Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Economics and Management
Department of Industrial Economics and Technology Management

Sondre Høyland
Jesper Anker Krogstad

The Snow Grooming Routing Problem with Multiple Depots and Heterogenous Fleet

Comparing an exact solution approach with LocalSolver

Master's thesis in Managerial Economics and Operations Research
Supervisor: Henrik Andersson
June 2019

NTNU
Norwegian University of Science and Technology

Sondre Høyland
Jesper Anker Krogstad

The Snow Grooming Routing Problem with Multiple Depots and Heterogenous Fleet

Comparing an exact solution approach with LocalSolver

Master's thesis in Managerial Economics and Operations Research
Supervisor: Henrik Andersson
June 2019

Norwegian University of Science and Technology
Faculty of Economics and Management
Department of Industrial Economics and Technology Management

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis concludes our Master of Science degrees at the Norwegian University of Science and Technology (NTNU). We specialize in Managerial Economics and Operations Research at the Department of Industrial Economics and Technology Management. The thesis is based on the work we carried out in the subject TIØ4500 - Managerial Economics and Operations Research, in the fall of 2018.

We want to thank Heidi Arnesen at Trondheim Bydrift and Hege Blichfeldt-Sheriff at Skiforeningen for giving us a thorough introduction to the complexity of grooming track networks for cross-country skiing.

We also want to thank our supervisors, Professor Henrik Andersson, Senior Research Scientist at SINTEF Truls Flatberg and Associate Professor Anders N. Gullhav, for their invaluable guidance. Their ideas and feedback have been crucial to our work.

Sondre Høyland and Jesper Anker Krogstad
Trondheim, June 24, 2019

Abstract

There are more than 30,000 kilometers of cross-country skiing tracks in Norway, and to maintain these track networks municipalities and local ski clubs spend more than 250 million NOK every year. The primary cost driver is the daily grooming operations which are manually planned based on the experience of the snowcat operators. Large networks with many vehicles starting at different depots complicate the problem of finding effective routes. The result is unnecessary high costs due to sub-optimal route choices, yielding a benefit of solving the route planning problem.

Employees of the municipal enterprise responsible for the cross country facilities in Trondheim, Trondheim Bydrift, explains that today's planning of grooming activities is based on experience and old habits. As Trondheim is an area known for unstable weather conditions, long-term planning lack robustness. Meetings are therefore conducted every morning to handle the variations. The multifaceted Snow Grooming Problem (SGP) involves multiple depots and a heterogeneous fleet of vehicles, where track segments have numerous attributes. First, they are ranked as requested, priority or regular. Requested segments are segments booked for grooming by stakeholders like ski clubs, ensuring top quality for their training sessions and competitions. The requested segments are handled as mandatory by the facilitators of the track network, and a specific time window is set for grooming operations to obtain a certain standard. The remaining segments are considered optional, where popular segments are prioritized over peripheral parts of the network. The track segments are either broad or narrow. Small vehicles can traverse all tracks, but large vehicles can only traverse broad track segments.

In this report, we further develop an intuitive mathematical model for the SGP faced by cross-country skiing facilitators, which was implemented and tested in the project report leading up to this thesis. The goal is finding optimal routes for the daily grooming operation conducted throughout the winter season, covering as much of the network as possible with the resources available. Solving the arc routing model to optimality by exact

solution methods was not found to handle sufficiently large instances. To solve large, real instances the model is implemented in LocalSolver, a heuristic based commercial solver. A transformation of the arc routing model into a vehicle routing problem is necessary to enable implementation in LocalSolver and is therefore conducted.

The arc routing model is formulated as a Mixed Integer Linear Program (MILP) and implemented in the commercial software Xpress. The model is capable of solving instances up to 20 nodes and 32 edges to optimality within 3,600 seconds, while larger instances require additional runtime. LocalSolver outperforms the exact model, returning satisfactory solutions within seconds on instances successfully solved by Xpress. The heuristic is also able to return solutions on the larger instances, but with significant gaps within the runtime limit of 3,600 seconds.

To our knowledge, this report constitutes the first known attempt of solving the SGP. An interesting area for future research may be to implement the VRP formulation in Xpress for additional comparisons between the ARP and VRP formulations, as well as the heuristic approach. Solving the SGP as a set-partitioning problem before optimizing the routes is also an interesting approach currently not studied.

Sammendrag

Det er mer enn 30 000 kilometer med langrennsløyper i Norge, og for å vedlikeholde og preppe disse løypene bruker kommuner og skiklubber med enn 250 millioner kroner i året. Den største kostnadsdriveren er den daglige preppingen, som er planlagt manuelt basert på erfaring hos maskinførerne. Store nettverk med flere preppemaskiner som starter i forskjellige depoter kompliserer problemet med å finne effektive ruter. Dette resulterer i en unødvendig høy kostnad på grunn av suboptimale rutevalg, og medfører at ruteplanleggingsproblemet med fordel kan løses med optimeringsteknikker.

I Trondheimsområdet er det Trondheim Bydrift som har ansvaret for løypenettverket og vedlikeholdelse av dette. Deres ansatte forteller dagens metode for valg av prepperuter er basert på erfaring og gammel vane. Siden området rundt Trondheim er kjent for ustabile værforhold, mangler langtidsplanleggingen robusthet. Daglige morgenmøter blir avholdt for å ta høyde for variasjoner i vær og andre forhold som spiller inn i planleggingen. Problemet med å preppe skiløyper blir kalt for SGP (the Snow Grooming Problem). Aspekter ved dette problemet som må tas høyde for er at det har flere depoter, at flåten av kjøretøy er heterogen og at løypesegmentene har forskjellige attributter. Løypene blir enten kategorisert som vanlige, prioriterte eller obligatoriske. De obligatoriske løypene er segmenter der prepping er bestilt av en brukergruppe, og har et tidsvindu der prepping må forekomme for gi riktig kvalitet og standard på skiløypene. De resterende løypene blir behandlet som valgfrie, mens populære løyper blir prioritert over perifere deler at løypenettet. Løypene er enten brede eller smale. Små preppemaskiner kan traversere alle løyper, mens store maskiner kun kan kjøre på brede løyper.

I denne oppgaven utvikler vi en matematisk modell for SGP, og sikter mot å finne effektive prepperuter for instanser basert på det lokale løypenettet rundt Trondheim. Målet er å finne gode ruter for den daglige preppingen som blir gjort i vinterhalvåret, der mest mulig av nettverket blir preppet med de tilgjengelige ressursene. Først introduserer vi en kantrutemodell (arc routing problem), som er en intuitiv måte å forstå problemet

på. Å løse denne modellen til optimalitet ved eksakte løsningsmetoder har vist seg å ikke håndtere tilstrekkelige store instanser. Derfor blir en transformasjon til et noderutingsproblem (vehicle routing problem) presentert, og en implementering av dette problemet i en kommersiell heuristikkbasert solver.

Kanruteproblemet er formulert som et MILP (Mixed Integer Linear Program) og implementert i den kommersielle programvaren Xpress. Denne modellen er kapabel til å løse instanser på opptil 20 noder 32 kanter til optimalitet på under 3600 sekunder, mens større instanser krever ytterligere løsningstid. LocalSolver utkonkurrerer den eksakte modellen, og de samme løsningene som den eksakte modellen på 0.20% av tiden for de små instansene. Den håndterer også de større instansene, men for de aller største er gapene relativt store etter 3600 sekunder.

Dette verket fremstår som det første kjente forsøk på å løse SGP. Både problemet og løsningsmetodene er nye med hensyn til litteraturen, så vårt fokus har vært på tekniske aspekter. Problemet er krevende med mange elementer, men ved initielle forsøk virker resultatene lovende. For videre forskning kan det være interessant å implementere VRP-formuleringen i Xpress, for å sammenligne ARP- og VRP-formuleringene videre. Å løse SGP som et set-partitioning problem før man finner optimale ruter kan også være en interessant angrepsvinkel for videre studier.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There are 5360 kilometers of cross-country skiing tracks in Norway's top ten most popular ski facilities. Municipalities, ski clubs and commercial actors in these areas spend more than 50 million NOK in total every season to maintain their track networks (Dagens Næringsliv, 2013).

According to numbers from Statistisk Sentralbyrå (2015b), 30125 km of tracks were planned groomed in 2015, where 6190 km of these were under the responsibility of a municipality. Interpolating the costs from the popular facilities results in a rough estimate of 281 million NOK spent every year grooming Norwegian track networks. These networks are complex, consisting of broad and narrow track segments for the different styles of cross-country skiing. Popular routes must be prioritized and groomed more often than peripheral segments solely used by enthusiasts to maintain a certain standard across the whole network. Grooming requests from local ski clubs, winter sport events and competitions must be handled and included in the planning of the operations. These events often set high standards for the snow grooming, requiring the grooming to be done within specific time windows. Requests vary from being weekly requests known from the start of the season to urgent short-term requests. The fact that requested segments are mandatory complicates the planning.

According to Visit Lillehammer (2016), there were more than 700 snowcats grooming track networks and alpine slopes in 2016. Popular areas have fleets of vehicles grooming simultaneously to meet demand. Statistics Norway show an increase in cabins and holiday homes in Norway, where an expected consequence is an increase in demand for cross-country tracks (Statistisk Sentralbyrå, 2015a). Current track networks will therefore probably undergo an increase in skier traffic, requiring additional networks or expansion of the current ones to be practical. This will further increase the complexity of the SGP.

New technology with GPS tracking of snowcats and websites like *skisporet.no* has made it easier for skiers to find freshly groomed tracks, but for the facilitators of the networks, helpful tools for planning grooming operations is hard to find.

The purpose of this report is to describe the problem of grooming a track network, and based on a mathematical formulation, develop a model optimizing snow grooming operations on test instances. We examine an extension of the classical arc routing problem, where we look at the problem of having multiple depots and a heterogeneous fleet. The track network consists of broad and narrow segments, which are either requested, prioritized or regular. Requested segments have to be served within a given time window while the others can be groomed without regard to time. The optional segments are diversified by priority based on popularity for skiers, but are groomed without precedence to ensure an efficient route through the network. This model can be the core of a decision support system used by popular cross-country skiing areas where the planning of such operations is complex. Arc routing problems for snow plowing and road gritting have similar attributes, but this constitutes the first known attempt of solving the Snow Grooming Problem.

The report starts with an introduction to the different aspects of snow grooming in Chapter 2. Literature related to arc routing is presented in Chapter 3, with special emphasis on attributes relevant to the SGP. Important aspects of the SGP is described in detail in Chapter 4 where we give a detailed description of the problem and its special attributes. A mathematical model of the problem is outlined in Chapter 5. Specifics regarding the implementation in LocalSolver is presented in Chapter 6 followed by test instance generation and conversion in Chapter 7. In Chapter 8, we present the results of a comprehensive computational study and a discussion of the most significant results. In Chapter 9, we conclude our findings and give suggestions for further research.

Instead of making our own heuristic, we have chosen to use a commercial heuristic based solver.

Chapter 2

Background

This chapter starts with an introduction to the process of grooming ski tracks in Section 2.1. In Section 2.2 different tracks networks are presented, followed by Section 2.3 where the various stakeholders of snow grooming operations are discussed. In Section 2.4, the current planning procedure and problems for our test cases, Trondheim Bydrift and Skiforeningen, are introduced and discussed.

2.1 Operations of Snow Grooming

In Norway, the municipalities are often responsible for the ski track networks, usually sharing the responsibility with local ski clubs and commercial actors. The responsibility includes off-season preparations, snow grooming and maintaining the track networks during the season, and for some networks also depositing snow for the next season. Normally the municipality has its own machine park which enables them to cover the complete set of tasks required to ensure quality tracks for its users. An exception to this is the track network in the Oslo area. Here, the responsibility lies with Skiforeningen, an organization whose purpose is to expand the interest in cross country skiing (Skiforeningen, 2019).

Machinery

Grooming the snow, creating the corduroy texture and tracks, requires special machinery and vehicles. The most important asset is the snowcat (see Figure 2.1), a belt-driven vehicle with various areas of use. With an attached snow grooming system the snowcat can fulfill all snow grooming operations for cross-country skiing networks. For the remainder

of the report, we refer to this combined setup as *snowcat* only.

The snowcat shuffles the snow to remove air and compress it, making the snow harder and more durable when skied upon. Weather conditions determine the extent of shuffling needed, but in general, more massive machines and slower speeds have a positive impact on the final result (Raap, 2015). The optimal operating speed is therefore dependent on the current snow conditions, where the extremes are 1 km/h for very icy conditions and as high as 17 km/h for optimal mid-compact conditions. The standard operating speed lays around 12-15 km/h. The temperature also influences the choice of speed, where lower temperatures mean the snowcat must operate a slower pace.

Snowcats are produced in various sizes for different purposes. The ones used for cross-country skiing often are smaller than the ones for alpine slopes, to manage the curvy and narrow features found in many track networks. For cross-country skiing, two different scenarios are influencing the choice of snowcat size. Tracks set for only classic cross-country skiing are narrower, while tracks for skate (often include classic simultaneously) are wider and therefore require larger machinery. A groom for a combination of classic and skate can be seen in Figure 2.2.

Figure 2.1: Example of snowcat used for snow grooming operations

Figure 2.2: Track segment for both classic and skate cross country skiing

Off- and Early-Season Preparations

Track networks require maintenance during the bare season to facilitate good results in the winter. The surroundings and the underlying surface can vary from gravel or asphalt roads in an open landscape to humid swamps in a dense forest. To improve the standard of the network, facilitators look to leveling out bumpy surfaces, removing stumps and other vegetation obstructing the snowcat. Segments like swamps and smaller stream crossings might have to be strengthened with plastic nets and small bridges respectively, to lengthen the season of the track (Vethe, 2018). Re-designing the network is also quite

easily obtainable in less dense forests above the tree line, to increase the network's season in use.

Snowcats cannot be used for all snow levels. They require at least 50 centimeters of regular snow or 30 centimeters of compressed snow (Finnland, 2018). Compressing the snow can be done by snowmobiles to create the necessary base layer. An additional effect of the base layer is lowering the ground temperature in the area, resulting in lower melting rates of snowfalls to come.

Figure 2.3: Granåsen snow deposit

Since the base layer and ground temperature are crucial for ski tracks, especially at the start of the season, some municipalities deposit snow during the summer (see Figure 2.3). In Granåsen, Trondheim they have used this technique to kickstart the season for several years. At the end of the season in 2018, they deposited 28000 m^3 of snow and covered it in sawdust to minimize melting. Approximately 75 % of the snow survived the summer, which is sufficient for 3,3 kilometers of ski tracks, the ski jump facility and the biathlon shooting range before the first snowfall (Jensen, 2018).

2.2 Track Networks

Networks vary in size and complexity from place to place. Some areas have huge and intricate networks, while others have only a couple of closed loops of tracks. The complexity comes from different track widths, the combination of prioritized and requested track segments, track crossings, bridges, lakes, swamps and road crossings. Small networks are usually groomed by a single snowcat, while large networks have a larger fleet and potentially multiple depots. Norway has hundreds of track networks, and Figure 2.4

shows an overview of those in southern Norway. Figure 2.5 is a more detailed map of the track network in Bymarka, Trondheim.

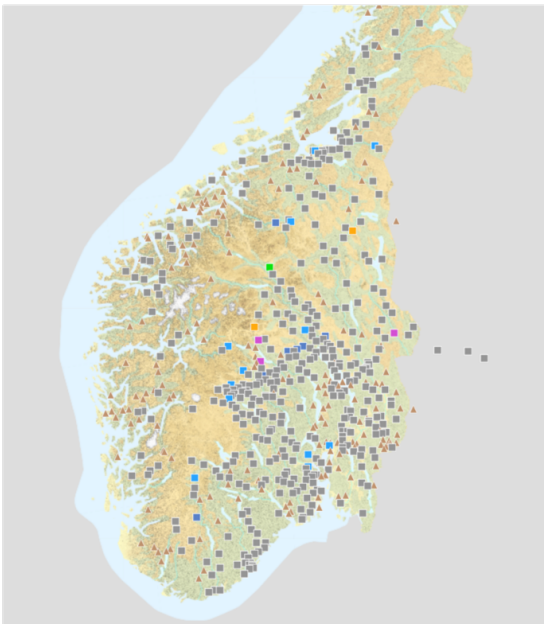


Figure 2.4: Overview of track networks in southern Norway

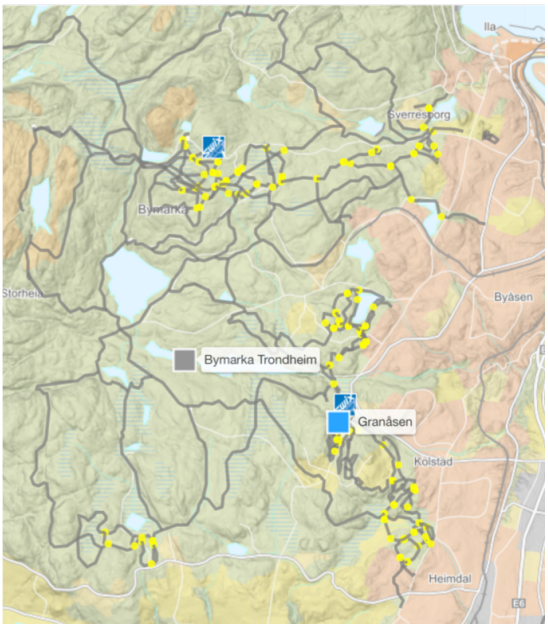


Figure 2.5: The track network in Bymarka, Trondheim

Track networks can be organized in different ways. The snowcat operators themselves usually administer small and medium networks. That is the case for Trondheim Bydrift in the Granåsen network. More extensive networks, like the one Skiforeningen is responsible for, are divided into smaller and more manageable networks, clustering nearby track segments to vehicle depots. Each network has its own snowcats and drivers, which can either be Skiforeningens own employees or volunteers from local ski clubs, working free of charge to ensure a top product for its athletes. Although each area is mostly autonomous in Skiforeningen's case, they also handle grooming requests and passes them on to the responsible groomers of the designated area.

