.

Carlsson, D., Rönnqvist, M. and Westerlund, A., “Extraction of logs in forestry using operations research techniques”. Dept of Mathematics, Linköpings Universitet, *LiTH-MAT-R-1998-17*. 1998.

Cherkassky, B.V., Goldberg, A.V. and Radzik, T., “Shortest paths algorithms: Theory and experimental evaluation”. *Mathematical Programming,* 73:129-196, 1996.

Christiansen, M. and Nygreen, B., “A method for solving ship routing problems with inventory constraints”. *Annals of Operations Research,*81:357-378, 1998.

Cordeau, J-F. and Laporte, G., “The dial-a-ride problem: variants, modeling issues and algorithms”. *4OR- Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1: 89-101, 2003.

Cossens, P., “Evaluation of ASICAM for truck scheduling in New Zealand”. *Logging Industry Research Organisation,* 18(7), 1993.

D’Amours, S., Frayret, J-M. and Rousseau, A., “From the forest to the client- why have integrated management of the value creation network”. Available on the web, (in French) 2004.

Dantzig, G.B. and Wolfe, P., “The decomposition algorithm for linear programming”. *Econometrica,* 29:767-778, 1961.

Desrosiers, J., Dumas, Y., Solomon, M.M. and Soumis, F., “Time constrained routing and scheduling”. In Ball, M.O., Magnanti, T.L., Monmn, C.L. and Nemhauser, G.L., editors, *Network Routing,* Handbooks in Operations Research and Management Science 8:35-139, North-Holland, Amsterdam, 1995.

Dreyfus, S. E., “An appraisal of some shortest-path algorithms”. *Operations Research,* 17:395-412, 1969.

Dror, M., and Trudeau, P., “Savings by split delivery routing”. *Transportation Science*, 23:141-145, 1989.

Dumas, Y., Desrosiers, J., Soumis, F., “The pickup and delivery problem with time windows”. *European Journal of Operational Research*, 54:7-22, 1991.

Eppstein, D., “Finding the k shortest paths”. *SIAM Journal on Computing*, 28 (2):652-673, 1999.

Epstein, R., Morales, R., Seron, J., Traverso, P., and Weintraub, A., “A truck scheduling system improves efficiency in the forest industries”. *Interfaces*, 26 (4):1-12, 1996.

Epstein, R., Morales, R., Seron, J., and Weintraub, A., “Use of OR systems in the Chilean forest industries”. *Interfaces*, 29 (1): 7-29, 1999.

Epstein, R., Nieto, E., Weintraub, P., Chevalier, P., and Gabarro, J., “A system for the design of short term harvesting strategy”. *European Journal of Operational Research*, 119 (2):427-439, 1999.

Ho, S. C., and Haugland, D. “A tabu search heuristic for the vehicle routing problem with time windows and split deliveries”. *Computers and Operations Research*, 31(12):1947-1964, 2004.

Helsgaun, K., “An effective implementation of the Lin-Kernighan traveling salesman heuristic”. *European Journal of Operational Research*. 126, pp. 106-130. 2000

Ioachim, I., Gelinas, S., Desrosiers, J., Soumis, F., "A dynamic programming algorithm for the shortest path problem with time windows and linear node costs", *Networks*, 31:193-204, 1998.

Jimenez, V.M. and Marzal, A., “Computing the K shortest paths: a new algorithm and an experimental comparison”. 3rd Workshop on Algorithm Engineering, London, July 1999. LNCS, vol. 1668. Springer 1999.

Jorgensen, R. M., ”*Planlaegning i svensk skovbrug*”. Institut for Matematisk Modellering. Danmarks Tekniske Universitet. IMM-EKS-1999-8 (in Danish). 1999.

Katoh, N., Ibaraki, T., and Mine, H., “An efficient algorithm for *k* shortest simple paths”. *Networks*, 12:411-427, 1982.

Kohl, N., “*Exact methods for time constrained routing and related scheduling problems*”. PhD thesis. Department of Mathematical Modelling, Technical University of Denmark. 1995.

Kohl, N., and Karisch, S., “Airline crew rostering: problem types, modeling, and optimization”. *Annals of Operations Research*, 127:223-257, 2004.

Laporte, G., and Palekar, U., “Some applications of the clustered travelling salesman problem”. *Journal of the Operational Research Society,* 53:972-976, 2002.

Lasdon, L.S., “*Optimization theory for large systems*”. Macmillan series in Operations Research, 1970.

