Optimizing routes for snow plowing in Trondheim using evolutionary algorithms

Simon Glisic Randby

Master of Science in Computer Science

Submission date:	July 2015
Supervisor:	Jo Skjermo, IDI
Co-supervisor:	Anders Kofod-Petersen, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

In this thesis route optimization for snow plowing in Trondheim municipality (Norway) is explored. It is modeled as a node, edge, and arc routing problem, which has been shown to be NP-hard. To generate the routes a memetic algorithm is used, which is a type of genetic algorithm, where instead of mutation a simple local search to improve each genome can be performed. The routes that are generated by this algorithm are optimized for being as short as possible weighted by speed limits, so that longer stretches that can be passed through faster are preferred over shorter ones with a lower speed limit.

When working on the routes, map data from the municipality and the Norwegian Public Roads Administration is used. To verify the routes that are generated, they are compared with how the routes that cover the same set of roads are currently being driven by processing both with the algorithms fitness function.

The results show that the algorithm is capable of finding better routes than the ones that are currently driven in terms of the used fitness function, but has never found an optimal solution, even in simple constructed test cases. The fitness function however suffers from that it is rather simple, and does not take into account important factors such as the width of the roads and what that means for how many times a given road has to be traversed for it to be sufficiently clean.

What that means is that while the routes generated are not good for practical use (even though the drivers seem to like the perspective a third party suggestion gives on their work). The routes show that it is possible to draw up routes with better fitness than what is being driven by humans, and that processing routes for areas with the same amount of roads as the city center in Trondheim is feasible.

Sammendrag (Abstract in Norwegian)

I denne masteroppgaven er ruteoptimalisering for snøbrøyting i Trondheim kommune (Norge) utforsket. Det er modellert som et node, kant, og rettet kant-ruting problem, som er kjent å være NP-hardt. For å generere rutene er en memetisk algoritme brukt, som er en type genetisk algoritme der man i stedet for mutasjon kan gjøre et enkelt lokalt søk for å forbedre hvert genom. Rutene som genereres med denne algoritmen er optimalisert for å være så korte som mulig vektet med fartsgrenser, slik at lengre strekninger som kan kjøres gjennom kjappere foretrekkes fremfor kortere med laver fartsgrense.

For å prosessere rutene er kartdata fra kommunen og Statens Vegvesen brukt. For å verifisere rutene som genereres, blir de sammenlignet med hvordan rutene som dekker det samme settet av veier blir kjørt nå av brøytebilførerene med algortimens fitness-funksjon.

Resultatene viser at algoritmen er i stand til å finne bedre ruter enn de som for tiden kjøres når de vurderes med fitness-funksjonen, men den har ikke funnet optimale løsninger, selv i enkle konstruerte testproblemer. Fitness-funksjonen lider imidlertid av at den er ganske simpel, og ikke tar hensyn til viktige variabler slik som bredden på veiene, og hva den har å si for hvor mange ganger en gitt vei må traverseres for at den skal bli tilstrekkelig fri for snø.

Hva dette betyr er at de generete rutene ikke er gode for praktisk bruk (selv om sjåførene synes å like perspektivet de gir på arbeidet deres). Men rutene viser at det er mulig å lage ruter med bedre fitness-verdi enn ruter kjørt av mennesker, og at å generere ruter for områder med samme mengde veier som Trondheim sentrum er gjennomførbart i praksis.

Preface

This masters thesis in computer science was written in the spring of 2015 at the Norwegian University of Science and Technology's Department of Computer and Information Science. The work has been supervised by Anders Kofod-Petersen and Jo Skjermo.

Acknowledgements:

Thanks to Trondheim municipality and Rolf Magne Brødreskift for supplying map data I used in this work, and their advice and feedback on how snow plowing is done in practice.

For his help on processing the maps of the Norwegian Public Roads Administration, thanks to Christoffer Viken.

Thanks to Magnus Solheim Thrap for the long conversations about the subject of this thesis, and for the help on researching how to extract data from the Norwegian Public Roads Administration.

Simon Randby
Trondheim, July 15, 2015

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

This thesis will deal with the issue of snow plowing. What it is, how it can be modeled and processed, and how to approach it in practice. The first chapter will start out with looking into the background of the issue, and why it is of interest to try to find good routes for snow plowing. Then the goal of the thesis, and the research questions utilized to reach it are going to be stated. After which there will be a brief section discussing how each research question is going to be handled. Finally, at the end of the chapter the layout for the rest of the thesis will be outlined.