The layout of the network is seldom changed. For Skiforeningen, changing the layout is almost impossible due to strict regulations for outland areas in the Oslo area, stated in Klima- og miljødepartementet (2009) and the number of private landowners. The networks are therefore to a large extent considered fixed, with no possibility of re-designing any segments. The only flexibility in these networks to enhance the effectiveness of the operations is the positioning of the depots. For Trondheim Bydrift the case is quite different, as they do not have any regulations and are the sole landowner of most of the district. This means they the possibility to perform changes and improvements of the network, either for the sake of the athletes or the groomers.

Track networks with stadiums and tracks meant for competitions are often very dense, meaning they have many intersections and relatively short distances between them. Competition tracks are mostly loops of 3, 5 or 10-kilometer loops, and should be groomed in a single route so that skiers can use them at once. Grooming dense networks is often more challenging to plan because of the numerous possibilities, while scattered networks with fewer segments between intersections and longer track segments pose fewer options.

2.2.1 Lakes

The vast majority of tracks are on solid ground, but open fields such as frozen swamps and lakes are popular among skiers. For the snowcats and their operators, this poses a potential hazard of going through the ice. From 2006 up until today, April 2019, 65 snowcats have gone through the ice, killing six people (NVE, 2019). To mitigate the risk of accidents, a minimum requirement for ice thickness along the entire crossing is enforced in some places. In other areas, they do not permit grooming iced waters with snowcats at all. Skiforeningen, as an example, only permits snowmobiles to groom over waters. This complicates the planning even further and often implicates that larger lakes become a boundary between two divided track networks.

2.3 Stakeholders

There are many stakeholders with different agendas regarding the snow grooming operation, the track network and the surrounding nature. The general public is the largest stakeholder, visiting popular cabins serving food as well as the outskirts of the track network. They are to some extent interested in quantity over quality when it comes to grooming, and that the track networks are large enough and to handle popular days. The second largest stakeholder is the ski clubs and high-schools for athletes. They will normally have a set of routes that they regularly use for organized training activities, and are mostly interested in top quality for these specific routes. They will also hand in grooming requests for special events such as competitions. Primary schools and kindergartens often have special requests for the grooming on an occasional basis, often open fields with snow structures as jumps and bumps.

Environmentalists prefer as little activity as possible in nature, demanding less activity in the forests and the mountains. Today's snowcats have fossil fuel engines, and track networks result in the cutting of trees and other interventions in nature. Asphalting track segments close to parking lots to make rollerskiing possible during the summer is also on

their agenda to stop. The municipalities want to facilitate for an active, local, population, while at the same time preserve the regional nature and minimize local pollution in the area.

2.4 Planning of Snow Grooming Operations

Decisions regarding which segments of the network to prioritize early in the season are made in collaboration with the stakeholders. On a day to day basis, it is up to the snowcat operators to determine which tracks to service and when to service them. Planning longer time-spans is difficult and not robust enough, due to aspects like weather changes and short-term grooming requests.

Aspects of Planning

During periods of consistent whether a weekly schedule for the network can be followed, but with an occurrence of precipitation or a change in temperature, the schedule is of little use. A snowfall resets the problem where all previously groomed segments will have to be groomed again. Since consistent weather is the exception for most of the season, the operators experience much overtime.

Fixed grooming schedules for more extended time periods are sub-optimal because the process involves many unpredictable factors. The most unpredictable factor is the weather. To obtain a good result, previous weather conditions combined with forecasts of the coming period has to be taken into account. Snowfall, temperature and wind influence when tracks should be groomed. The optimal scenario is to groom before a cold weather period, resulting in a hard and durable groom, and right after a snowfall for a fresh top layer.

Grooming of track segments can in some municipalities be requested by stakeholders in advance, either as a repetitive request throughout the season or as a one-time request on an occasional basis. These requested tracks *must* be of a certain standard at a given time of the day. To fulfill the requirements regarding the quality, the requested tracks must be groomed within a given time window. The time window is set based on weather forecasts and projected traffic of skiers in the network, a rule of thumb being finished an hour before the start of the event. The total duration of the time window depends on the size of the requested route. The possibility of requesting the grooming of certain tracks results in a further complication of the route planning.

These segments are often characterized by being close to large parking lots or leading to serviced cabins. Other segments are used less, but still, they have to be groomed regularly because of snowfalls. The goal is, therefore, to ensure top track quality for the most popular tracks, while at the same time not neglecting the other segments, substantiating the importance of planning. Track networks with multiple depots are faced with a more complicated planning problem. Coordination of routes to minimize overlapping and non-groomed track segments is obtainable, but hard to plan since the route selection of the operators often is done ad hoc with incomplete information of the others' plan.

With different sizes of snowcats and different widths of tracks comes the problem of assigning the right snowcat to the right track segment. Large snowcats are typically too wide to traverse narrow tracks with only classic skiing, meaning they can only be used on wide tracks (with both classic and skate). The small snowcats are used for the narrow tracks, but can also be delegated to grooming wide tracks. For a small snowcat to service a wide track, it must traverse it twice.

Chapter 3

Literature Survey

Our problem is a type of Arc Routing Problem (ARP), and in this chapter we will focus on ARPs. We review the existing operations research literature on arc routing problems, with special emphasis on the elements special to the snow grooming problem. We start by introducing the history of arc routing and basic notation in Section 3.1. In Section 3.2 we introduce the fundamental routing problems and the notable difference between node routing and arc routing. In Section 3.3 we look at how the fundamental problems can be extended to cover more complex problems, in particular, the attributes related to the snow grooming problem. Section 3.4 gives an overview of recent research on ARPs and compares this to the Snow Grooming Problem. In Section 3.5 we put the snow grooming problem in context with the relevant literature, and our contribution to the field is discussed alongside our motivation for exploring this subject.

Note that in the following, we assume the reader is familiar with the basic terminology and notation within Operations Research.

3.1 Arc Routing

Arc routing problems are a family of problems within operational research, where the aim is to determine the least-cost traversal of a specified arc subset, which may be subject to various constraints (Eiselt et al., 1995). Such problems arise in many practical contexts such as transport logistics and street sweeping and have been studied thoroughly by mathematicians and operations researchers. Mail delivery and garbage collection are Vehicle Routing Problems (VRPs), which means that the activity is on the nodes rather than on the arcs, but are often formulated as ARPs. The first person to formalize an ARP was Euler (1741), with the famous *Bridges of Königsberg* problem. The city of Königsberg (now Kaliningrad) had seven bridges connecting two islands with the two sides of the river. The people of Königsberg wanted to find a route that took them across each bridge *exactly* one time. Euler proved this impossible, and in the process lay the foundations for modern graph theory.

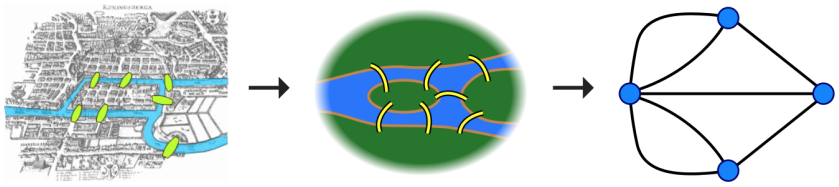


Figure 3.1: The *Bridges of Königsberg* problem, from geografic view to graph representation

There is some basic notation that is common to most ARPs, according to Laporte and Osman (1995). Let $G = (V, E \cup A)$ be a graph where $V = \{v_1, ...v_n\}$ is a set of *nodes*, $A = \{(v_i, v_j) : i \neq j, v_i, v_j \in V\}$ is a set of directed *arcs*, and $E = \{(v_i, v_j) : i < j, v_i, v_j \in V\}$ is a set of undirected *edges*. In most problems, either $A = \emptyset$ or $E = \emptyset$, resulting in either a undirected or directed problem. If $A, E \neq \emptyset$, the problem is called a mixed problem. Associated with each arc or edge there can be a cost c_{ij} or distance d_{ij}.

The term *unicursal* or *Eulerian path* describes a traversal through G containing each arc exactly once and each vertex at least once. If the graph is directed, the number of arcs entering and leaving each vertex must be equal for the existence of an Eulerian path. If an Eulerian path starts and ends in the same vertex, it is called an *Eulerian Cycle*.

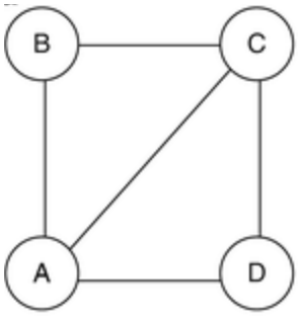


Figure 3.2: Eulerian path: A-B-C-D-A-C

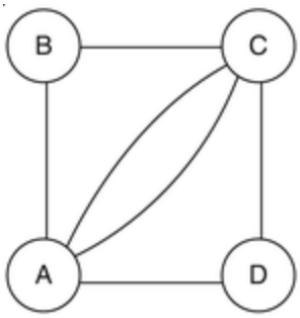


Figure 3.3: Eulerian cycle: A-B-C-D-A-C-A

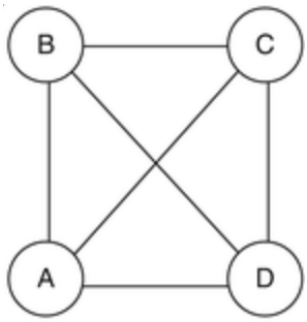


Figure 3.4: Neither Eulerian path or cycle exists

3.2 Fundamental Arc Routing Problems

Arc routing is the process of finding the optimal route through a network, and there are many variations of it. The problem is defined on a graph $G = (V, E \cup A)$, where node v_1 is a depot from which vehicles depart. The objective can be to find a least-cost traversal, to minimize dead mileage, to maximize traversed arcs, etc. Several side constraints can occur such as vehicle capacity, requested arcs, time windows, precedence relations between arcs, and many more.

The most basic ARP is the Chinese Postman Problem (CPP). The problem was first introduced by Kwan (1962) from Shangtun Normal College, China. Kwan defined the problem as "*a mailman has to cover his assigned segment before returning to the post office. The problem is to find the shortest walking distance for the mailman*". Thus the problem is not to cover all *nodes*, but rather to cover all *arcs*. Street sweeping and garbage collection are practical problems that fall under this category. The problem is similar to the *Bridges of Königsberg* problem, as it spans over problems where no Eulerian paths can be found and where the objective is to find a least-cost traversal.

The Windy Postman Problem (WPP) is similar to the CPP, but the cost of traversing an arc is conditioned by the direction it is traversed. The name comes from the fact that it is more strenuous to walk against the wind than with it. In practice, it could be that an arc is either uphill or downhill or that sea currents influence arcs over water.

The arc routing equivalent to the VRP is known as the Capacitated Arc Routing Problem (CARP). It was first suggested by Golden and Wong (1981), and is formally stated as follows: *Given a connected undirected graph G = (N,E,C,Q), where C is a cost matrix and Q is a demand matrix, and given a number of identical vehicles each with capacity W,*

find a number of tours such that 1) Each arc with positive demand is serviced by exactly one vehicle, 2) The sum of demand of those arcs serviced by each vehicle does not exceed W, and 3) The total cost of the tours is minimized. The cost occurs on each traversal, and each arc can be traversed multiple times.

3.3 Extensions

This section will cover ARPs which all have some of the components of the snow grooming problem, and together they will be collectively exhaustive.

In the previously mentioned problems, the fleet is comprised of a set of identical vehicles, known as a *homogeneous fleet*. When the vehicles are *not* identical, e.g. they are characterized by different capacities or costs, the fleet becomes *heterogeneous* (HF). This increases the problem complexity (Baldacci et al., 2008).

The Hierarchical Chinese Postman Problem (HCPP) is a variant of CPP where there are precedence relations on the edges (Corberán and Prins, 2010). The set E is partitioned into subsets, and if subset E_i precedes subset E_j, the edges of E_i must be serviced before the edges of E_j. Generally, the HCPP is NP-hard, but according to Dror et al. (1987) it can be solved in polynomial time if the precedence relations are linear and each subset E_i induces a connected graph.

The Undirected Capacitated Arc Routing Problem with profits (UCARPP) is in the group of ARPs where the objective is not to find a least-cost traversal, but rather to maximize profits ($max\ z$). This group has many names, including "orienteering" and "prize-collecting" problems. The UCARPP is defined on a given graph where profit, demand and travel times are associated with each edge of a set of profitable edges. A homogeneous fleet of capacitated vehicles is given to serve the profitable edges. The profit can be collected by one vehicle only that also has to service the demand on the edge. To maximize the total collected profits, the aim is to find a set of routes that satisfy the capacity and total time constraints. Archetti et al. (2010) treats this problem thoroughly, and gives both exact and heuristic solution methods.

Privatized Rural Postman Problems (PRPP), also referred to as the Prize-collecting Rural Postman Problem, is analyzed thoroughly by Aráoz et al. (2006). The problem is defined on an undirected graph, which contains a distinguished vertex d, called the Depot. Each edge has a profit and a cost associated with traversing it, although the profit can only be collected on the first traversal. The objective is to find a closed cycle C^* that passes

through d and maximizes the sum of the values on the edges traversed in C^*. Solutions to this problem are further researched in the works of Aráoz et al. (2009).

The Team Orienteering Arc Routing Problem (TOARP) was recently reviewed by Archetti et al. (2013). In addition to a set of potential customers that have a profit that is collected when it is serviced, that is, when the associated arc is traversed, the problem contains a set of regular customers that have to be serviced. The profit from a customer can only be collected once. A fleet of homogeneous vehicles with a given maximum traveling time is available. The objective is to identify the customers which maximize the total profit collected while satisfying the given time limit for each vehicle.

Time Windows (TW) are introduced to ARPs when there exist time horizons for the demand. An arc $(ij) \in \mathcal{A}$ have a demand D_{ij} and a time window $[\underline{T}_{(ij)}, \overline{T}_{)ij)}]$, where $\underline{T}_{ij}$ and $\overline{T}_{ij}$ represent the first and latest time to service arc (ij). Time windows can be enforced hard or soft. Hard enforcing means that the constraints must be met, whereas soft enforcing imposes a punishment for breaking the constraints. Reghioui et al. (2007) introduces CARP with Time Windows (CARPTW) and a greedy randomized adaptive search procedure (GRASP) heuristic to solve it.

In the work of Amberg et al. (2000) on Multiple Center Capacitated Arc Routing Problems (MCCARPs), two interesting extensions are introduced: multiple centers (MD) and customer priority (CP). *Multiple centers* expand the model by introducing more than one depot node so that vehicles start at different depots. *Customer priority* comes from the fact that some edges may have a higher priority than others. Possible reasons for this can be to prioritize backlogged edges, edges that are more valuable than others or edges that for some other reasons are deemed more important than others. Amberg et al. (2000) suggest adding an additional objective function to account for customer priority.

3.3.1 The Snow Grooming Problem Compared to Other ARPs

The Snow Grooming Problem (SGP) is multifaceted with respect to the attributes associated with it. As Table 3.1 shows, all the attributes are covered to some degree by other ARPs. To properly solve the Snow Grooming Problem all the mentioned attributes must be taken into account, as we will show in our mathematical model in Chapter 5.