Linnainmaa, S., Savola, J., Jokinen, O., ”EPO: A knowledge based system for wood procurement management”. Paper presented at the *7th Annual Conference on Artificial Intelligence*, Montreal, 1995.

Lohmander, P. And Olsson, L., “Adaptive optimisation in the roundwood supply chain”. *System Analysis- Modelling- Simulation*, 2004 (forthcoming).

Magnanti, T.L., “Combinatorial optimization and vehicle fleet planning: Perspectives and prospects”. *Networks*, 11:179-213, 1981.

Martell, D.L., “A review of operational research studies in forest fire management”. *Canadian Journal of Forest Research*, 12(2):119-140, 1982.

Martell, D.L., and Tithecott, A., “Development of daily airtanker deployment models”. In *Proceedings of the 1991 Symposium on Systems Analysis in Forest Resources*, 1991.

Martins, E.Q.V., “An algorithm for ranking paths that may contain cycles”. *European Journal of Operational Research*, 18:123-130, 1984.

Martins, E.Q.V., Pascoal, M.M.B. and Santos, J.L.E., “The K shortest paths problem”. Available on the web.1998.

Martins, E.Q.V and Santos, J.L.E., “A new shortest paths ranking algorithm”. *Investiacao Operacional*, 20 (1):47-62, 2000.

Näsberg, M., “Mathematical programming models for optimal log bucking”. Dissertation No.132. Department of Mathematics, Linköping University. 1985.

Olsson, L., “*Optimisation of forest road investments and the roundwood supply chain*”. Doctoral dissertation. Dept. of Forest Economics, SLU. Acta Universitatis agriculturae Sueciae. Silvestria vol. 310. 2004.

Oppen, J., and Lokketangen, A., ”Node aggregation in vehicle routing”. *Paper presented at the Norwegian Informatics Conference*, Stavanger. 2004.

Palmgren, M., ”*Optimisation methods for log truck scheduling*”. Theses No. 880, Linköpings Universitet. 2001.

Palmgren, M., Rönnqvist, M., and Värbrand, P., “A solution approach for log truck scheduling based on composite pricing and branch and bound”. *International Transactions in Operational Research,* 10:433-447, 2003.

Palmgren, M., Rönnqvist, M., and Värbrand, P., “A near-exact method for solving the log-truck scheduling problem”. *International Transactions in Operational Research*, 11:447- 464, 2004.

Papadimitriou, C.H, “The euclidian travelling salesman problem is NP-complete”.*Theoretical Computer Science,* 4:237-244, 1977.

Report No.5, "Efficient, sustainable and ecologically sound forestry”. Conference Papers presented at Elmia Wood. 1993.

Ryan, D.M., Foster, B.A., ”An integer programming approach to scheduling”. *Computer Scheduling of Public Transport*, 269-280, 1981.

Rönnqvist, M., Ryan, D., ”Solving truck despatch problems in real time”. *Proceedings of the 31th Annual Conference of the Operational Research Society of New Zealand*, 165-172, 1995.

Sahlin, H., “*A solution approach for dispatch problems arising in vehicle routing*”. Dept of Mathematics, Linköpings Universitet, LiTH-MAT-Ex-98-14. 1998.

Savelsbergh, M.W.P., Sol, M., ”The general pickup and delivery problem”. *Transportation Science*, 29:17-29, 1995.

Sigurd, M., Pisinger, D. and Sig, M., “The pickup and delivery problem with time windows and precedences”. *Transportation Science*, .38:197-209, 2004.

Solomon, M. M., “Algorithms for the vehicle routing and scheduling problems with time window constraints”. *Operations Research,* 35 (2):254-265, 1987.

Solomon, M. M., and Desrosiers, J. “Time window constrained routing and scheduling problems”. *Transportation Science,* 22:1-13, 1988.

Vanderbeck, F., and Wolsey, L. A., “An exact algorithm for IP column generation”. *Operations Research Letters*, 19:151-160, 1996.

Weintraub, A.,Epstein, R., Morales, R., Seron, J., Traverso, P., “A truck scheduling system improves efficiency in the forest industries”. *Interfaces*, 26:1-12, 1996.

Xu, H., Chen, Z-l., Rajagopal, S., and Arunapuram, S., “Solving a practical pickup and delivery problem”. *Transportation Science*, 37:347-364, 2003.

Yen, J. Y., “Finding the K shortest loopless paths in a network”. *Management Science*, 17:712-716, 1971.