1.1 Background and Motivation

In the municipality of Trondheim (in Norway) there are cold winters, with lots of snow. Removing the snow is an important logistical challenge for the municipality due to the impact it has on the usability of the road infrastructure. The size and complexity of the road network make it hard to be sure that planned routes are as short and efficient as possible. Additionally, the operations can in themselves prove an obstacle to traffic while carried out.

The municipality desires to minimize the resources spent and externalities caused by the snow plowing. To achieve these goals they want to see whether better routes than the ones currently being driven can be generated programmatically. In order to address the concern, this thesis will build upon the work in the pre-project leading up to it [Randby and Thrap, 2014]. The focus in the thesis will therefore be on whether better routes can be generated using memetic algorithms (MAs).

1.2 Goal and Research Questions

The main goal for this thesis can be stated as follows:

Goal Make a system that can generate optimal routes for snow plowing in Trondheim.

In the context of this thesis, *system* refers to a set of scripts and programs. Most importantly it encompasses the module that handles road-map data, and an MA that will process the map data to yield routes.

To reach this goal, the first research question that should be addressed is obtaining and presenting the relevant related work that gives the context for this thesis. Answering this might show whether optimal routes are possible to find at all. Additionally, the answer can show why an MA was chosen as the approach after the pre-project [Randby and Thrap, 2014].

The second research question that should be answered, given that an MA has been selected as the approach, is how to configure it so that it performs well. Being closely related to genetic algorithms (GAs), MAs will also perform horribly if initialized with the wrong parameters, and finding good ones is non-trivial. Therefore, not only the parameters themselves but also the process of obtaining them should be shown.

Finally, in addressing the goal there needs to be a research question that asks to determine whether the MA independently can optimize a route in Trondheim for snow plowing. Answering this should reveal whether the goal has been completely reached, what (if any?) areas are left to explore in the future work, and whether the MA performs satisfactorily for the application.

RQ1	Obtain and present the related work that gives the context for this thesis.
RQ2	Configure the memetic algorithm so that it performs well.
RQ3	Determine whether the memetic algorithm independently can find a route in Trondheim optimized for snow plowing.

Table 1.1: Research questions

1.3 Research Method

To address RQ1 large parts of Chapter 2 will be dedicated to presenting the relevant related work that was found in the pre-project of this thesis. This leads to our implementation in Chapter 3. RQ2 will be treated in Chapter 4, where the parameters are tuned in a set of quantitative experiments. Finally, RQ3 is considered in Chapter 5. In the chapter there will be a set of quantitative experiments to determine the quality of the routes produced by the MA. The generated routes will then be compared to the routes currently being driven. The comparison will be in terms of how the algorithm's fitness function perceives them. Once the comparisons are done, the best routes that have been found will be selected and shown to the municipality's domain experts. The assessment of the routes by the experts will then be presented in the thesis.

1.4 Thesis Structure

The first chapter after the introduction, named Theory, will start out by giving the necessary background on snow plowing. After that the related work will be presented. In the following chapter, Architecture and Implementation, details of design decisions and important details of the implementation will be given. The next chapter, Memetic Algorithm Configuration, will outline the process of how the MA was tuned and what parameters were found to give the best performance. In the Experiments and Results chapter that comes next, the performance of the MA will be tested, and the experimental setup for how it processed Trondheim will be shown. After which the resulting data will be presented and discussed. Finally, in the Conclusion and Evaluation chapter the entire process will be evaluated, and findings from the results presented. At the very end, the future work will outline how the work in this thesis can be extended.

Chapter 2

Theory

From meetings with the municipality and the Norwegian Public Roads Administration (NPRA) much background theory has come to light. Here the material is going to be presented. First in a section that covers theory on winter road maintenance. Then the rest of the chapter focus will be on routing problems, and how theory can be applied to the situation in Trondheim.

2.1 Winter Road Maintenance

In places where the temperature regularly goes below the freezing point of water, winter road maintenance has to be performed. Under such conditions, ice can grow on the roads, and they might become blocked by snow. The work done in ensuring that the roads have enough friction to be safe for use by pedestrians and vehicles, and clear enough of snow so that they may pass at all, is called winter road maintenance.

Figure 2.1 Snowplow in a Trondheim alley. Copyright Kent Syrstadløkk 2015, with permission.