Table 3.1: SGP compared to other ARPs

	NDA	**HF**	**Q**	**Max z**	**$P_{(ij)}$**	**MD**	**TW**	**CP**
CPP	x	-	-	-	-	-	-	-
HCPP	x	-	-	-	-	-	-	x
CARP	-	-	x	-	-	-	-	-
UCARP	x	-	x	-	-	-	-	-
MCCARP	-	-	x	-	-	x	-	x
CARPTW	-	-	x	-	-	-	x	-
PRPP	x	x	x	x	x	-	-	-
TOARP	-	x	x	x	x	-	-	x
SGP	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**

Table 3.1 compares the Snow Grooming Problem to other relevant ARPs. Readability is sacrificed for compactness, so a description of the abbreviations follows: *NDA* is whether the graph is directed or undirected (NonDirectional Arcs). *HF* stands for Heterogeneous Fleet. *Q* is whether or not the vehicles are capacity constrained, and *Max z* is if the objective function is a minimizing or maximizing function. $P_{(ij)}$ tells us if there is a price to be collected on each arc/edge (ij). *MD* is an indicator of the existence of Multiple Depots, *TW* stands for Time Windows and *CP* tells us if there is Customer Priority. The conclusion to be drawn from Table 3.1 is that not only is this the first time the Snow Grooming Problem is researched, but it is also the first time a problem with all these features are looked at.

3.4 Applications of ARPs

For applications on ARPs, we will be reviewing the most recent research on the field. The following articles all share similarities with the Snow Grooming Problem presented in this thesis and will be discussed and compared thematically. The articles are cited repeatedly and are therefore assigned a letter for recognition to ease the study.

[A] Solving the Large-Scale Min–Max K-Rural Postman Problem for Snow Plowing Quirion-Blais et al. (2017a)

[B] Optimization Models for a Real-World Snow Plow Routing Problem Kinable et al. (2016)

[C] A Decision Support Approach for Postal Delivery and Waste Collection Services Abbatecola et al. (2016) (vis at brukes ARP-metoder)

[D] Routing Problems with Time Dependencies or how Different are Trash Collection or Newspaper Delivery from Street Sweeping or Winter Gritting? Golden et al. (2017)

[E] A case study of combined winter road snow plowing and de-icer spreading Quirion-Blais et al. (2017b)

3.4.1 Objective Function

All the covered articles aim to minimize the objective function, but there are some interesting differences. Article [A] and [E] consider snow plowing, and aim to minimize the total *time* spent plowing. [A] has a set of priority classes, and weighs the time according to class. [E] also weighs according to priority classes, but introduces a penalty for the time spent deadheading (i.e. driving without plowing). They both have a time counter and regard the latest completion time of all the jobs.

Article [B] also considers snow plowing, but their objective function is a bit different as it is only concerned about the last job that is finished. The objective is to minimize the duration of the longest route, i.e. to minimize the makespan of the schedule.

In article [C] and [D] the aim is to minimize the total *length* covered. Article [C] covers the problems of postal delivery and waste collection, while article [D] introduces a basic model that can be applied to various arc routing problems.

The Snow Grooming Problem, on the other hand, aims to *maximize* the total length of groomed ski tracks. This is because the fixed cost of the fleet is considered constant, and the problem owners wish to provide as good a service as possible. The practical difference between these objectives is that when maximizing with fixed costs, another constraining parameter must be used. In the case of SGP, there is a penalty for traversing already groomed tracks, which is somewhat comparable to deadheading in snow plowing operations.

3.4.2 Arcs, Edges or Mixed

To get from one node to another, one must traverse an arc or an edge. An edge is a connection that has no orientation and is undirected, whereas an arc is directed and can

only be traversed in the given direction. Depending on whether the graph of the problem contains edges, arcs, or a mix of these generally induce different solving strategies. In the context of article [A] and [B], the graph is a mix of arcs and edges. In [A], all roads are defined as edges, unless they are one-way restricted. Arcs must be serviced in the given direction, and edges must be serviced once in any direction. In [B], roads with one and two lanes respectively translates to two or four unidirectional arcs. If a road is narrow enough to be serviced by only one traversal, it is considered an edge.

Article [C], [D] and [E] all deal with arcs exclusively. This is something that separates the models in these articles from the SGP, as the SGP only contains undirected edges. This simplifies the formulation of the problem, but complicates the graph transformations, which will be discussed in Chapter 6.

3.4.3 Classes of Arcs and Edges

Labeling edges and arcs as "priority" or "requested" is a way to group edges and arcs into classes with different attributes. Several priority classes can be implemented, and handling these classes can be done in different ways. Both [A] and [E] all have a set of priority classes. Common for both articles is that the arcs/edges in a higher priority must be completed before continuing serving the lower prioritized ones. For [A] this rule is implemented on a route level, where individual vehicles must start with the highest priority classes that are available. In [E], on the other hand, the classes are enforced globally over the entire network. Here, no vehicle can start on lower priority classes before all higher classes are completed.

In article [C] and [D] there is no mention of priority classes or requests of any sort, while in [B] priority classes is mentioned as a way to incorporate more features to the model. None of the articles mention the term "requested arcs/edges". This is probably because *all* arcs/edges are requested, and the object is to minimize the cost of servicing them. This attribute is diametrically different from the SGP being a maximization problem. Since the SGP is modeled as a prize collecting problem, it is unreasonable to say that all edges must be serviced. Instead, there is a subset of requested edges for tracks that *must* be serviced. The SGP includes no hierarchy of priority classes, and there is no enforcement regarding time windows for these classes. Instead, they are weighted higher in the objective function.

3.4.4 Time Windows

The introduction of time windows for certain or all arcs/edges is done if there exist some constraints as to *when* the traversal must happen. This can be done to ensure that roads are plowed before rush hours or that mail is delivered in predetermined time slots. Of the five articles in this study, only [C] and [D] address time windows in their models. It is interesting to notice that the models containing objectives measured in *time* does not contain time windows and that the models minimizing *distance* have them. This shows that all models contain both distance and time variables; thus implementing time windows can readily be done. For the SGP, there exist time windows only on requested edges, whereas the time windows introduced in [C] and [D] yields for all arcs.

3.4.5 Exact, Heuristic or Metaheuristic Solution Method

The reader is considered to be familiar with the difference between exact and heuristic solutions, but the distinction between heuristics and metaheuristics may require an explanation. In short, a heuristic is a problem-specific technique fine-tuned to take full advantage of the particularities of the problem. A metaheuristic can be viewed as an extension used to guide an direct the heuristic, so it does not get trapped in a local optimum. The reviewed articles are not consistent in the use of these terms. According to this definition, all the reviewed heuristics are in fact metaheuristics, but the term heuristics will be used for the rest of this report.

The articles display a wide variety of solutions: To solve the problem in article [A] a construction-improvement metaheuristic, based on an improved adaptive large-neighborhood search is given. Article [B] suggests a greedy construction with late acceptance improvement of neighborhood operators heuristic, as well as an exact model and a constraint programming model. In article [C] a two-phase heuristic algorithm is proposed. The first phase is based on a clustering strategy, and the second phase is based on a farthest insertion heuristic. Article [D] gives an exact model with zigzag constraints to give better practical solutions. The model in [E] is solved with an adaptive large neighborhood search heuristic. This algorithm is divided into two steps. Initially, a solution is built using a simple construction heuristic. Then, the initial solution is sent to a best neighborhood improvement step.

Of the articles under review, only [B] presents more than one way of solving the problem. In the SGP an exact model is proposed, but no heuristic. Instead, a way of implementing the model in the commercial solver LocalSolver is shown.

3.4.6 Graph Transformation

In an early paper Pearn et al. (1987) shows how an arc routing problem can be transformed into a vehicle routing problem. More recent studies, like the one done by Longo et al. (2006) show improvements to this transformation, and also that the transformation in many cases is beneficial to the run time of the solver. Accordingly, article [A] and [D] have chosen to introduce a transformation from ARP to VRP. Their approach is based on Longo et al. (2006) studies, but diverge slightly to accommodate the attributes of the specific problem. Arcs and edges are treated differently in the transformation, and the priority characteristic in the ARP model must survive the transformation.

The transformation in [A] starts with splitting the nodes and adds arcs for every possible turn. It then continues by doubling all edged into pairwise opposite arcs. For all the arcs in the ARP model, it then creates a node in the corresponding VRP model and uses Dijkstra's algorithm to compute the distances between the new nodes.

Article [C] contains problems usually modeled as ARPs but chooses to define it as VRP from the start. This way, there is no transformation.

The difference between article [A] and [D] is that the problem in [D] comprises of 2 lane roads, and these road segments are transformed into four arcs represented by four nodes. In [D] there is also no splitting of nodes or suggestion of algorithm for distance computation.

The transformation used in the SGP is thoroughly introduced in Chapter 6.

3.5 Our Contribution

As Corberán and Prins (2010) points out, ARP is not nearly as well researched as its node routing counterpart, although there has been an impressive development in the last two decades. We aim to participate in and contribute to this development as we use arc routing methodology on the problem of grooming ski track networks. As far as our literature research goes, this has never been done before. With this article, we are therefore breaking new ground in the use of arc routing.

Table 3.2 shows an overview of Section 3.4. It clearly shows how the SGP distinguishes itself from other ARPs.

Table 3.2: Comparing the SGP with recent research

Article	Objective function	Arcs, edges or mixed	Priority or request	Time windows	Solution method	Graph transformation
A	Minimizing total completion time	Mixed	Priority	No	Heuristic	Yes
B	Minimize the makespan	Mixed	No	No	Exact, Constraint Programming and Heuristic	No
C	Minimizing total length covered	Arcs	No	Yes	Heuristic	No, defined as VRP
D	Minimizing total length covered	Arcs	No	Yes	Exact	Yes
E	Minimizing total completion time	Arcs	Priority	No	Heuristic	No
SGP	**Maximize length of groomed tracks**	**Edges**	**Priority and request**	**Yes**	**Exact and using LocalSolver**	**Yes**

Chapter 4

Problem Description

In this report, we address an extension of the classic arc routing problem. The problem arises in cross-country skiing facilities around the world when scheduling grooming of the track networks. The network in Granåsen and various networks in the Oslo and Akershus area are used as examples. These networks are groomed and maintained by the municipal enterprises, Trondheim Bydrift and Skiforeningen, respectively. Issues that arise when modeling the problem is discussed in detail in this chapter.

The network is a set of tracks and intersections between them, where each track segment belongs to one of three different size categories; wide, narrow and scooter trail. The characteristics of a track segment can only change at the intersections. In general, the network is non-directional, meaning that it does not matter in which direction the vehicle traverses the edge. A network of ski tracks generally contains many intersections and relatively few track segments between every pair of intersections. Each segment of the track has a well-defined length and topography.

On narrow tracks, the small snowcat can choose to groom either two classic sections or a combination of classic and skate. This is a dynamic setting the operator can set with negligible setup time and cost. For broader track segments, a little snowcat will have to traverse the segment twice to be able to groom the whole width. The solution is often to acquire a larger snowcat for networks with many skate segments, which will only have to traverse these segments once. These large snowcats are unable to traverse the narrower segments. A noteworthy mention is that although wide track segments are named skate segments in this report, they are generally groomed for classic as well (see Figure 2.2, p. 5). Every intersection poses a possibility for change in track type, complicating the route planning and choice of snowcat type for the route. Different vehicles have similar attributes apart from width and range. They operate at identical speeds under the

same weather conditions. Within each vehicle size, there are no significant differences. Therefore all track segments of the same type can be groomed by the same vehicles. The employees have full-time contracts, so costs related to wages are fixed and the available working hours can be fully utilized if necessary.

Figure 4.1 is a snippet of the larger track network in Bymarka, Trondheim, which is presented in the form of a graph in Figure 4.2. The green highlighted segments are prioritized and the red highlighted segments are requested. The double lines represent a wide track segment. The node 10 is an auxiliary node created to connect node 6 and 7 with two different edges. The distance from node 6 to node 10 is therefore set to 0. Node 5 is the depot node, where the fleet consists of one small and one large snowcat.

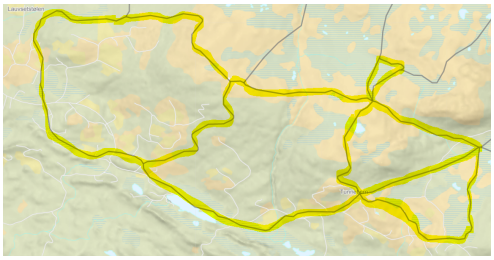


Figure 4.1: Small part of the track network in Bymarka, Trondheim

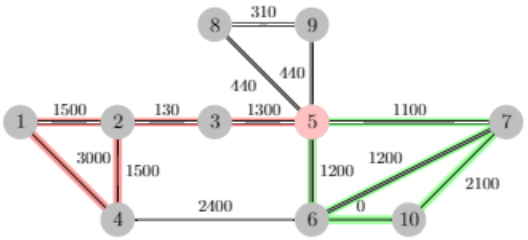


Figure 4.2: Graphical representation of the map in Figure 4.1. Arc distances are labelled, whereas node 10 is an auxilary node

Some networks contain multiple depots, meaning that vehicles start and stop at different locations. When a track segment is groomed, there is no added benefit by grooming it over again, and it may even be undesirable for the skiing conditions during periods of consistent weather conditions. This gives that when faced with multiple depots, the aim is to groom as much of the track network as possible while minimizing double-grooming.

The sequence of which the tracks should be groomed is dependent on various factors. Geographic positioning relative to the vehicle depot, user popularity, and local weather differences are the most important ones. There are also some track segments that are requested by ski clubs and other organizations. The request includes the track segments in question and information about the time of their activities. These segments *have* to be groomed in a given time window before the start time of the event to ensure a good end product. Popular segments are labeled as prioritized during planning to ensure that they are of a good standard.

The problem aims to determine how the track network should be groomed. That is, which vehicles should groom which track segments, and at what time. The purpose is to ensure the effective use of the resources available.

Chapter 5

Mathematical Model

In this section, we introduce the mathematical model based on the problem description in Chapter 4. Some assumptions and simplifications are made to the initial problem, and these are presented in Section 5.1. The notation used in the model is introduced in Section 5.2. In Section 5.3 we present the objective function first, and then the constraints are explained group-wise.

5.1 Modelling Assumptions

Based on the problem description in Chapter 4, some assumptions have to be made. The aim of the problem is to maximize the length of groomed tracks, while at the same time minimizing the time spent grooming. This incentivizes finding efficient driving routes and discourages traversing edges more than once.

Costs of grooming operations are not included in the model. Describing the problem for daily operations, long-term overhead costs like inventory and wages are considered sunk. The weather conditions affect the snow conditions, consequently affecting the grooming operations. Inconsistent weather conditions complicate long-term planning of grooming operations. We have simplified the complexity resulting from weather conditions by limiting the planning horizon to one day. In a single day, the weather conditions are known and consistent and do not include future weather conditions. This way, the weather is factored out of our model.

Juridically, employees are to follow the Work Environment Act (Lovdata, 2006), which is legislation stipulating allowable labor hours for the worker. In practice, the employees

work longer shifts than the law accepts due to snowfalls and other unexpected weather changes. By handling weather conditions as a fixed input, the only impact the weather has is on the operating speed of the vehicles. The model is therefore based on fixed working hours imposed as an upper bound with no possibility of overtime.

The vehicles have a range that is assumed to be constant. In reality, it would vary according to operating speed, and it is possible for the drivers to bring along extra fuel if they embark on long rides. It is hard to quantify the relations between fuel consumption and driving speed, so the available range is determined by fuel capacity and a constant consumption given by the manufacturer.

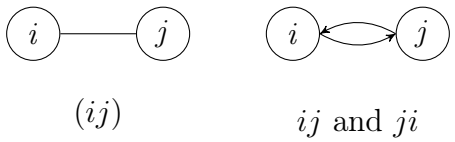


Figure 5.1: Edges and arcs

5.2 Notation

We denote the track segments as *edges* and the intersections as *nodes*. The problem is modeled over a network of edges and nodes. When more than one segment directly connects two intersections, additional nodes and edges are introduced. To ensure a logical flow through the network, the edges are modeled as directed arcs going both ways between two intersections, as shown in Figure 5.1. The arc notation is required to ensure flow and keep track of the time consumption, while the edge notation is required since the direction the segments are traversed is irrelevant for the quality, and therefore irrelevant for the objective.

Each edge can be traversed multiple times and each node can be visited each time, which creates a challenge with describing the route of a vehicle. Since the time an edge is traversed is essential, we introduce the notion of legs to describe the sequence, ass seen in Gundersen et al. (2017). The leg index numbers each edge in the route consecutively, allowing for multiple traversals of edges and multiple visits at nodes.

To handle requested edges, that need to be visited within a given time window, a new variable is created. This variable selects one, of possibly many, traversal and forces it to be within the time window. On broad edges, the request is duplicated to account for the different sizes of vehicles. A large snowcat is able to service both requests at one traversal, whereas the small vehicle can only service one at a time.

Sets

$\mathcal{K}$	Set of vehicles, $k \in \mathcal{K}$
$\mathcal{K}^S$	Set of small vehicles, $\mathcal{K}^S \subseteq \mathcal{K}$
$\mathcal{K}^L$	Set of large vehicles, $\mathcal{K}^L \subseteq \mathcal{K}$
$\mathcal{N}$	Set of possible legs, $n \in \mathcal{N}$
$\mathcal{V}$	Set of nodes, $i, j \in \mathcal{V}$
$\mathcal{E}$	Set of edges, $(ij) \in \mathcal{E}$
$\mathcal{E}^P$	Set of prioritized edges, $\mathcal{E}^P \subseteq \mathcal{E}$
$\mathcal{E}^B$	Set of broad edges, $(ij) \in \mathcal{E}^B$, $\mathcal{E}^B \subseteq \mathcal{E}$
$\mathcal{E}^R$	Set of requested edges with time windows, $(ij) \in \mathcal{E}^R$, $\mathcal{E}^R \subseteq \mathcal{E}^B$
$\mathcal{A}$	Set of arcs, $ij \in \mathcal{A}$

Parameters

$L_{(ij)}$	Length of edge (ij), and arcs ij and ji
S	Operating speed for all vehicles
$T_{(ij)}$	Time used to traverse edge (ij), and arcs ij and ji, $T_{(ij)} = \frac{L_{(ij)}}{S}$
H_k	Available work hours for vehicle k
R_k	Range of vehicle k
$i(k)$	First and last node that vehicle k must visit
W	Award per length for grooming a prioritized edge
C	Penalty per time unit used
$\underline{T}_{(ij)}, \overline{T}_{(ij)}$	Lower and upper bound of time window for edge $(ij) \in \mathcal{E}^R$

Variables

x_{kijn}	$\begin{cases} 1, & \text{if vehicle } k \text{ grooms arc } ij \text{ on leg } n \\ 0, & \text{otherwise.} \end{cases}$
$w_{k(ij)n}$	$\begin{cases} 1, & \text{if edge } (ij) \in \mathcal{E}^R \text{ is serviced by vehicle } k \text{ on leg } n \\ 0, & \text{otherwise.} \end{cases}$
$v_{(ij)}$	$\begin{cases} 1, & \text{if edge } (ij) \text{ is serviced} \\ 0, & \text{otherwise.} \end{cases}$
t_{kn}	start time for vehicle k on leg n
$\tau_{k(ij)}$	time when edge $(ij) \in \mathcal{E}^R$ is serviced by vehicle k

5.3 Model

Objective function

$$\max z = \sum_{(ij)\in\mathcal{E}\setminus\mathcal{E}^P} L_{(ij)} v_{(ij)} + W \sum_{(ij)\in\mathcal{E}^P} L_{(ij)} v_{(ij)} - C \sum_{k\in\mathcal{K}} \sum_{ij\in\mathcal{A}} \sum_{n\in\mathcal{N}} T_{(ij)} x_{kijn} \tag{5.1}$$

The objective function maximizes the length of tracks groomed, rewards grooming of prioritized tracks and penalizes the traversal of a track more than once.

Routing constraints

$$x_{k0i(k)1} = 1 \qquad k \in \mathcal{K} \tag{5.2}$$

$$\sum_{ij\in\mathcal{A}} x_{kijn} = \sum_{ji\in\mathcal{A}} x_{kji(n+1)} \qquad k \in \mathcal{K}, j \in \mathcal{V}, n \in \mathcal{N} \setminus |\mathcal{N}| \tag{5.3}$$

$$\sum_{n\in\mathcal{N}} x_{ki(k)0n} = 1 \qquad k \in \mathcal{K} \tag{5.4}$$

Constraints (5.2) and (5.4) state that vehicle k must start and end its route in its depot $i(k)$, where 0 is an artificial depot to handle multiple depots. This is done in ARP form, by fixating that the first traversed edge for vehicle k starts in super node 0 and ends at its depot $i(k)$, and vice versa with its last traversed edge. The super node is not a part of the set of nodes $\mathcal{V}$. Constraints (5.3) ensure the flow of vehicles through the nodes. It states that if vehicle k enters node j, it must exit this same node on its next leg.

Non-directional constraints

$$v_{(ij)} \leq \sum_{k\in\mathcal{K}^S} \sum_{n\in\mathcal{N}} x_{kijn} + \sum_{k\in\mathcal{K}^S} \sum_{n\in\mathcal{N}} x_{kjin} \qquad (ij) \in \mathcal{E} \setminus \mathcal{E}^B \tag{5.5}$$

$$v_{(ij)} \leq \sum_{k\in\mathcal{K}^L} \sum_{n\in N} x_{kijn} + \sum_{k\in\mathcal{K}^L} \sum_{n\in N} x_{kjin} + \frac{1}{2} \sum_{k\in\mathcal{K}^S} \sum_{n\in N} x_{kijn} + \frac{1}{2} \sum_{k\in\mathcal{K}^S} \sum_{n\in N} x_{kjin} \qquad (ij) \in \mathcal{E}^B \tag{5.6}$$

Constraints (5.5) and (5.6) are constraints that keep track of if an edge (ij) is serviced or not. The right side of the constraints contain both arc ij and arc ji, so the model becomes non-directional. Constraints (5.6) also says that for broad edges a small vehicle must service the edge twice, while large vehicles only need one service traversal. Constraints (5.6) ensure that only small vehicles are counted on narrow edges.

Capacity constraints

$$\sum_{ij\in\mathcal{A}}\sum_{n\in\mathcal{N}} T_{(ij)}x_{kijn} \leq H_k \qquad k\in\mathcal{K} \tag{5.7}$$

$$\sum_{ij\in\mathcal{A}}\sum_{n\in\mathcal{N}} L_{(ij)}x_{kijn} \leq R_k \qquad k\in\mathcal{K} \tag{5.8}$$

Constraints (5.7) make sure the route assigned to vehicle k is shorter than h_k hours. Constraints (5.8) constrain the length of the route based on the vehicles' ranges. Only *one* of these constraints will be binding for each vehicle, depending on the set speed S.

Time window constraints

$$\tau_{k(ij)} \geq t_{kn} - M(1-w_{k(ij)n}) \qquad k\in\mathcal{K},(ij)\in\mathcal{E}^R,n\in\mathcal{N} \tag{5.9}$$

$$\tau_{k(ij)} \leq t_{kn} + M(1-w_{k(ij)n}) \qquad k\in\mathcal{K},(ij)\in\mathcal{E}^R,n\in\mathcal{N} \tag{5.10}$$

$$\underline{T}_{(ij)} \leq \tau_{k(ij)} \leq \overline{T}_{(ij)} \qquad k\in\mathcal{K},(ij)\in\mathcal{E}^R \tag{5.11}$$

$$t_{k(n+1)} \geq t_{kn} + \sum_{ij\in\mathcal{A}} T_{(ij)}x_{kijn} \qquad k\in\mathcal{K},n\in\mathcal{N} \tag{5.12}$$

Constraints (5.9) and (5.10) link the τ-variable to the t-variable. Constraints (5.11) make sure that the requested edges are serviced within their time window. Constraints (5.12) update the starting time for vehicle k on leg $(n+1)$ by adding the time used on leg n to the starting time of leg n.

Requested edges constraints

$$w_{k(ij)n} \leq x_{kijn} + x_{kjin} \qquad k\in\mathcal{K}^L,(ij)\in\mathcal{E}^R,\ n\in\mathcal{N} \tag{5.13}$$

$$2\sum_{k\in\mathcal{K}^L}\sum_{n\in\mathcal{N}} w_{k(ij)n} + \sum_{k\in\mathcal{K}^S}\sum_{n\in\mathcal{N}} w_{k(ij)n} = 2 \qquad (ij)\in\mathcal{E}^R\cap\mathcal{E}^B \tag{5.14}$$

$$\sum_{k\in\mathcal{K}^S}\sum_{n\in\mathcal{N}} w_{k(ij)n} = 1 \qquad (ij)\in\mathcal{E}^R\cap(\mathcal{E}\setminus\mathcal{E}^B) \tag{5.15}$$

Constraints (5.13) link the x-variable with the w-variable, so that a requested edge is considered serviced independent of which direction it is serviced. Constraints (5.14) state that the prioritized track segments must be serviced, either once by a large vehicle or twice by a small vehicle. Constraints (5.15) state that narrow requested edges must be serviced by a small vehicle

Non-negativity and binary constraints

$$x_{kijn} \in \{0,1\} \qquad k \in \mathcal{K},\ ij \in \mathcal{A},\ n \in \mathcal{N} \tag{5.16}$$

$$w_{k(ij)n} \in \{0,1\} \qquad k \in \mathcal{K},\ (ij) \in \mathcal{E}^R,\ n \in \mathcal{N} \tag{5.17}$$

$$v_{(ij)} \in \{0,1\} \qquad (ij) \in \mathcal{E},\ i < j \tag{5.18}$$

$$t_{kn} \geq 0 \qquad k \in \mathcal{K},\ n \in \mathcal{N} \tag{5.19}$$

$$\tau_{k(ij)} \geq 0 \qquad k \in \mathcal{K},\ (ij) \in \mathcal{E}^R \tag{5.20}$$

Constraints (5.16)-(5.18) are binary constraints, while (5.19) and (5.20) ensure non-negativity.

Variables x_{kijn} and $w_{k(ij)n}$ for large vehicles are not generated for narrow edges.

5.3.1 Big-M

Constraints (5.9) and (5.10) contains an M-parameter. The Big-M method is used to ensure that all feasible solutions are obtainable for all allowed values of the variables, while at the same time tightening the formulation. In this case, Big-M is used to require that requested edges are constrained by time windows ($\tau_{k(ij)} = t_{kn}$ when $w_{k(ij)n} = 1$), while the remaining edges are not. The value of M is derived as follows:

For:	$\underline{w_{k(ij)n} = 0}$	$\underline{w_{k(ij)n} = 1}$
	$\tau_{k(ij)} \geq t_{kn} - M$	$\tau_{k(ij)} \geq t_{kn}$
	$\tau_{k(ij)} \leq t_{kn} + M$	$\tau_{k(ij)} \leq t_{kn}$
$\implies$	$-M \leq \tau_{k(ij)} - t_{kn} \leq M$	$\tau_{k(ij)} = t_{kn}$

We see that for M to be as tight as possible, we have that $M = max\{|\tau_{k(ij)} - t_{kn}|\} = max\{|\tau_{k(ij)}|, |t_{kn}|\}$. The maximum values that $\tau_{k(ij)}$ and t_{kn} can take is $max\{\frac{R_k}{S}, H_k\}$. This is analogue to saying that the latest time a leg can be assigned is either the range of the vehicle divides with the speed at which it operates or the available work hours, whichever is the largest. For the rest of the thesis, we use M for simplicity.

5.4 Symmetry-Breaking Constraints

To improve the model introduced in Section 5.3 it is possible to add symmetry-breaking constraints. Symmetries are easily discoverable in the ARP formulation. Even though the fleet of vehicles is heterogeneous, the vehicles are partitioned into subgroups consisting of identical vehicles. Swapping routes in-between identical vehicles yields identical solutions from a practical point of view, but are mathematically different. In these cases, one can permute which vehicle is assigned to each route without changing the optimal solution. Introducing symmetry-breaking constraints removes part of the solution space without cutting away any solutions of practical difference.

To reduce the symmetric solutions we introduce two pairs of lexicographic ordering constraints, 5.21-5.22 and 5.23-5.24.

Number of arcs

$$\sum_{(ij)\in\mathcal{E}^B}\sum_{n\in\mathcal{N}} x_{kijn} \geq \sum_{(ij)\in\mathcal{E}^B}\sum_{n\in\mathcal{N}} x_{(k+1)ijn} \qquad k \in \mathcal{K}^L \mid k < |\mathcal{K}^L| \quad (5.21)$$

$$\sum_{(ij)\in\mathcal{E}}\sum_{n\in\mathcal{N}} x_{kijn} \geq \sum_{(ij)\in\mathcal{E}}\sum_{n\in\mathcal{N}} x_{(k+1)ijn} \qquad k \in \mathcal{K}^S \mid k < |\mathcal{K}^S| \quad (5.22)$$

Constraints (5.21) and (5.22) state that the vehicle with the lowest index number should service the most arcs

Service period

$$\sum_{(ij)\in\mathcal{E}^B}\sum_{n\in\mathcal{N}} T_{(ij)}x_{kijn} \geq \sum_{(ij)\in\mathcal{E}^B}\sum_{n\in\mathcal{N}} T_{(ij)}x_{(k+1)ijn} \qquad k \in \mathcal{K}^L \mid k < |\mathcal{K}^L| \quad (5.23)$$

$$\sum_{(ij)\in\mathcal{E}}\sum_{n\in\mathcal{N}} T_{(ij)}x_{kijn} \geq \sum_{(ij)\in\mathcal{E}}\sum_{n\in\mathcal{N}} T_{(ij)}x_{(k+1)ijn} \qquad k \in \mathcal{K}^S \mid k < |\mathcal{K}^L| \quad (5.24)$$

Constraints (5.23) and (5.24) state that the route with the longest operation duration is designated the vehicle with the lowest index

The two sets of constraints are mutually exclusive, and cannot be implemented at the same time with the guarantee that the optimal solution will not be cut. In Chapter 8.1.1 we show the effect of implementing these sets of constraints.

5.5 Reducing Number of Variables Generated

In an attempt to minimize the runtime of the model, a way to reduce the number of variables generated is suggested. The variable $x_{k(ij)n}$ is by default made for every possible index. This is done uncritically and without insight into the structure of the problem. Choosing not to generate variables that are not in the solution space can possibly make the model run faster. We introduce an algorithm for cutting variables and examine the effect it has on the computation time.

The key insight to this algorithm is that edges far out in the network cannot be reached on early legs (low n).

Algorithm 1 Reducing variable generation

Initialize n
Initialize edges, $(\mathcal{E}^n = \mathcal{E}^0 = \mathcal{E})$
Initialize starting points in depot nodes, $(i(k))$
while Entire network not traversed, $(\mathcal{E}^n \neq \emptyset)$ **do**
    Take one step in all possible directions not yet traversed, start with $n = 1$
    Subtract traversed edges in step n from $\mathcal{E}^n$, $(\mathcal{E}^{n-1} \leftarrow \mathcal{E}^n$ - newly traversed edges)
    $n \leftarrow n + 1$
end while
for each step taken, (n) **do**
    Exclude $x_{k(ij)n}$ from the variable generation for all (ij) in $\mathcal{E}^n$
end for
Return variables

The example graph in 5.2 is used as an illustration for the Algorithm 1. The red edges are all traversable from the depot node 5 and therefore excluded from the set $\mathcal{E}^0$ to make $\mathcal{E}^1$. The green edges are traversable in leg $n = 2$, and similarly cut from the set $\mathcal{E}^1$ to

make $\mathcal{E}^2$. The algorithm continues until all edges are traversed. With the sets $\mathcal{E}^n$ created, the cutting of unnecessary variables can begin.

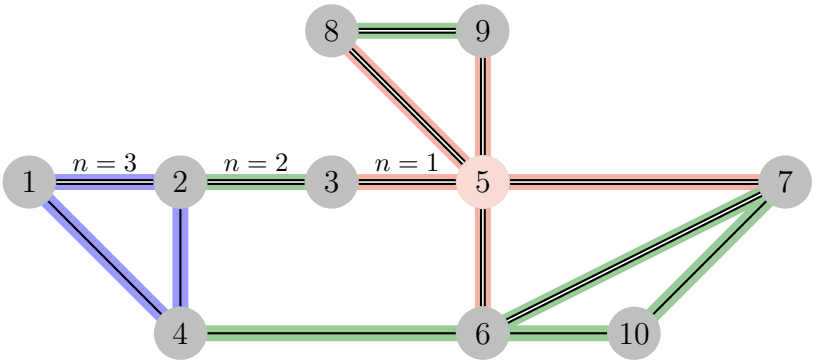


Figure 5.2: Algorithm 1 on an example graph

Chapter 6

Implementation in LocalSolver

6.1 LocalSolver Implementation

As the authors concluded in the project thesis leading up to this masters thesis, solving the Snow Grooming Problem as an ARP to optimality is not effective. The linear solution method is only able to handle rather small instances in a reasonable time, resulting in an insufficient method for practical use. Track networks may contain more than a hundred arcs, while the ARP model only handled 37 within 3600 seconds. To solve larger instances, other solution methods are required. In this chapter, we introduce an approach where the problem is translated into a Vehicle Routing Problem (VRP) and solved in a commercial heuristic based solver called LocalSolver.