The three most common ways of performing winter road maintenance are gritting, salting, and plowing. Gritting describes the act of spreading granular materials, such as pebbles or sand, on slippery surfaces so that they get more friction. It is often performed on walkways and smaller roads as less resource intensive way to temporarily make it safe for pedestrians and roads with light traffic. In practice, sand and combinations of sand and salt are preferred over pebbles when gritting.

Salting is similar to gritting in the sense that the same vehicles and rudimentary process is used to distribute it. However it acts in a different way than gritting, and there are other constraints to consider when using it. As opposed to gritting, salting is not done to increase the friction on top of the snow or ice that may be on the road. Instead, it is done to melt ice, or ensure that ice does not form in the first place. Thus, a desired amount of friction is achieved by keeping the road clean. The main challenge with salting and the reason that gritting may be preferred over salting in many situations is that salting on top of a layer of snow has disastrous results. One gets a slippery, gooey matter that cannot easily be removed from the road other than by scraping it off. Therefore, it is almost always performed as a preventative measure, usually right after a stretch of road has been plowed.

Snow plowing is when snow is cleared by pushing it aside. In daily life, various tools are used for clearing snow in this way, such as for instance general purpose or specialized snow shovels. However, for road maintenance vehicle mounted snowplows are the preferred appliance. Although they are not suitable for improving friction arising due to ice, snowplows are great at handling snow that is a direct obstacle to the traffic. Also, they can improve the friction by removing densely packed and polished tracks in the snow.

2.1.1 Organization of Winter Road Maintenance in Norway and Trondheim

In Norway, the winter road maintenance is a responsibility that is divided between the NPRA, the local municipalities, and the owners of private roads (which are responsible for maintaining their own roads). The NPRA has issued a handbook, R610 [Norwegian Public Roads Administration, 2014], which specifies the constraints on when and what kind of winter road maintenance should be performed, and what condition roads should be in after they have been serviced. In the handbook, they define five classes of winter maintenance standards for roads, as shown in Table 2.1.

Class Name	Class Description
Winter maintenance class A – DkA	Approved road condition is bare road (dry or wet).
Winter maintenance class B – DkB	Approved road condition is bare road (dry or wet), hard snow/ice is permissible outside rut[1]for limited periods.
Winter maintenance class C – DkC	Approved road condition is bare road (dry or wet) in mild periods and hard snow/ice in cold periods.
Winter maintenance class D – DkD	Approved road condition is hard snow/ice.
Winter maintenance class E – DkE	Approved road condition is hard snow/ice. Friction as low as 0.20 is acceptable. DkE may not be used for national roads.

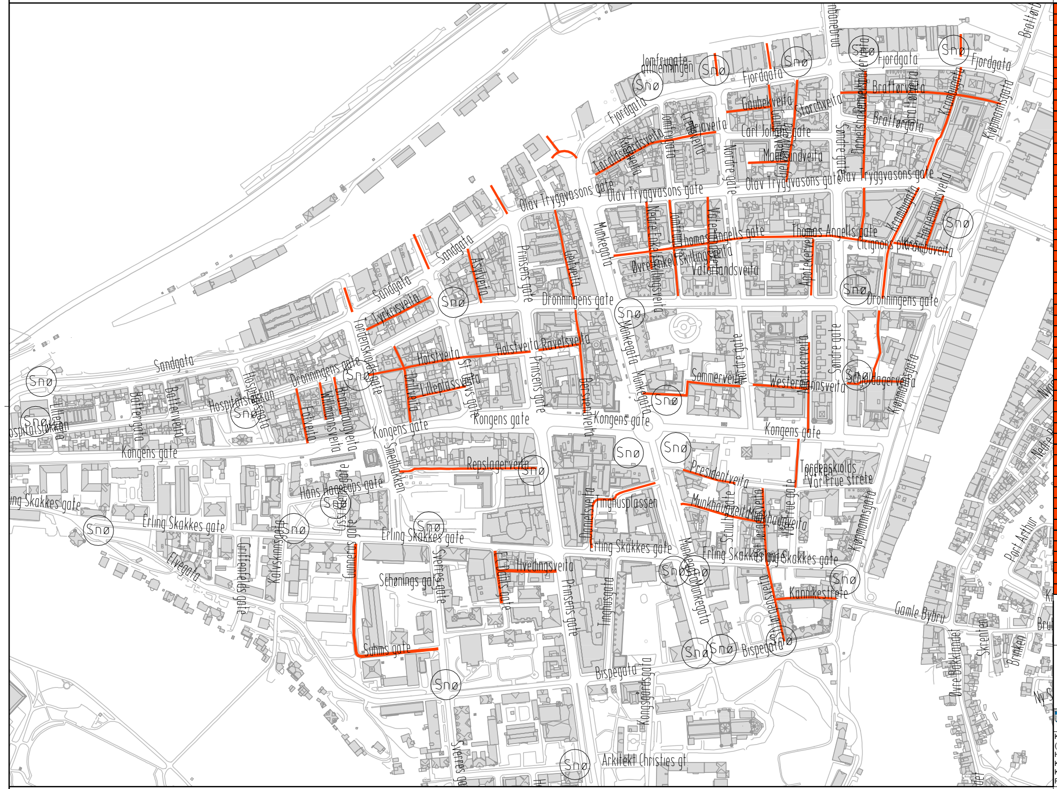
Table 2.1: Winter maintenance standard classes for roads specified by the NPRA (authors translation)