6.1.1 LocalSolver

LocalSolver is a mathematical optimization solver unlike classical solvers such as mixed-integer linear programming, constraint programming and nonlinear programming. What is unique with LocalSolver is that it can combine different optimization techniques without needing any parameter tuning. The exact working of the solver is a trade secret, but according to their website LocalSolver (2019) the solver is able to hybridize different optimization techniques dynamically. They claim that it combines constraint propagation and inference techniques, local and direct search techniques, linear and mixed-integer programming, as well as nonlinear programming techniques. Benoist et al. (2011) introduces a closer look into the inner workings of LocalSolver, and shows how it is based on a local search procedure with specially engineered ways of evaluating moves.

LocalSolver is superior to other OR solution technologies when it comes to scalability, and according to Blum and Santos (2019) it outperforms several other solvers. LocalSolver scales nearly linear to problem size. This makes it able to efficiently handle large, real-life problems with reliability and robustness, which is out of scope for classical solvers. Since we have chosen LocalSolver as our commercial solver, we must translate the model from ARP to VRP. This is because LocalSolver does not handle arc type problems. This is likely an economic decision based on the fact that VRP is more researched and more widely used.

6.2 Translating Model from ARP to VRP

Based on the literature study presented in Chapter 3 arc routing is a less researched field than node routing for optimization problems. Solution methods for node routing problems, such as vehicle routing problems (VRP), are therefore more effective and can handle larger instances. Translating an ARP into a VRP result in a more extensive network of nodes and arcs/edges, but the solution methods possible for VRPs are still outperforming ARP methods. Another highly connected reason for translating to VRP is that LocalSolver is not capable of directly understanding ARPs.

The arc routing model introduced in Chapter 5 is an intuitive presentation of the snow grooming problem, where edges and nodes are track segments and intersections, respectively. The characteristics of a VRP limits the possibility of visiting a node more than once, which in the SGP is both legal and necessary to not cut feasible solutions.

Many of our ideas come from Laporte (1997). Their approach on how to transform ARPs to Traveling Salesman Problems (TSPs) is shown to work well on low-density graphs containing few edges. This is according to graphs usually seen in the SGP.

Since each track segment can be traversed in both directions, we start by defining the ARP edges as pairs of opposite arcs. The edge ij for $i < j$ in the original ARP graph gives arcs $\hat{s}$ and $\tilde{s}$ which in turn is turned into nodes in the VRP. Arc $\hat{s}$ is defined for arc (ij) with $i < j$, and $\tilde{s}$ is defined for arc (ij) with $i > j$. The completion of *one* of these nodes corresponds to a fulfilled traversal in the original ARP. For broad arcs we have the nodes $\hat{s}, \hat{s}_2$ and $\tilde{s}, \tilde{s}_2$. To successfully traverse a broad arc in the ARP model, two of these four nodes need to be completed. A small snowcat can visit one node at a time, whereas a large snowcat can visit two at a time. The set of nodes in the VRP is denoted $\mathcal{S}$. To preserve the attributes of the nodes corresponding to edges in the ARP, the pairs of nodes are concatenated into jobs in the set $\mathcal{P}$. Various subsets of these nodes and jobs

are introduced in the complete model below.

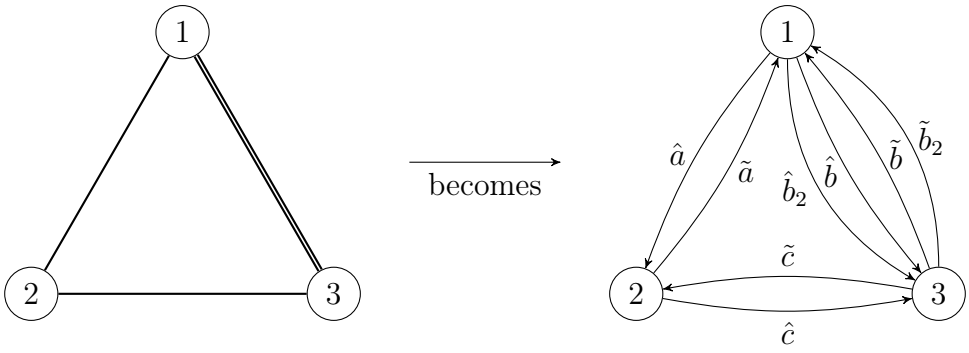


Figure 6.1: From edges to arcs

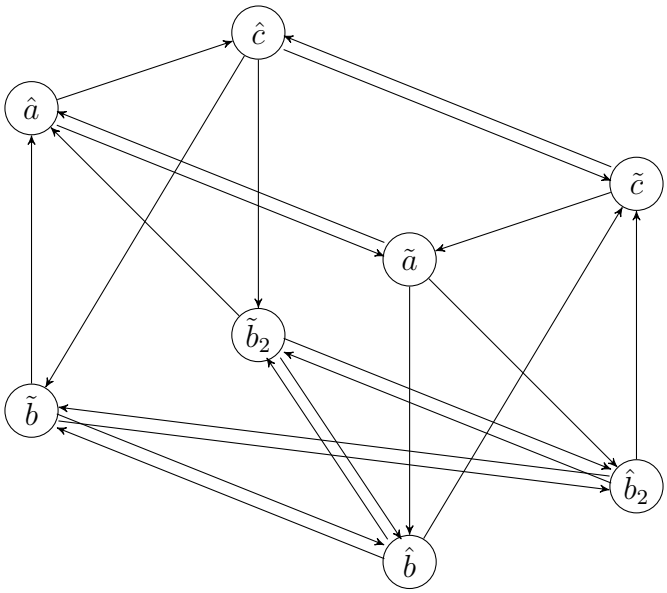


Figure 6.2: Arcs turned into nodes, with new arcs connecting the nodes

The arcs in figure 6.2 gives the underlying information of neighboring jobs and how these jobs are connected in the initial ARP. In VRPs on the other hand, the vehicles must be able to visit any node, regardless of the current position. This means one must have a distance matrix that takes into account the shortest path from any node s to all other nodes and provides the corresponding distance. To produce this distance matrix, the Floyd-Warshall algorithm (Pallottino, 1984) is used on the underlying graph. Figure 6.3 shows the complete transformation to a VRP graph. Together with the distance matrix, we have the basis for our new VRP model.

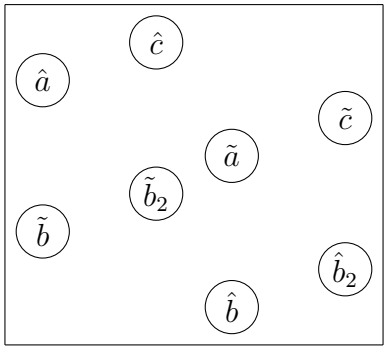


Figure 6.3: The complete VRP graph after transformation

6.2.1 Depot Nodes

The depot node in the ARP model in translated into the VRP by constructing an artificial depot (D_A) and assigning two arcs between these. This gives us segments O_k and D_k for *Origin* and *Destination*, which is the first and last job vehicle k must do. The length of these is set to zero.

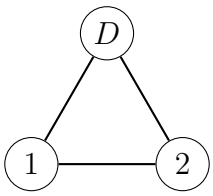


Figure 6.4: Network with depot

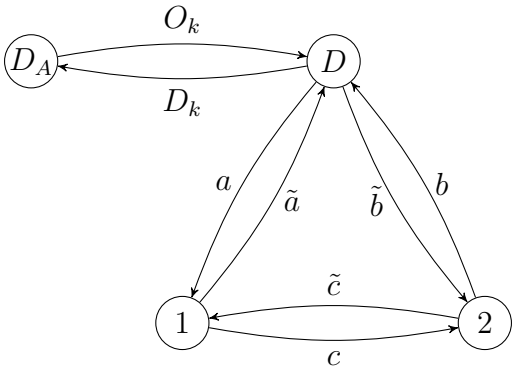


Figure 6.5: Network with artificial depot introduced

For implementation, all arcs are given time windows. The arcs which are not requested are given non-constraining time windows that are open the entire planning horizon. We also set $t_{kO} = 0$ for all k, which corresponds to all vehicles start at time 0.

6.3 The VRP Model

Sets

$\mathcal{K}$	Set of vehicles, $k \in \mathcal{K}$
$\mathcal{K}^S$	Set of small vehicles, $\mathcal{K}^S \subseteq \mathcal{K}$
$\mathcal{K}^L$	Set of large vehicles, $\mathcal{K}^L \subseteq \mathcal{K}$
$\mathcal{A}$	Set of arcs, $(st) \in \mathcal{A}$
$\mathcal{S}$	Set of nodes, $s \in \mathcal{S}$
$\mathcal{P}$	Set of jobs, $p \in \mathcal{P}$
$\mathcal{P}^P$	Set of prioritized jobs, $p \in \mathcal{P}^P,\ \mathcal{P}^P \subseteq \mathcal{P}$
$\mathcal{S}_p$	Set of nodes corresponding to job $p, s \in \mathcal{S}_p,\ \mathcal{S}_p \subseteq \mathcal{S}$
$\mathcal{P}^B$	Set of broad jobs, $p \in \mathcal{P}^B,\ \mathcal{P}^B \subseteq \mathcal{P}$
$\mathcal{P}^R$	Set of requested jobs, $p \in \mathcal{P}^R,\ \mathcal{P}^R \subseteq \mathcal{P}$
$\mathcal{N}$	Set of all nodes including O_k and D_k

Parameters

L_p	Segment length of job (segment) p
L_{st}	Distance between node s and node t
S	Operating speed for all vehicles
T_{st}	Time usage of doing job s and traversing to job t, $T_{st} = \frac{L_{st}}{S}$
H_k	Available work hours for vehicle k
R_k	Range of vehicle k
O_k, D_k	Origin and destination node for vehicle k
W	Unit award for finishing a prioritized job
C	Penalty per time unit used
$\underline{T}_s, \overline{T}_s$	Lower and upper bound of time window for job $s \in \mathcal{S}^R$

Variables

x_{kst}	$\begin{cases} 1, & \text{if vehicle } k \text{ travels from node } s \text{ to node } t \\ 0, & \text{otherwise.} \end{cases}$
v_p	$\begin{cases} 1, & \text{if job } p \text{ fulfilled} \\ 0, & \text{otherwise.} \end{cases}$
t_{ks}	Time vehicle k starts to service node s

$$\max z = \sum_{p \in \mathcal{P} \setminus \mathcal{P}^P} L_p v_p \; + \; W \sum_{p \in \mathcal{P}^P} L_p v_p \\ - \; C \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{N}} \sum_{t \in \mathcal{N}} T_{st} x_{kst} \tag{6.1}$$

$$\sum_{t \in \mathcal{N}} x_{kO_k t} = 1 \qquad k \in \mathcal{K} \tag{6.2}$$

$$\sum_{s \in \mathcal{N}} x_{ksu} = \sum_{t \in \mathcal{N}} x_{kut} \qquad k \in \mathcal{K}, u \in \mathcal{S}, \tag{6.3}$$

$$\sum_{s \in \mathcal{N}} x_{ksD_k} = 1 \qquad k \in \mathcal{K} \tag{6.4}$$

$$v_p = 1 \qquad p \in \mathcal{P}^R \tag{6.5}$$

$$v_p = \sum_{k \in \mathcal{K}^S} \sum_{s \in \mathcal{S}_p} \sum_{t \in \mathcal{N}} x_{kst} \qquad p \in \mathcal{P} \setminus \mathcal{P}^B \tag{6.6}$$

$$v_p = \sum_{k \in \mathcal{K}^L} \sum_{s \in \mathcal{S}_p} \sum_{t \in \mathcal{N}} x_{kst} + \frac{1}{2} \sum_{k \in \mathcal{K}^S} \sum_{s \in \mathcal{S}_p} \sum_{t \in \mathcal{N}} x_{kst} \qquad p \in \mathcal{P}^B \tag{6.7}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{S}} T_{st} x_{kst} \leq H_k \qquad k \in \mathcal{K} \tag{6.8}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{S}} L_{st} x_{kst} \leq R_k \qquad k \in \mathcal{K} \tag{6.9}$$

$$t_{ks} + T_{st} - M_{st}(1 - x_{kst}) \leq t_{kt} \qquad k \in \mathcal{K}, s, t \in \mathcal{S} \tag{6.10}$$

$$\underline{T}_s \leq t_{ks} \leq \overline{T}_s \qquad k \in \mathcal{K}, p \in \mathcal{P}^R, s \in \mathcal{S}_p \tag{6.11}$$

$$x_{kst} \in \{0, 1\} \qquad k \in \mathcal{K}, \; s, t \in \mathcal{N}, \tag{6.12}$$

$$v_p \in \{0, 1\} \qquad p \in \mathcal{P} \tag{6.13}$$

$$t_{ks} \geq 0 \qquad k \in \mathcal{K} \tag{6.14}$$

The objective function (6.1) maximizes the length of fulfilled nodes and is punished for the time used. Constraints (6.2) and (6.4) indicate that each vehicle must leave their depot origin O_k and arrive at their depot destination D_k. Constraints (6.3) state that after a vehicle arrives at a node, it has to leave for another node. Constraints (6.5) demand that all requested jobs are done. Constraints (6.6) and (6.7) keep track of jobs who are to be considered fulfilled in the original ARP problem, for narrow and broad jobs respectively. Constraints (6.8) and (6.9) are capacity constraints, stating that no vehicle

can be assigned a route that exceeds its available working hours or range. Constraints (6.10) establish the relationship between the vehicle departure time from a node and its immediate successor. Finally constraints (6.11) affirm that service within the given time windows is accounted for. (6.12) and (6.13) are binary constraints, and (6.14) are non-negativity constraints.

The Big M in constraint (6.10) can be expressed as:

$$M_{st} = max\left(\overline{T}_s + T_{st} - \underline{T}_t\right) \quad (s,t) \in \mathcal{A} \tag{6.15}$$

For each vehicle, the service start variables in constraints (6.2) impose a unique route direction. This eliminates any subtours and makes classical VRP subtour elimination constraints redundant.

6.3.1 Differences Between the Arc Routing and Vehicle Routing Model

We see that in the VRP model the ARP variable x_{kijn} becomes the VRP variable x_{kst}. It is worth noting that the n-index is missing in the VRP variable. In the ARP model, this index keeps track of which leg the arc (ij) is traversed on. Due to the nature of the VRP model, this index is redundant. We also note that the ARP variable $\tau_{k(ij)}$ is missing in the VRP model, along with the equations (5.9) and (5.10), which is also due to the removed n-index.

Chapter 7

Test Instances

This chapter presents the test instances used to compare the different models in Chapter 8. Section 7.1 describes how the instances are generated for the ARP model introduced in Chapter 5. For the instances to be possible to use as input for the VRP model introduced in Section 6.3 a translation of the instances is necessary. The most crucial considerations regarding this translation is presented in Section 7.2.

7.1 Instance Generation for the ARP Model

Making realistic test cases which mimic real life track networks is challenging. Track networks vary greatly with regards to graph density and how intersections are intertwined and might include special attributes not found generally. Making fictitious networks is therefore considered to not fully capture the nature of real networks. Accordingly, we have decided to replicate real networks in our data sets. These are made manually based on the maps from a public GIS tool covering Norwegian track networks. Although this process is labor intensive, it is considered necessary for getting good test instances with the appropriate features. Information regarding the networks is gathered through interviews with Trondheim Bydrift and Skiforeningen, which are the organizers of the grooming operations in the Trondheim and Oslo municipality, respectively. The information gathered ranges from which track segments are priority or requested and which are broad or narrow. Their fleet of vehicles is also regarded, and the range and operating speed of the vehicles are asserted. For prioritized edges, the corresponding time windows are allocated. Edges that are not requested have time windows set equal to the whole time-span of the operations, never constraining the model.

With the map over the track network as support, we start by numbering all intersection and placing them in the set of all nodes. Then all track segments from intersections i to j with $i < j$ is named ij an put into the set off all edges. Afterward, the subsets of arc type and arc width are created based on the information from Trondheim Bydrift and Skiforeningen. The length of all edges is gathered from www.skisporet.no. Traversing times are computed proportionally based on the length of the edges, meaning no special considerations are taken regarding a change in altitude or local conditions possibly affecting the speed. This minor simplification is backed by the interviews with Trondheim Bydrift and Skiforeningen, where the operating speed is considered constant for equal conditions.

In the real track networks, there are sometimes two track segments directly connecting two intersections. This proposes a problem with naming since there cannot be more than one edge called ij. To solve this an auxiliary node is placed on one of the edges, and a new edge with length 0 is added for connection. For a descriptive figure of this operation, see Figure 4.2 in the Chapter 4.

A total of 11 different test instances for different network sizes have been generated for both the exact and heuristic method. Different fleet compositions, totaling 1-4 vehicles, implies that each instance presented in Table 7.1 holds 24 internal variations. In addition to these, larger instances containing 50-173 nodes have been generated solely for the VRP formulation, to test its performance on practical problem sizes.

Table 7.1: Characteristics of the test instances. Each column holds information regarding the size of the instance. The number of intersections and track segments is indicated by *#Nodes* and *#Edges*, respectively. *#Broad*,*#Priority* and *#Requested* indicates how many broad edges there are in the network, and how many prioritized and requested edges the instance includes. *#Legs* specifies the maximum amount of legs a vehicle's route can exist of.

Instance	**#Nodes**	**#Edges**	**#Broad**	**#Priority**	**#Requested**	**#Legs**
14nodes.txt	14	22	12	6	4	44
16nodes.txt	16	25	12	6	4	50
18nodes.txt	18	28	12	6	4	56
20nodes.txt	20	32	14	6	4	64
22nodes.txt	22	35	16	6	4	70
24nodes.txt	24	38	16	6	6	76
26nodes.txt	26	41	16	6	6	82
28nodes.txt	28	46	16	6	6	92
30nodes.txt	30	49	19	8	6	98
32nodes.txt	32	52	19	8	6	104
37nodes.txt	37	53	21	10	14	106

7.2 Conversion of Input Data for the VRP Formulation

The transition from ARP to VRP results in a substantial change in input data, based on the logic presented in Chapter 6.2. The VRP problem is larger in size than the original ARP, and the number of nodes depends on the number of narrow and wide arcs. The behavior of the transformation is shown in Table 7.2.

Table 7.2: Attributes of the ARP and VRP test instances. Each row shows the transition from the ARP to the VRP representation of the test instances, illustrating the increasing size of the problem. Each edge in ARP becomes a job consisting of a pair of nodes (each broad edge yields an additional, identical, duplicate of the pair) in the VRP. The VRP also includes the artificial depot nodes.

	ARP attributes			**VRP attributes**	
	#Intersections	**#Edges**	**#Broad**	**#Jobpairs**	**#Nodes**
14nodes.txt	14	22	12	22	70
16nodes.txt	16	25	12	25	76
18nodes.txt	18	28	12	28	90
20nodes.txt	20	32	14	32	98
22nodes.txt	22	35	16	35	104
24nodes.txt	24	38	16	38	114
26nodes.txt	26	41	16	41	122
28nodes.txt	28	46	16	46	128
30nodes.txt	30	49	19	49	140
32nodes.txt	32	52	19	52	146
37nodes.txt	37	53	21	53	150

Apart from the artificial depot in the original graph from the ARP, it is possible to reach any other edge in the graph regardless of the vehicle's current position. The transformed graph is therefore very dense, given every node is neighboring every other node, as shown in Figure 6.3. The distances between every node pair cannot be obtained directly from the original graph, and since a node in the transformed graph represents an arc with a different start and end position the distance matrices are computed based on an adapted version of the Floyd-Warshall algorithm. The shortest path algorithm was published by Floyd (1962) and is based on dynamic programming. It has a runtime complexity of $O(V^3)$ where V is the number of nodes in the transformed ARP graph, and the objective is to find the shortest path between every pair of nodes. The distance between a pair has to be the shortest existing path for the objective values of the two models to be comparable. Algorithm 2 is run twice, for both small and large snowcats.

Algorithm 2 Floyd-Warshall

Initialize set of nodes, $\mathcal{N}$
Initialize length of job at every node, $nodeDist$
Initialize start and end point for every node, (n_{start}, n_{end})
Initialize a large value M, e.g. $M = \frac{1}{2}\sum_{p=n}^{N} nodeDist(p)$
for (p in $\mathcal{N}$, q in $\mathcal{N}$) **do**
  if $n_{end}(p) = n_{start}(q)$ **then**
    $DistanceMatrix[p][q] = nodeDist(p)$
  else
    $DistanceMatrix[p][q] = M$
  end if
end for
for (p in $\mathcal{N}$, q in $\mathcal{N}$, r in $\mathcal{N}$) **do**
  if $DistanceMatrix[q][r] > DistanceMatrix[q][p] + DistanceMatrix[p][r]$ **then**
    $DistanceMatrix[q][r] = DistanceMatrix[q][p] + DistanceMatrix[p][r]$
  end if
end for
Return $DistanceMatrix$

The depot nodes are handled in their own vector, rather than as a part of the, where the distance from the depot to every node is computed with the same logic applied as the $DistanceMatrix$ regarding large snowcats.

Chapter 8

Computational Study

In this chapter, we present a computational study of the ARP model described in Section 5.3 and the VRP model in Section 6.3. Section 8.1 provides a preliminary analysis of the exact and heuristic method, ensuring that the best performing versions of both are the ones being compared in 8.2. In Section 8.3 larger test instances not solvable with the exact method are run in LocalSolver. The ARP model has been implemented in Xpress IVE, a commercial optimization software. Version 1.24.24 64 bit of Xpress IVE was used, with Xpress Mosel version 4.8.3 and Xpress Optimizer Version 33.01.02 with the convergence criteria for the duality gap set to 0.001%. The software was run on a computer with the operating system Windows 10 Educational 64-bit. The computer had an Intel(R) Core(TM) i7-7700 CPU @ 3.6 GHz processor with 32 GB RAM internal memory. The VRP model has been implemented in LocalSolver, a heuristic based, commercial solver described in Section 6.1. The LocalSolver Version 8.5 was run on a computer with the operating system macOS Mojave 10.14.4, with Intel(R) Core(TM) i5 CPU @ 2.9 GHz processor with 8 GB RAM internal memory, to return results obtainable by a standard, laptop computer likely to be used by the problem owners.

8.1 Preliminary Computational Study

This section consists of analysis regarding each solution method on their own, ensuring well-performing versions of each are used when comparing the two. Section 8.1.1 presents results from the parameter tuning of the exact method as well as analyzing the effect of the symmetry-breaking constraints in the extended model. In Section 8.1.2, we examine and analyze the effect of different adjustable parameters in LocalSolver.

8.1.1 Parameter Tuning of the Exact Method

In this section, we analyze the impact the input parameters have on runtime and objective value. The practical resource parameters is tested in Section 8.1.1, and the modeling parameter constraining the maximum number of legs in a route is tested in Section 8.1.1.

Resource Parameters

In this section, a comparison of two different approaches for the input parameters is conducted. Obtaining a basis for comparison as the size of data sets increases are done by two different approaches. Aspects like the size of the resource constraints, the number of prioritized and requested edges, the number of legs and the time windows affect the attributes and complicates a comparison when the size increases. We leave the time windows equal for all cases, with no additional requested edges from the smallest instance, *14nodes.txt*.

To mitigate other factors than the actual size of the problem, we use two different approaches. The first approach runs all instances with identical input parameters, based on the average number of legs. This approach causes smaller instances to include excessive amounts of variables and constraints, while the larger instances will be constrained by it. We ran all sets with a maximum of 76 legs, and the range constraints set at 78,000 meters. The second approach increases the parameters and resource constraints, incrementally keeping the ratio between size, parameters, and resources constant. The number of legs, *nLegs* equals twice the number of edges, while the range of snowcats is equal to the average length of an edge multiplied by the number of edges in the data set. Numerous nested loops in the mathematical model increase the search space by n^2 for an increase in *nLegs*, heavily affecting running time. The results can be found in Table 8.1, where the small differences in objective value for the models solved to optimality is due to an accepted duality gap of 0.001% in Xpress.

Table 8.1: Results and run times for different data set sizes with two approaches for setting the value of the parameters

		Identical Parameters			Ratio Parameters				
	Arcs	**Obj.val.**	**Time(s)**	**Gap(%)**	**nLegs**	**Range**	**Obj.val.**	**Time(s)**	**Gap(%)**
14nodes.txt	22	41,538	8.0	-	44	44,000	41,537.6	33.8	-
16nodes.txt	25	50,440	13.4	-	50	50,000	50,435.2	4.6	-
18nodes.txt	28	55,783	126.7	-	56	56,000	55,779.6	175.4	-
20nodes.txt	32	63,631.2	431.3	-	64	64,000	63,631.6	718.7	-
22nodes.txt	35	68,221.6	>3,600.0	0.02	70	70,000	68,230.8	3,175.8	-
24nodes.txt	38	70,827.6	1,706.9	0.01	76	76,000	70,821.6	>3,600.0	0.01
26nodes.txt	41	78,114.8	>3,600.0	0.02	82	82,000	78,113.2	>3,600.0	0.03
28nodes.txt	46	82,319.2	>3,600.0	0.01	92	92,000	82322.8	>3,600.0	0.01
30nodes.txt	49	85,018.8	>3,600.0	0.01	98	98,000	85,015.2	>3,600.0	0.02
32nodes.txt	52	82,995.6	>3,600.0	4.74	104	104,000	83,538.8	3613.5	4.05

The model handles data sets with a size up to 20 nodes, while larger instances run for the maximum 3,600 seconds. The optimality gap is low for all instances but starts to increase for the 32 node instance. Running the model for the real instance at 37 nodes does not find any solution in the first 3,600 seconds and after 8 hours (28,800 seconds) the dual gap is still at 34.89%. After 96 hours the computer crashes due to the size of the branch and bound tree. The dip in runtime for *24nodes.txt* run with identical parameters yields an integer solution was found early, making cuts in the branch and bound tree in the Xpress solver possible at an early stage.

Number of Legs in a Route

In this section, we examine the impact of an input parameter, *nLegs*, which defines the maximum number of segments a snowcat can traverse in a route. The test is run with the *20nodes.txt* data set, where all other parameters are identical for each run. We find accepting duality gaps less than 0.001% suitable given the absolute values of the objective function and runtime of the solver. The accepted duality gaps cause the objective values in Table 8.2 to diverge slightly. For cases where $nLegs \geq 22$ the model stops at different, equivalently good solutions. When the snowcats complete their routes in a maximum of 22 legs, the objective value is **63,634.4**. Logically, all instances with additional legs from the optimal route, here, more than 22, return the same value but the snowcats are traversing different distances, indicating that the routes are different. This yields that additional track segments are covered, where the penalty equals the reward. A consequence of setting *nLegs* excessively high increases the runtime. Setting *nLegs* lower than the optimal value limits the search space, cutting the optimal solution from the

feasible region. Limiting *nLegs* also increases the runtime of the model, which can be interpreted as a complication of the problem.

Table 8.2: Results and run times for *20nodes.txt* with variations of parameter *nLegs*. *Dur.* is the total duration in minutes of a snowcat's route and *Dist.* is the total route distance in meters.

				Small snowcat		Large snowcat	
nLegs	**Obj.val**	**Time(s)**	**Gap(%)**	**Dur.**	**Dist.**	**Dur.**	**Dist.**
50	63,631.2	271.2	0.01	240.2	60,050	114.8	28,700
25	63,632.0	22.3	0.01	225.4	56,350	141.2	35,300
24	63,632.4	2.3	-	187.0	46,750	125.2	31,300
23	63,634.0	8.2	-	177.8	44,450	106.0	26,500
22	**63,634.4**	**17.9**	**-**	**169.8**	**42,450**	**111.6**	**27,900**
21	63,434.8	45.8	-	169.8	42,450	116.4	29,100
20	61,735.6	99.0	-	170.2	42,550	106.8	26,700

Symmetry-Breaking Constraints

In this section, we analyze the effect of the two pairs of symmetry-breaking constraints introduced in Section 5.4. The constraints were implemented in the exact ARP model in Xpress. The two pairs are mutually exclusive, meaning they cut unique solutions from the feasible region when implemented together. They were therefore implemented one at the time, and a comparison of their results is presented in Table 8.3. Each test instance is modified with three different fleet compositions, where $|\mathcal{K}^S|$ and $|\mathcal{K}^L|$ indicates the number of small and large snowcats, respectively. *Original* is the model without any symmetry-breaking constraints. *#Jobs Completed* is the original model including the first set of constraints, given by Equations 5.21-5.22, while *Route Duration* includes the second set, given by Equations 5.23-5.24. *>3,600.0* indicates that the first integer solution was found after the 3,600-second mark but within a reasonably extended time window. The maximum runtime was set to 3,600 seconds to ensure that the model yields results within a time limit satisfying practical use of the model.

Table 8.3: Symmetry-breaking constraints effect on runtime and objective value.

			Original		**#Jobs Completed**		**Route Duration**	
	$\|\mathcal{K}^S\|$	$\|\mathcal{K}^L\|$	**Time(s)**	**Gap(%)**	**Time(s)**	**Gap(%)**	**Time(s)**	**Gap(%)**
14nodes.txt	1	2	6.9	0.01	6.5	0.00	4.6	0.01
14nodes.txt	2	1	9.1	0.01	5.3	0.01	6.2	0.01
14nodes.txt	2	0	109.0	0.01	874.9	0.01	105.5	0.01
16nodes.txt	1	2	9.2	0.01	5.4	0.01	7.0	0.01
16nodes.txt	2	1	12.1	0.01	6.2	0.01	152.9	0.01
16nodes.txt	2	0	334.8	0.01	>3,600.0	0.01	385.1	0.01
18nodes.txt	1	2	304.5	0.01	73.8	0.02	973.1	0.02
18nodes.txt	2	1	>3,600.0	25.44	>3,600.0	49.25	>3,600.0	13.37
18nodes.txt	2	0	1198.9	0.01	>3,600.0	54.32	>3,600.0	4.09
20nodes.txt	1	2	15.2	0.01	14.3	0.01	337.5	0.01
20nodes.txt	2	1	11.5	0.01	17.6	0.01	538.4	0.01
20nodes.txt	2	0	3401.5	0.01	>3,600.0	0.01	>3,600.0	25.42
Average			**>751.1**	**2.13**	>1283.7	8.64	>1109.2	3.75

The average results yield that the symmetry-breaking constraints are outperformed by the original model, even though the feasible region has decreased in size. A plausible reason for this outcome is the unique attributes of the Xpress solver where the symmetry-breaking constraints can prevent the internal heuristics from finding good primal solutions. Given that both the average gap and computation time of the original model are unbeaten by the two extended formulations, neither are included in the complete implementation of the model.

8.1.2 Parameter Testing in LocalSolver

LocalSolver is mostly running the estimation of the parameters internally. The parameters that can be modified are presented in Table 8.4 with their default value, our calibrated value, and a short description. The level of annealing was an adjustable parameter prior to version 8.0 of LocalSolver, but is no longer affecting the search algorithm substantially and is therefore not studied further and set to its default level.

Table 8.4: Adjustable Parameters in LocalSolver. The parameter names are equal to the required input in the command line interface (*Terminal* on macOS. The *Default value* is the value LocalSolver uses if the respective parameter is not assigned a value. *Calibrated value* is the value in this study, used to ensure best possible results.

Parameter	Default value	Calibrated value	Description
lsNbThreads	2	4	Number of threads used to parallelize the search.
lsAnnealingLevel	1	1	Annealing level of the local search.
lsSeed	0	0	Seed of the pseudorandom number generator.

The LocalSolver team recommends the number of threads not to exceed the number of cores, a common attribute of a Central Processing Unit (CPU) bound application. The Intel Core i5 has two real and two virtual cores, resulting in a maximum thread count of four. Increasing the number of threads, increases the robustness of the solver (that is, the chance to find better solutions) and the speed of the search, given the underlying computer power to back it. To validate this rather intuitive statement, we ran our test instances with the number of cores varying from 1-4. Given that all instances returned feasible solutions within 3 seconds, the maximum runtime was set to 10 seconds to discover differences. For the smaller test instances, we did not obtain significant differences, while a thread count of 2 surprisingly outperformed the others on the larger instances. This counter-intuitive behavior of the solver, as well as the significant gaps, required further testing. Increasing the run time limit, which is necessary for the larger instances, four threads produce better results. Given that the practical SGP includes large instances, which require more than a 10-second runtime, the calibrated value is therefore equal to the maximum available computer cores. Test results supporting this can be found in Appendix C.1.

The annealing level of the local search must be an integer between 0 and 9. 0 is the equivalent of a standard descent, meaning that moves deteriorating the current solution are rejected. Increasing the parameter will allow the number of moves in a worse direction, meaning moves yielding lower objective value than the current solution. Intuitively, a high annealing level increases the chance of better solutions, but slows down the convergence of the search, affecting runtime. LocalSolver deprecated the significant impact of the annealing level parameter in version 8.0 and is, therefore, left unchanged. A few random spot checks varying the annealing level on the larger instances returned insignificant differences, backing the statement. Equivalent spot checks were run for the seed of

the pseudorandom number generator with similar results, hence no need to calibrate the default value. Fixing the seed value has no effect on the results since it still generates different numbers from each iteration, impacting the spot check of the annealing parameter as well.

8.2 Comparing the Formulations

In this section, the best performing models for the ARP and VRP in the preliminary computational study are compared. That is the ARP model without any symmetry-breaking constraints and the VRP model with the calibrated LocalSolver parameters given in Table 8.4.