Besides being sorted into one of these maintenance classes, each road is also sorted into a functional class by the NPRA. In total, the NPRA operates with ten functional classes depending on factors such as the amount of daily traffic or size. However, Trondheim municipality only operates with six functional classes of roads in their maps. They are shown in the list below as translated by the author, and an example can be seen in Figure 2.2. Because the text will mainly be considering winter road maintenance from the municipality's perspective, those are the definitions that will be used further in this text.

[1]track in the road made by the wheels when many vehicles often pass

Figure 2.2 Map of a route driven in Trondheim. Copyright Trondheim Kommune 2015, with permission.

VEST - Sone TRONDHEIM SENTRUM **310174_KV_B_Midtbyen**



	Gatenavn	Lengde [km]
🟥	Apotekerveita	0,12
🟥	Apotekerveita fra Dronningens gt.	0,08
🟥	Asylveita	0,07
🟥	Bersvendveita	0,10
🟥	Brandhaugveita	0,05
🟥	Brattørveita	0,21
🟥	Danielsbakerveita	0,14
🟥	Danielveita	0,06
🟥	Drillveita	0,12
🟥	E.C Dahls gate	0,08
🟥	Fjordgata 12-16	0,03
🟥	Fjordgata 42-46	0,03
🟥	Fjordgata 54-56	0,03
🟥	Fotveita	0,07
🟥	Gaubekveita	0,08
🟥	Gaubekveita til Nordre gate	0,06
🟥	Geitveita	0,12
🟥	Gjelvangveita	0,14
🟥	Gunnerus gate	0,15
🟥	Holstveita	0,22
🟥	Hornemannsveita	0,08
🟥	Hvedningsveita	0,08
🟥	Jomfrugata	0,13
🟥	Kannikestrete	0,08
🟥	Kattveita	0,07
🟥	Kongens gate 16-18	0,06
🟥	Krambugata fra Dronningens gt.	0,32
🟥	Krambugata til Dronningens gt.	0,06
🟥	Krambuveita	0,07
🟥	Lilleplassveita	0,08
🟥	Moursundveita	0,05
🟥	Munkhaugveita	0,11
🟥	Nedre Enkeltskillingsveita	0,09
🟥	Peter Egges plass	0,03
🟥	Presidentveita	0,09
🟥	Prinsens gate 67	0,04
🟥	Ravelsveita	0,11
🟥	Ravnkloa	0,07
🟥	Repslagerveita	0,18
🟥	Sandgata 10-12	0,05
🟥	Sangata 24B	0,03
🟥	Schjoldagerveita	0,08
🟥	Sommerveita	0,09
🟥	St. Jørgensveita	0,21
🟥	Suhms gate	0,11
🟥	Taraldsgårdveita	0,10
🟥	Thomas Angells gate	0,37
🟥	Tinghusplassen	0,09
🟥	Tyrkrisveita	0,11
🟥	Vaterlandsveita	0,10
🟥	Vår Frue strete	0,17
🟥	Vår Frues gate	0,06
🟥	Westermannsveita	0,07
🟥	Willmannsveita	0,06
🟥	Ørjaveita	0,08

Total lengde **5,56 km**

N

Tegnet av: asplan viak

Trondheim bydrift *for miljø og trivsel*

Koordinatsystem: EUREF89 UTM- SONE 32 (22)
Høydereferanse: NN 2000
Kartuttrekk pr dato: 06122013
Kilde: Trondheim kommune
Revideringsansvarlig: Anders Svanekil

Evt. ekstra info.