The optimal routes for the arc routing example presented in Figure 4.1 and 4.2 are represented by dotted and dashed lines in Figure 8.1, representing the route of the small and large snowcat, respectively. Double lines indicate a wide track segment, where green segments are prioritized, and red segments are requested.

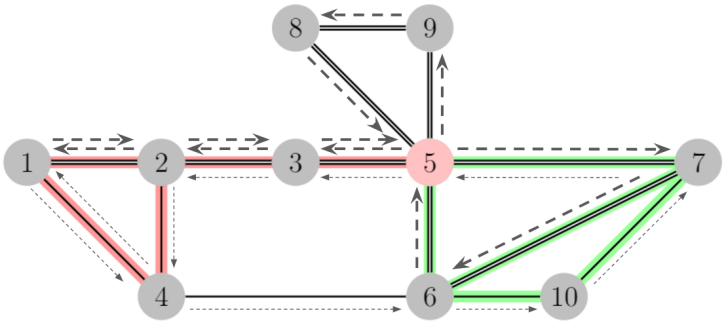


Figure 8.1: Optimal traversal for a small(dotted lines) and a large snowcat(dashed lines) from depot node 5.

For all instances that yielded feasible solutions in the exact model within 3,600 seconds, the heuristic model returned identical routes in 0.20% of the time. The Xpress solver finishes the branch in its branch and bound procedure, resulting in some iterations exceeding the time limit by a few seconds in Table 8.5. The gap for the exact method is the percentage gap between the best solution and the best bound found in Xpress, while the gap for the heuristic is the percentage gap between the equivalent numbers in LocalSolver.

Table 8.5: Results of running test instances in the ARP and VRP model. *Exact* refers to results obtained by the exact ARP model implemented in Xpress, while *Heuristic* is the VRP model implemented in LocalSolver. *Gap* is the gap found by each solver independently.

	Exact		**Heuristic**	
	Time(s)	**Gap(%)**	**Time(s)**	**Gap(%)**
14nodes.txt	33.8	0.00	1.0	0.04
16nodes.txt	4.6	0.00	3.0	0.04
18nodes.txt	175.8	0.00	4.0	0.04
20nodes.txt	718.7	0.00	2.0	0.04
22nodes.txt	3175.8	0.00	5.0	0.04
24nodes.txt	3615.8	0.00	10.0	0.04
26nodes.txt	3607.8	0.00	4.0	0.05
28nodes.txt	3625.0	0.00	8.0	0.04
30nodes.txt	3603.0	0.00	7.0	0.05
32nodes.txt	3613.5	0.04	4.0	0.05
37nodes.txt	3672	7.02	7.0	0.07
Average	**2552.2**	**0.64**	**5.0**	**0.45**

As described in Chapter 6, the VRP model implemented in LocalSolver has to handle larger input data than the equivalent ARP formulation of the problem. The impact of this increase is canceled out by the better solution methods available for VRPs. There are also generated fewer variables in the VRP since it handles flow without *legs*, creating fewer variables and a tighter feasible region. It is clear that the exact method is insufficient as the size of the test instances increase, while the heuristic method yields promising results.

8.3 Heuristic Results on Large Instances

In this section, the performance of the heuristic approach is tested by running the larger test instances the exact method is unable to solve.

To be of practical use for the snow grooming operators, the heuristic should yield good solutions within 3,600 seconds. To test the performance of the heuristic method, we ran four test instances, ranging from 50 to 173 nodes. The smallest instance is expanded incrementally to create the larger instances, where the 173 node instance is an exact replication of the track network found in Bymarka, Trondheim. All fleets consist of two small snowcats and one large. One large and one small snowcat is placed in Granåsen,

while the second small one is placed in the last added node, for *Bymarka.txt* that is Skistua. By doing so, we ensure that the second snowcat is always starting at a point closest to its real depot, Skistua, for smaller test instances as well. Figure 8.2 shows the decrease in gap size for different runtime limits. The yellow graph for the *100nodes.txt* is not visible due to the model returning gap sizes beyond 100% for all time limits. An explanation is that the depot positioning is not optimal and many combinations of the small snowcats has to be checked.

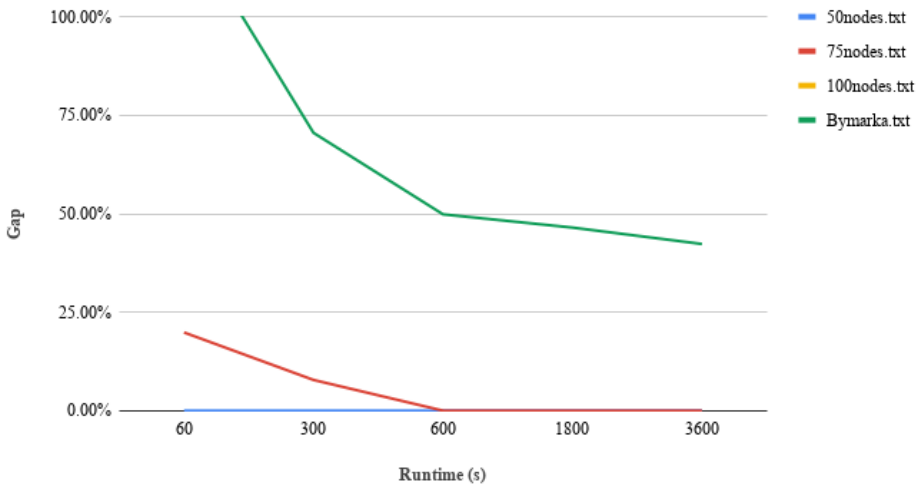


Figure 8.2: Gap results for different runtime limits

Up to 75 nodes, LocalSolver obtains a good solution within 300 seconds, whereas complicated features of the 100 node case create problems and very slow convergence of the search. The real-life instance, *Bymarka.txt*, yields improving results when extending the time limit but is unable to reach a satisfactory good solution within 3,600 seconds. After an extended run of 18,000 seconds, the gap is still as large as 35.78%.

Chapter 9

Concluding Remarks

In our master's thesis, we have described the Snow Grooming Problem (SGP). The problem is found when planning snow grooming operations in ski track networks. Snow grooming vehicles are often called snowcats, and their job is to shuffle the snow and compress it to make durable ski track for cross country skiing. Special for the SGP is the use of a heterogeneous fleet of vehicles to service different sized track segments. A large snowcat can only be used on broad track segments, whereas a small snowcat can traverse both broad and narrow segments. If a small snowcat is to groom a broad track successfully, it must traverse it twice. The problem also takes into account that tracks have different priorities and that some tracks are required and have time windows for when they are to be groomed. These interdependencies make the SGP a problem where computational effort grows rapidly with the size of the track network. The objective of the problem is to maximize the length of tracks that can be groomed during the time and resources the grooming crew has at their disposal. To solve this problem, an exact arc routing model has been proposed and implemented in the exact commercial solver Xpress. This model has also been transformed into a vehicle routing problem and implemented in the commercial heuristic solver LocalSolver. The models have been tested on instances that replicate real track networks.

The arc routing model did not yield promising results, as it only handled small instances and obtained optimal solutions insufficiently fast. The model can solve instances of up to 20 nodes and 32 edges to optimality within 3,600 seconds, while the practical instances are at least twice as large. To handle the larger instances, a heuristic approach was studied. Instead of developing a heuristic from scratch, the commercial heuristic based solver LocalSolver was utilized. To use this solver, a transformation of the model from arc routing to vehicle routing was needed. The transformation involved splitting edges into a

pair of opposite arcs, which in turn give a node in the new graph. The connections between the original arcs and the length of these are used as input in a distance calculation. This calculation is based on the Floyd-Warshall algorithm and is used to give the distance from and to all nodes in the vehicle routing graph. The heuristic method yielded promising results and returned optimal solutions in 0.20% of the time the exact method needed on the comparable test instances. For larger instances, the heuristic was able to handle data sets of twice the size within 3,600 seconds, but returned unsatisfactory large gaps for the real-life instances. As the results of using LocalSolver are promising, we recommend it for future research.

9.1 Future Research

The model proposed to solve the SGP contains all the relevant constraints to mimic the real-life problem properly. We, therefore, claim that there is no need to enrich the two models presented in this thesis, apart from allowing multiple time windows for the same track segment. An interesting area for future research may be to implement the VRP formulation in Xpress for additional comparisons between the ARP and VRP formulations, as well as the heuristic approach. Solving the SGP as a set-partitioning problem before optimizing the routes is also an interesting approach currently not studied.

The SGP can also be reformulated as a design problem to find optimal fleet size or locations of depots. Admittedly, the practical value of this reformulation is debatable. For fleet size, a design model might be of value, although the possible variation in fleet composition is usually sparse. Depot locations are also usually hard to move around. If the aim is to find the best locations for an additional depot, there are usually not many possible positions, and the proposed model can be run with these additional locations to see which location is preferable.

Constructing a bespoke heuristic for the SGP and comparing this with the results from LocalSolver is considered a viable approach. Approaches with genetic algorithms and/or variations of neighborhood search algorithms have yielded good results for similar arc routing problems, and can also be used on the SGP.

Bibliography

Abbatecola, L., Fanti, M. P., Mangini, A. M., and Ukovich, W. (2016). A decision support approach for postal delivery and waste collection services. *IEEE Transactions on Automation Science and Engineering*, 13(4):1458–1470.

Amberg, A., Domschke, W., and Voß, S. (2000). Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124(2):360–376.

Aráoz, J., Fernández, E., and Meza, O. (2009). Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, 196(3):886–896.

Aráoz, J., Fernández, E., and Zoltan, C. (2006). Privatized rural postman problems. *Computers & operations research*, 33(12):3432–3449.

Archetti, C., Feillet, D., Hertz, A., and Speranza, M. G. (2010). The undirected capacitated arc routing problem with profits. *Computers & Operations Research*, 37(11):1860–1869.

Archetti, C., Speranza, M. G., Corberán, Á., Sanchis, J. M., and Plana, I. (2013). The team orienteering arc routing problem. *Transportation Science*, 48(3):442–457.

Baldacci, R., Battarra, M., and Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. In *The vehicle routing problem: latest advances and new challenges*, pages 3–27. Springer.

Benoist, T., Estellon, B., Gardi, F., Megel, R., and Nouioua, K. (2011). Localsolver 1. x: a black-box local-search solver for 0-1 programming. *4or*, 9(3):299.

Blum, C. and Santos, H. G. (2019). Generic cp-supported cmsa for binary integer linear programs. In *International Workshop on Hybrid Metaheuristics*, pages 1–15. Springer.

Corberán, A. and Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69.

Dagens Næringsliv (2013). Norges beste langrennsløyper. `https://www.dn.no/trening/norges-beste-langrennsomrader/1-1-5010947`. Accessed: 2018-11-21.

Dror, M., Stern, H., and Trudeau, P. (1987). Postman tour on a graph with precedence relation on arcs. *Networks*, 17(3):283–294.

Eiselt, H. A., Gendreau, M., and Laporte, G. (1995). Arc routing problems, part i: The chinese postman problem. *Operations Research*, 43(2):231–242.

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140.

Finnland, V. (2018). Professional snow groomer. Sparebankhytta, Granåsen.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345.

Golden, B., Nossack, J., Pesch, E., and Zhang, R. (2017). Routing problems with time dependencies or how different are trash collection or newspaper delivery from street sweeping or winter gritting? *Procedia Engineering*, 182:235–240.

Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, 11(3):305–315.

Gundersen, A. H., Johansen, M., Kjær, B. S., Andersson, H., and Stålhane, M. (2017). Arc routing with precedence constraints: An application to snow plowing operations. In *International Conference on Computational Logistics*, pages 174–188. Springer.

Jensen, H. H. (2018). Lørdag åpner disse løypene. `https://www.adressa.no/pluss/nyheter/2018/11/13/L\T1\ordag-Åpner-disse-l\T1\oypene-17871264.ece`. Accessed: 2018-10-04.

Kinable, J., van Hoeve, W.-J., and Smith, S. F. (2016). Optimization models for a real-world snow plow routing problem. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 229–245. Springer.

Klima- og miljødepartementet (2009). Lov om naturområder i oslo og nærliggende kommuner (markaloven). LOV-2016-08-12-79.

Kwan, M.-K. (1962). Graphic programming using odd or even points. *Chinese Math*, 1(273-277):110.

Laporte, G. (1997). Modeling and solving several classes of arc routing problems as traveling salesman problems. *Computers & operations research*, 24(11):1057–1061.

Laporte, G. and Osman, I. H. (1995). Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262.

LocalSolver (2019). Overview of the localsolver heuristic approach. `https://www.localsolver.com/product.html`. Accessed: 2019-05-01.

Longo, H., De Aragao, M. P., and Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the cvrp. *Computers & Operations Research*, 33(6):1823–1837.

Lovdata (2006). Working environment act. `https://lovdata.no/dokument/NLE/lov/2005-06-17-62`. Accessed: 2019-05-08.

NVE (2019). Isulykker og hendelser. `http://www.varsom.no/ulykker/isulykker-og-hendelser/`. Accessed: 2019-04-01.

Pallottino, S. (1984). Shortest-path methods: Complexity, interrelations and new propositions. *Networks*, 14(2):257–267.

Pearn, W.-L., Assad, A., and Golden, B. L. (1987). Transforming arc routing into node routing problems. *Computers & operations research*, 14(4):285–288.

Quirion-Blais, O., Langevin, A., Lehuédé, F., Péton, O., and Trépanier, M. (2017a). Solving the large-scale min–max k-rural postman problem for snow plowing. *Networks*, 70(3):195–215.

Quirion-Blais, O., Langevin, A., and Trépanier, M. (2017b). A case study of combined winter road snow plowing and de-icer spreading. *Canadian Journal of Civil Engineering*, 44(12):1005–1013.

Raap, K. (2015). Top tips for effective snow grooming. `http://www.snowmobileinfo.org/snowmobile-access-docs/Top-Tips-for-Effective-Trail-Grooming-presentation-notes.pdf`. Accessed: 2018-11-11.

Reghioui, M., Prins, C., and Labadi, N. (2007). Grasp with path relinking for the capacitated arc routing problem with time windows. In *Workshops on Applications of Evolutionary Computation*, pages 722–731. Springer.

Skiforeningen (2019). Om oss. `https://www.skiforeningen.no/omoss/`. Accessed: 2019-02-05.

Statistisk Sentralbyrå (2015a). Bygningsmassen. https://www.ssb.no/bygningsmasse. Accessed: 2018-11-21.

Statistisk Sentralbyrå (2015b). Tabell 11262: Rekreasjon i tettsted. https://www.ssb.no/statbank/table/11262. Accessed: 2018-11-21.

Vethe, K. O. (2018). Skiforeningen vil «forsterke» myr med nett – kan få nei. https://www.budstikka.no/kjekstadmarka/nyheter/asker/skiforeningen-vil-forsterke-myr-med-nett-kan-fa-nei/s/5-55-752646. Accessed: 2018-11-12.

Visit Lillehammer (2016). Turtips og status på løypene. https://www.lillehammer.com/nyheter/2016/1/5/skisporet-no-gir-deg-oppdatert-informasjon-om-langrennsloypene-a2604. Accessed: 2018-11-19.

Appendix A

Compressed Arc Routing Model

Sets

$\mathcal{K}$	Set of vehicles, $k \in \mathcal{K}$
$\mathcal{K}^S$	Set of small vehicles, $\mathcal{K}^S \subseteq \mathcal{K}$
$\mathcal{K}^L$	Set of large vehicles, $\mathcal{K}^L \subseteq \mathcal{K}$
$\mathcal{N}$	Set of possible legs, $n \in \mathcal{N}$
$\mathcal{V}$	Set of nodes (vertices), $i, j \in \mathcal{V}$
$\mathcal{E}$	Set of edges, $(ij) \in \mathcal{E}$
$\mathcal{E}^P$	Set of prioritized edges, $\mathcal{E}^P \subseteq \mathcal{E}$
$\mathcal{E}^B$	Set of broad edges, $(ij) \in \mathcal{E}^B,\ \mathcal{E}^B \subseteq \mathcal{E}$
$\mathcal{E}^R$	Set of requested edges with time windows, $(ij) \in \mathcal{E}^R,\ \mathcal{E}^R \subseteq \mathcal{E}^B$
$\mathcal{A}$	Set of arcs, $ij \in \mathcal{A}$

Parameters

$L_{(ij)}$	Length of edge (ij)
S	Operating speed for all vehicles
$T_{(ij)}$	Time used to traverse edge (ij), $T_{(ij)} = \frac{L_{(ij)}}{S}$
H_k	Available work hours for vehicle k
R_k	Range of vehicle k
$i(k)$	First and last node that vehicle k must visit
W	Unit award for grooming a prioritized edge
C	Penalty per time unit used
$\underline{T}_{ij}, \overline{T}_{ij}$	Lower and upper bound of time window for edge $ij \in \mathcal{E}^R$

Variables

x_{kijn} $\begin{cases} 1, & \text{if vehicle } k \text{ grooms arc } ij \text{ on leg } n \\ 0, & \text{otherwise.} \end{cases}$

$w_{k(ij)n}$ $\begin{cases} 1, & \text{if } (ij) \in \mathcal{E}^R \text{ is serviced by vehicle } k \text{ on leg } n \\ 0, & \text{otherwise.} \end{cases}$

$v_{(ij)}$ $\begin{cases} 1, & \text{if edge } (ij) \text{ is serviced} \\ 0, & \text{otherwise.} \end{cases}$

t_{kn} start time for vehicle k on leg n

τ_{kij} time when $(ij) \in \mathcal{E}^R$ is serviced by vehicle k

Objective function

$$\max z = \sum_{(ij)\in\mathcal{E}\setminus\mathcal{E}^P} L_{(ij)} v_{(ij)} \; + \; W \sum_{(ij)\in\mathcal{E}^P} L_{(ij)} v_{(ij)} - \; C \sum_{k\in\mathcal{K}} \sum_{ij\in\mathcal{A}} \sum_{n\in\mathcal{N}} T_{(ij)} x_{kijn} \tag{A.1}$$

$$x_{k0i(k)1} = 1 \qquad k \in \mathcal{K} \tag{A.2}$$

$$\sum_{ij\in\mathcal{A}} x_{kijn} = \sum_{ji\in\mathcal{A}} x_{kji(n+1)} \qquad k \in \mathcal{K}, j \in \mathcal{V}, n \in \mathcal{N} \setminus |\mathcal{N}| \tag{A.3}$$

$$\sum_{n\in\mathcal{N}} x_{ki(k)0n} = 1 \qquad k \in \mathcal{K} \tag{A.4}$$

$$v_{(ij)} \le \sum_{k\in\mathcal{K}^S} \sum_{n\in\mathcal{N}} x_{kijn} + \sum_{k\in\mathcal{K}^S} \sum_{n\in\mathcal{N}} x_{kjin} \qquad (ij) \in \mathcal{E} \setminus \mathcal{E}^B \tag{A.5}$$

$$v_{(ij)} \le \sum_{k\in\mathcal{K}^L} \sum_{n\in N} x_{kijn} + \sum_{k\in\mathcal{K}^L} \sum_{n\in N} x_{kjin} + \frac{1}{2} \sum_{k\in\mathcal{K}^S} \sum_{n\in N} x_{kijn} + \frac{1}{2} \sum_{k\in\mathcal{K}^S} \sum_{n\in N} x_{kjin} \qquad (ij) \in \mathcal{E}^B \tag{A.6}$$

$$\sum_{ij\in\mathcal{A}} \sum_{n\in\mathcal{N}} T_{(ij)} x_{kijn} \le H_k \qquad k \in \mathcal{K} \tag{A.7}$$

$$\sum_{ij\in\mathcal{A}} \sum_{n\in\mathcal{N}} L_{(ij)} x_{kijn} \le R_k \qquad k \in \mathcal{K} \tag{A.8}$$

$$\tau_{k(ij)} \ge t_{kn} - M(1 - w_{k(ij)n}) \qquad k \in \mathcal{K}, (ij) \in \mathcal{E}^R, n \in \mathcal{N} \tag{A.9}$$

$$\tau_{k(ij)} \le t_{kn} + M(1 - w_{k(ij)n}) \qquad k \in \mathcal{K}, (ij) \in \mathcal{E}^R, n \in \mathcal{N} \tag{A.10}$$

$$\underline{T}_{(ij)} \le \tau_{k(ij)} \le \overline{T}_{(ij)} \qquad k \in \mathcal{K}, (ij) \in \mathcal{E}^R \tag{A.11}$$

$$t_{k(n+1)} \geq t_{kn} + \sum_{ij \in \mathcal{A}} T_{(ij)} x_{kijn} \qquad k \in \mathcal{K}, n \in \mathcal{N} \tag{A.12}$$

$$w_{k(ij)n} \leq x_{kijn} + x_{kjin} \qquad k \in \mathcal{K}^L, (ij) \in \mathcal{E}^R \tag{A.13}$$

$$n \in \mathcal{N} \tag{A.14}$$

$$2 \sum_{k \in \mathcal{K}^L} \sum_{n \in \mathcal{N}} w_{k(ij)n} + \sum_{k \in \mathcal{K}^S} \sum_{n \in \mathcal{N}} w_{k(ij)n} = 2 \qquad (ij) \in \mathcal{E}^R \cap \mathcal{E}^B \tag{A.15}$$

$$\sum_{k \in \mathcal{K}^S} \sum_{n \in \mathcal{N}} w_{k(ij)n} = 1 \qquad (ij) \in \mathcal{E}^R \cap (\mathcal{E} \setminus \mathcal{E}^B) \tag{A.16}$$

$$x_{kijn} \in \{0, 1\} \qquad k \in \mathcal{K},\ ij \in \mathcal{A},\ n \in \mathcal{N} \tag{A.17}$$

$$w_{k(ij)n} \in \{0, 1\} \qquad k \in \mathcal{K},\ (ij) \in \mathcal{E}^R,\ n \in \mathcal{N} \tag{A.18}$$

$$v_{(ij)} \in \{0, 1\} \qquad (ij) \in \mathcal{E},\ i < j \tag{A.19}$$

$$t_{kn} \geq 0 \qquad k \in \mathcal{K},\ n \in \mathcal{N} \tag{A.20}$$

$$\tau_{k(ij)} \geq 0 \qquad k \in \mathcal{K},\ (ij) \in \mathcal{E}^R \tag{A.21}$$

Appendix B

Compressed Vehicle Routing Model

Sets

$\mathcal{K}$	Set of vehicles, $k \in \mathcal{K}$
$\mathcal{K}^S$	Set of small vehicles, $\mathcal{K}^S \subseteq \mathcal{K}$
$\mathcal{K}^L$	Set of large vehicles, $\mathcal{K}^L \subseteq \mathcal{K}$
$\mathcal{A}$	Set of arcs, $(st) \in \mathcal{A}$
$\mathcal{S}$	Set of nodes, $s \in \mathcal{S}$
$\mathcal{P}$	Set of jobs ($p \in \mathcal{P}$
$\mathcal{P}^P$	Set of prioritized jobs, $s \in \mathcal{S}^P,\ \mathcal{S}^P \subseteq \mathcal{S}$
$\mathcal{S}_p$	Set of nodes corresponding to job $p, s \in \mathcal{S}_p,\ \mathcal{S}_p \subseteq \mathcal{S}$
$\mathcal{P}^B$	Set of broad jobs, $p \in \mathcal{P}^B,\ \mathcal{P}^B \subseteq \mathcal{P}$
$\mathcal{P}^R$	Set of requested jobs, $p \in \mathcal{P}^R,\ \mathcal{P}^R \subseteq \mathcal{P}$
$\mathcal{N}$	Set of all nodes including O_k and D_k

Parameters

L_p	Segment length of job (segment) p
L_{st}	Distance between node s and node t
S	Operating speed for all vehicles
T_{st}	Time usage of doing job s and traversing to job t, $T_{st} = \frac{L_{st}}{S}$
H_k	Available work hours for vehicle k
R_k	Range of vehicle k
O_k, D_k	Origin and destination node for vehicle k
W	Unit award for finishing a prioritized job
C	Penalty per time unit used
$\underline{T}_s, \overline{T}_s$	Lower and upper bound of time window for job $s \in \mathcal{S}^R$

Variables

x_{kst} $\begin{cases} 1, & \text{if vehicle } k \text{ travels from node } s \text{ to node } t \\ 0, & \text{otherwise.} \end{cases}$

v_p $\begin{cases} 1, & \text{if job } p \text{ fulfilled} \\ 0, & \text{otherwise.} \end{cases}$

t_{ks} Time vehicle k starts to service node s

$$\max z = \sum_{p \in \mathcal{P} \setminus \mathcal{P}^P} L_p v_p \; + \; W \sum_{p \in \mathcal{P}^P} L_p v_p - \; C \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{N}} \sum_{t \in \mathcal{N}} T_{st} x_{kst} \tag{B.1}$$

$$\sum_{t \in \mathcal{N}} x_{kO_k t} = 1 \qquad k \in \mathcal{K} \tag{B.2}$$

$$\sum_{s \in \mathcal{N}} x_{ksu} = \sum_{t \in \mathcal{N}} x_{kut} \qquad k \in \mathcal{K}, u \in \mathcal{S}, \tag{B.3}$$

$$\sum_{s \in \mathcal{N}} x_{ksD_k} = 1 \qquad k \in \mathcal{K} \tag{B.4}$$

$$v_p = 1 \qquad p \in \mathcal{P}^R \tag{B.5}$$

$$v_p = \sum_{k \in \mathcal{K}^S} \sum_{s \in \mathcal{S}_p} \sum_{t \in \mathcal{N}} x_{kst} \qquad p \in \mathcal{P} \setminus \mathcal{P}^B \tag{B.6}$$

$$v_p = \sum_{k \in \mathcal{K}^L} \sum_{s \in \mathcal{S}_p} \sum_{t \in \mathcal{N}} x_{kst} + \frac{1}{2} \sum_{k \in \mathcal{K}^S} \sum_{s \in \mathcal{S}_p} \sum_{t \in \mathcal{N}} x_{kst} \qquad p \in \mathcal{P}^B \tag{B.7}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{S}} T_{st} x_{kst} \leq H_k \qquad k \in \mathcal{K} \tag{B.8}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{S}} L_{st} x_{kst} \leq R_k \qquad k \in \mathcal{K} \tag{B.9}$$

$$t_{ks} + T_{st} - M_{st}(1 - x_{kst}) \leq t_{kt} \qquad k \in \mathcal{K}, s, t \in \mathcal{S} \tag{B.10}$$

$$\underline{T}_s \leq t_{ks} \leq \overline{T}_s \qquad k \in \mathcal{K}, p \in \mathcal{P}^R, s \in \mathcal{S}_p \tag{B.11}$$

$$x_{kst} \in \{0, 1\} \qquad k \in \mathcal{K}, \; s, t \in \mathcal{N} \tag{B.12}$$

$$v_p \in \{0, 1\} \qquad p \in \mathcal{P} \tag{B.13}$$

$$t_{ks} \geq 0 \qquad k \in \mathcal{K} \tag{B.14}$$

Appendix C

Additional Results from Computational Study

In this appendix, the underlying test results for the calibrated value of the parameter *nbThreads* is presented in Section C.1 and the heuristic gap results for Figure 8.2 is given in Section C.2.

C.1 nbThreads Parameter

Table C.1 shows the results with a 10 second time limit. Here, counterintuitively, the model returns better results when using 2 threads rather than 3 or 4, but with increasing time limits 4 threads outperforms options with fewer threads.

Table C.1: Gap sizes for thread count variations with 10 second runtime limit

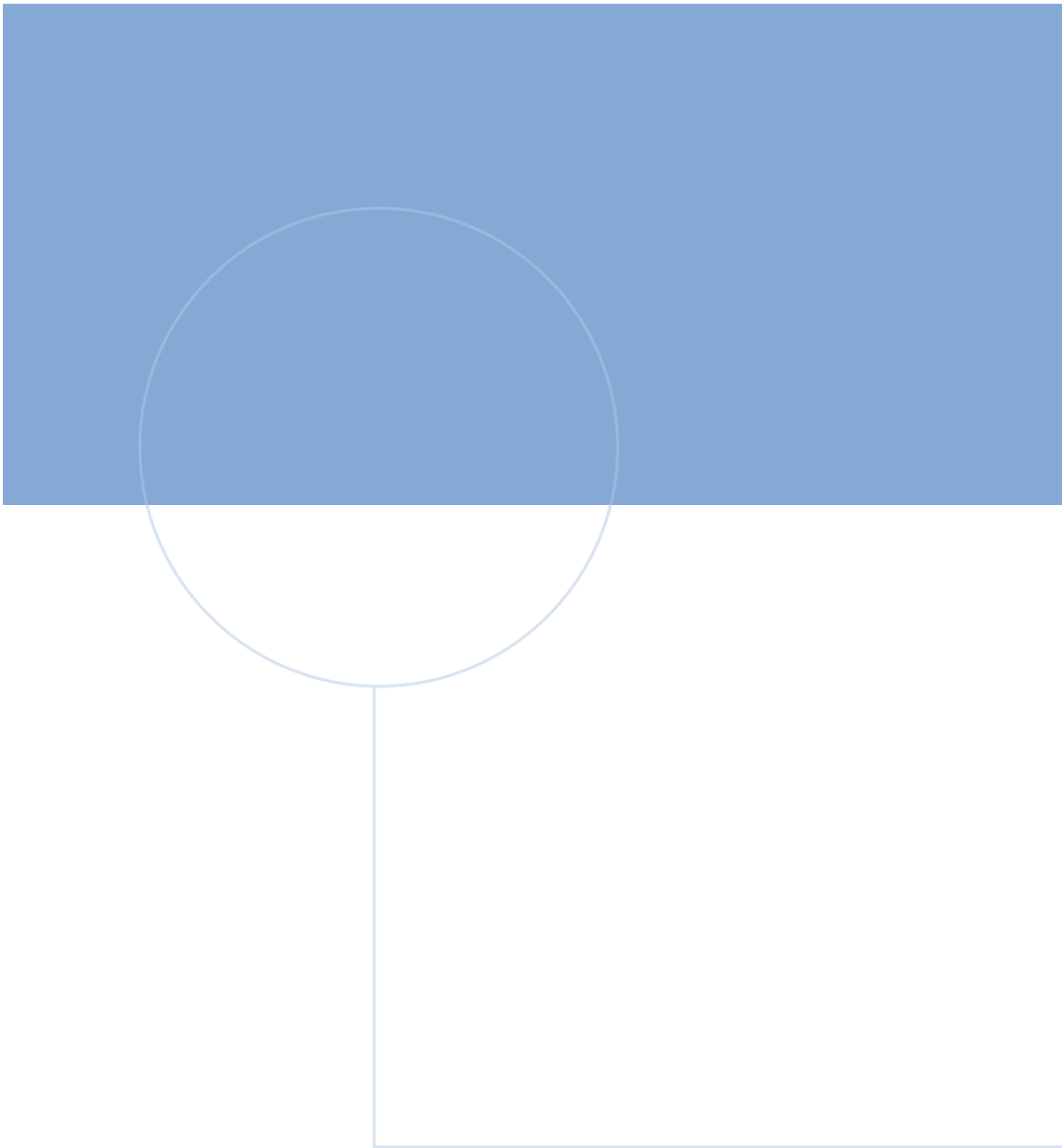
	1 core	**2 cores**	**3 cores**	**4 cores**
Instance	**Gap(%)**	**Gap(%)**	**Gap(%)**	**Gap(%)**
14nodes.txt	0.04	0.04	0.04	0.04
16nodes.txt	0.04	0.04	0.04	0.04
18nodes.txt	0.04	0.04	0.04	0.04
20nodes.txt	0.04	0.04	0.04	0.04
22nodes.txt	0.04	0.04	0.04	0.04
24nodes.txt	0.04	0.04	0.04	0.04
26nodes.txt	0.04	0.04	0.04	0.04
28nodes.txt	0.04	0.04	0.04	0.04
30nodes.txt	0.05	0.04	0.04	0.04
32nodes.txt	0.05	0.04	0.04	0.05
37nodes.txt	0.07	0.07	0.07	0.06
Bymarka.txt	89.82	88.81	95.21	103.16

C.2 Gap Sizes for Various Runtime Limits

Underlying run results for Figure 8.2.

Table C.2: Gap sizes for different runtime limits

	50nodes.txt		**75nodes.txt**		**100nodes.txt**		**Bymarka.txt**	
Runtime(s)	**Obj. val**	**Gap(%)**	**Obj. val**	**Gap(%)**	**Obj. val**	**Gap(%)**	**Obj. val**	**Gap(%)**
10	15836.2	0.07%	28801.4	19.91%	28676.5	161.27%	59450.2	124.17%
60	15836.7	0.07%	32019.0	7.86%	30537.2	145.35%	78141.2	70.55%
300	15837.3	0.06%	34512.7	0.06%	33588.6	123.06%	88921.7	49.87%
600	15837.3	0.06%	34512.7	0.06%	33588.6	123.06%	90961.9	46.51%
1800	15837.3	0.06%	34512.7	0.06%	34807.8	115.24%	93622.2	42.35%
3600	15837.3	0.06%	34512.7	0.06%	34807.8	115.24%	94062.1	41.68%



NTNU
Norwegian University of
Science and Technology