TEGNFORKLARING

🟩	**Fylkesveg**	🟥	**Boliggate**	Snø	**Snøopplager**
🟦	**Kommunalveg**	🟦	**GS-veg**	Sand	**Strøsandlager**
		🟧	**Sykkelfelt**		

Målestokk: 1:4000 (A3)

Revisjonsdato	**01.10.2014**
Revisjonsnr.	**02**
Tegningsnr.	**310174**

- Riksveg (National road)
- Fylkesveg (County road)
- Kommunalveg/Hovedveg (Municipal road)
- Boliggate (Street)
- GS-veg (abbreviation for Gang- og Sykkelvei, translates to Footpaths and Cycle Paths)
- Sykkelfelt (Bicycle lane/path)

Out of these, the municipality is responsible for maintaining all, except national and county roads, for which the responsibility falls to the NPRA. As of February the 25th 2015, there are 212 km of municipal roads, 336 km of streets, and 180 km of footpaths and cycle paths that the municipality of Trondheim is responsible for maintaining during the winter ([Trondheim Kommune, 2015], only in Norwegian). To make servicing all these 728 km of roads feasible, they have divided the road network into several smaller zones.

An important factor in deciding the zones is what amount of work can be performed in one trip with the equipment. For gritting, the size of each zone may depend on how much sand can be carried before the vehicle has to return to a sand depot to refill. When plowing, the size of the zone might be restricted by how fast a vehicle can cover the entirety of it so that it is returned to the required condition before too much time has passed.

Another important thing that has been considered when creating the existing zones is what type of equipment is suitable for processing what kind of road. The large municipal roads might be suitable to service with a snowplow truck. Smaller alleys and streets, on the other hand, are better approached with smaller tractors that can fit through them. Therefore when looking at the zones it can be seen that many of them are inside the same area of the map, but containing different roads.

Something the zones in overlapping areas of the map do have in common are the places where the snow is to be stored and disposed of. In the countryside, the snow can often be dumped on the side of the road and left there. However, in cities, and densely populated areas, it is not so simple. There are various snow depots where the snow is moved for long-term storage, and the snow that gets cleared from the city centers proximity is put into the ocean at the harbor. However, all of the snow cannot be immediately moved to these locations while the roads are being cleared. Therefore, as seen on the municipality's maps, there are several areas where snow can be temporarily stored while the work is being performed.

To complete all of this work, the municipality has chosen to handle some of the zones themselves and set out the other to contractors. The municipality has a lot of the equipment needed for handling the biggest roads, and the intricate situation in the city center. To operate their equipment they use their own drivers. The contractors handling the remaining roads range from individuals that use their own vehicles (typically tractors), to specialized companies with several vehicles and drivers. All of the work is coordinated centrally, by the management at the municipality. The municipality organizes both the work carried out by the municipality's drivers and that done by the contractors. The management sets up the schedules for the maintenance work that needs to be regularly done and call out individual drivers when there has been a weather-incident.

Given this outline of how the municipality handles the snow removal part of winter road maintenance, how snow plowing can be modeled and solved needs to be addressed. The next section will explore the models used for similar problems to routing for snow plowing, and what methods have been used to try to find solutions to them.

2.2 The Node, Edge, and Arc Routing Problem

In the pre-project leading up to this thesis, a structured literature review is performed [Randby and Thrap, 2014]. The focus is on finding whether routing for snow plowing as in Trondheim is a known and solved problem. Moreover, in case it is, how should one go about trying to solve it.

In the structured literature review, it is found that the current take on problems like snow plowing has long roots. The simplest possible interpretation is that it is about traversing a graph G={N,E} in a certain way. Which would make it similar to the "Seven Bridges of Königsberg Problem", that is about finding a cycle that traverses the graph and visits all the edges exactly once. It has been solved by the Swiss mathematician Leonhard Euler in 1735 [Euler, 1741].

However, it is not until more than 200 years later (in 1962) that a graph traversal problem that is more relevant to snow plowing is proposed [Wøhlk, 2008]. The Chinese mathematician and former postman, Mei-Ko Kwan describes the problem of finding the shortest trip that visits all the edges in a graph at least once [Kwan, 1962]. It is the converse of the Vehicle Routing Problem (VRP), which is about visiting all the nodes of the graph at the lowest possible cost [Laporte, 1992]. The problem has become known as the Chinese Postman Problem (CPP).

Wøhlk [2008] presents that the CPP, if it contains only undirected edges or directed edges (arcs) (G={N,E} or G={N,A}), can be solved in polynomial time. However, if it is a mixed graph containing both (G={N,E,A}) it is NP-hard. The related VRP also being NP-hard has interesting implication for route optimization problem in general. Because many problems can be reduced to either the VRP or CPP, they are by extension NP-hard too. The NP-hardness makes heuristic approaches the most feasible way of obtaining good solutions. Coupled with the complexity of modeling real-life situations it is extremely difficult to improve the state of the art algorithm's and models at the same time.

Over the following decades, several variations on the CPP have been made to handle specific challenges. Here follow definitions of some of the most known formulations and how they are tied to snow plowing:

The Rural Postman Problem (RPP) arises when one wants to optimize solutions for when the postman does not have to visit all the edges, but only a subset of them [Pearn and Wu, 1995]. That is a situation that is likely to arise in snow plowing, where one for instance has to service all the roads in a city with a certain amount of traffic. In such a scenario, one can pass through the less used roads to move between the ones that have to be plowed.

The Mixed Chinese Postman Problem (MCPP) is the approach to the CPP where one looks at the problem in a mixed graph (with both arcs and edges) [Pearn and Liu, 1995]. It introduces a relevant constraint when considering practical applications, such as snow plowing, where the underlying graph represents a road network. Not only is it a very natural representation of a road network containing both bi-directional and one-way roads, it can also be used to handle cases such as intersections with forbidden turns.

The Hierarchical Chinese Postman Problem (HCPP) [Ghiani and Improta, 2000] emerges when one wants to investigate how solutions would change if one required some edges to be serviced before others. It is a situation that often arises in practice. For instance in snow plowing one might want to service a more used road before a less used road.

The Windy Postman Problem (WPP) is the attempt at accounting for that an edge can have a different cost each time you pass through it [Dussault et al., 2013]. This is a relevant consideration when snow plowing because simply passing through a road and plowing it takes a different amount of resources and time. Even passing through a road that has already been plowed and one that has not can have separate costs.

The k-postman Chinese Postman Problem (k-CPP) is the attempt at modeling a scenario where there is a depot node, and k postmen [Edmonds and Johnson, 1973]. It can also be interpreted as one postman that can do k trips of a certain cost. If implemented correctly it can be argued that splitted trips from the k-CPP can be used for sectoring the road network between different contractors.

The Min-Max k-CPP (MM k-CPP) takes it a step further and looks at minimizing the trip with the maximal cost of the k trips [Frederickson et al., 1976]. It is relevant for the snow plowing case when there is a large road network, and there is no single vehicle that is capable of service all of it in one go.

All these variations on the CPP are focused on general graphs, although they arguably can be applied to a broad spectrum of tasks in real-world road networks. However the Arc Routing Problem (ARP) has in the meantime been defined to describe problems centered around servicing arcs in transport networks [Eiselt et al., 1995]. The main difference from the various CPPs tends to be the more vehicle oriented terminology that is used, which makes it somewhat difficult to describe how they relate to one another. It can both be argued that the ARP is a more general version of the CPP with a slightly different terminology, or that it is a special applied case of the CPP.

In later years, the literature dealing with applied routing with the goal of visiting/servicing the arcs and/or edges in a graph has been favoring the ARPs terminology over that of the CPP. This widespread use makes it an interesting approach to the issue of routing for snow plowing.

Especially the variation known as the Capacitated Arc Routing Problem (CARP) [Ulusoy, 1985] has been widely studied. It has much in common with the k-CPP, because it deals with the case of having a set of identical vehicles with a limited capacity located at a depot. The task of which is to service the roads in a network in such a way that each road gets serviced by one vehicle once. However, like the k-CPP it suffers from that it is too general to give a good description of complex problems like snow plowing.

This lack of clarity has in turn given rise to definitions that try to include more constraints so that the problems one wants to solve are more accurately described. Within the ARP paradigm, the Extended Capacitated Arc Routing Problem (ECARP) has been defined to handle this [Lacomme et al., 2004]. It tries to take into account things such as that the problem might be in a mixed graph, the edges/arcs can have a varying cost, special constraints such as u-turns being forbidden in ordinary intersections, or a maximum limit on trip lengths.

The problem definition found that fits the case of Trondheim best, however, is the Node Edge and Arc Routing Problem (NEARP) [Prins and Bouchenoua, 2005]. It describes the problem of servicing a subset of the nodes, edges, and arcs in a mixed graph, with a homogeneous set of vehicles. Like in the k-CPP and CARP, each vehicle is capacitated in terms of demand it can service (cost it can handle). In the same way as the CARP the NEARP is NP-hard. The elements of the graph that require servicing have a fixed demand, and all of the arcs and edges have a traversal cost while nodes have no traversal cost (although required nodes still do have a servicing cost).

Because it combines the CARP, VRP, and the General Routing Problem, it takes into account many of the features of the underlying network desired to use for structuring the graph, such as the assumption of a mixed graph. Another feature that makes it a better model is that it considers both nodes, arcs and edges as elements of the graph that can need servicing. This assumption makes sense in a snow plowing setting, where not only roads and one-way roads but also intersections need to be plowed.

Now with the definition of what kind of problem snow plowing in Trondheim having been presented, how it can be solved should be considered.

2.3 Routing Algorithm's

In the structured literature review, several methods of solving routing problems came to light. Clearly, the most desirable outcome would be a solution that is known to be optimal. Moreover, for the polynomial time cases such as the plain CPP where all the edges have to be serviced by one postman, exact solutions methods exist. However, for the NP-hard cases like the CARP, exact solutions quickly become infeasible as problem sizes grow. Some of the best exact approaches have been reported to perform well for problems with up to 7 vehicles, 50 nodes, and 97 edges [Belenguer and Benavent, 2003].

These limitations of the exact methods have lead to efforts in pursuing other ways of finding solutions. Some have tried probabilistic approaches [Christiansen et al., 2009], but the majority of work has been done on heuristic methods. Simulated Annealing [Eglese, 1994] and Tabu search [Brandão and Eglese, 2008] have been used successfully to give good results. It has also been shown that A* can give approximate solutions [Rao et al., 2011]. However, this approach suffers from that there is a significant tradeoff between the quality of the solutions it finds and its capability of processing larger problems within reasonable time.

Other methods that have shown to be robust in terms of quality of solutions are Ant Colony Optimization (ACO) [Santos et al., 2010] and Greedy Randomized Adaptive Search

Procedure (GRASP) [Usberti et al., 2013]. Both yield solutions of similar quality to those found by Tabu Search, but ACO outperforms GRASP on running time.

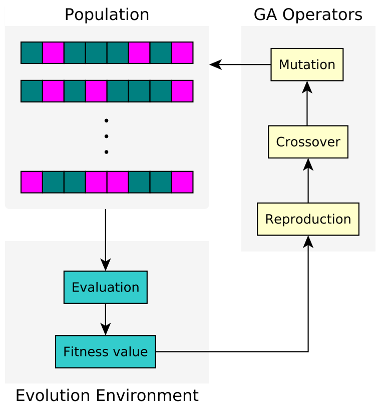
Lately, the most promising results have been given by Evolutionary Algorithms (EAs). GAs, a common type of EAs, have been shown to provide high-quality solutions for large instances of the CARP/NEARP [Lacomme et al., 2001], and generally to outperform other non-evolutionary methods [Wøhlk, 2008]. A further improvement on GAs has been made by MAs [Prins and Bouchenoua, 2005]. They have a reasonable running time, and they are capable of finding the highest quality solutions for the larger problem instances.

As such MAs can be considered a current state of the art approach to solving NEARP's, which is the reason they are chosen for the work with snow plowing in Trondheim in this thesis. The reasoning behind choosing an MA over an ACO or GRASP is a combination of two factors. Firstly, they all find the best known solutions so there no reason to prefer one of them based on quality. Secondly, in the pre-project [Randby and Thrap, 2014] more literature on MAs in relation to routing for winter road maintenance is found.

2.4 Evolutionary Algorithm's

Now that the reasons for using an EA has been laid out, what EAs are, and how they can be implemented should be discussed. EAs were one of the earliest bio-inspired algorithm's, whose outline were first discussed by Alan Turing in Turing [1950]. The idea is that EAs should mimic the biological processes in evolution, abstracting potential solutions to a population of individuals that have traits that are combined and altered through generations.

Figure 2.3 How evolution flows in genetic algorithms. Reproduced according to illustration in Liao and Sun [2001]



A common variant of EAs are GAs, as presented in Holland [1975]. An illustration of their general workflow can be found in Figure 2.3. In the paradigm of the GA, at the core of each individual there is a genome. This genome is an alternative representation of a solution to the problem one is trying to solve with the GA, which satisfies certain properties.

When there is a new generation, new individuals are made by combining existing members of the population. This combination is done by taking two or more individuals and combining their genomes into a new genome, which becomes the foundation for a new individual. This process is often called crossover. A genome must thus be constructed so that it can be merged with another genome in such a way that the output is a valid genome that represents a solution.

Each new genome, when created, has a chance of undergoing mutation. In practice, this often means that one of its components gets an altered value, or two components swap places in the genome, yielding a new altered genome. What this does is that it ensures that there is always some random variation in the population, making it less prone to becoming stuck in a local optimum.

It must also be possible to convert a genome to a solution of the problem the GA is trying to solve. The solution obtained from a genome is called an individual's phenotype. These phenotypes should take a reasonable time to generate, especially because they are often the foundation for calculating an individual's fitness.

The fitness can then be used to determine whether an optimal solution to the problem has been found. If not, for instance because it is unknown what a problems global optimal value is, the fitness is often used in determining what individuals are coupled when the next generation is created. It is also utilized to find what combination of pre-existing individuals and new individuals should make up the population at the end of the generation/beginning of the next generation.

To improve on GAs the concept of memetics can be introduced, turning them into MAs [Moscato et al., 1989]. Memetics is the term used for describing that individuals can be improved after being generated, mimicking the ability of real life organism in evolutionary systems to learn. In practice, this is often done by replacing the mutation step with doing a simple local search, such as hill climbing, to improve the genome [Lacomme et al., 2004].

With the genome being such a central piece in making an EA work, it is natural to let it be the first thing one considers when making a specialized EA for instance for snow plowing. Initially, the approach was to figure out what the genome was to represent. In the case of routing for snow plowing, a solution, and thus what the genome should represent is a route for a snow plow. However, what if there like in the k-CPP and CARP are several snow plows? Then the genome has to represent a set of trips for each vehicle. Ideally, the genome should be able to represent both these cases.

The solution found in the literature was to let the genome represent a single circular route, a grand tour of the underlying graph as done in Lacomme et al. [2001]. When dealing with multiple vehicles this tour is splitted into several shorter trips, one for each vehicle. Once the genome and the phenotype it represents is decided, one can begin to look into how the goodness of each individual should be determined, i.e., how to calculate the fitness.

2.4.1 Fitness for Snow Plowing

There are many things to keep in mind when trying to determine the fitness of a route in a complex operation like snow plowing. In interviews with the municipality and the NPRA, some of the most important factors that impact the execution of the plowing, and most likely should affect the goodness of a solution have been discussed.[2] They are shown in figures 2.4 and 2.5.

When one has gained an overview of what variables are required to find the fitness, the next task becomes setting up the calculation. The format of the benchmarks that have been chosen as the input format dictates how the values are supplied to the algorithm. Each edge and arc has an associated cost of passing through without doing work, the deadheading cost. Nodes have no deadheading cost. All the factors that affect traversing, simply passing through the graph, should, therefore, be stored in the deadheading cost.

[2]Meeting with Trondheim Municipality and the NPRA. Trondheim Bydrift, 3. etage, Tempeveien 22, 7031 Trondheim, Norway. 2014-09-02 09:00.

Figure 2.4 Factors arising from the environment

Amount of traffic. If there is much traffic, the plows should be expected to move slower.

Obstacles such as barriers and curbstones. Some barriers are designed so that they can be forced by sturdy vehicles such as snow plows moving slowly while other barriers cannot be passed. Other obstacles such as curbstones can be driven over by some vehicles, while others would be stopped or would damage the curbstone in passing over it.

Road marking and regulation. Some places snow plows could be exempt in the marking, allowing them to pass through and service roads other traffic would normally not be permitted to access. Road markings such as forbidden turns should also be taken into consideration when making and working on the graph based on the underlying road network.

Slope of the road. The slope of the road influences how quickly it can be worked on, and in some instances what type of equipment is suitable for the job (for instance whether tire chains have to be used).

Speed limit. The speed limit will determine how fast a road can be worked on, how suitable it is for accessing other parts of the road network, as well as its traffic pattern.

Weather – Quality of the snow. A lot of wet, heavy, and dense snow will require the vehicles to use more power than a dry, light, and powdery snow.

Weather – Slipperiness of the road. If there is much ice and the road is very slippery the vehicles can be expected to work slower.

Width of the road. Affects whether it can be done by a single vehicle in one pass or not, and what kind of vehicle is suitable.

Figure 2.5 Constraints on how the work should be carried out

Equipment – Amount of vehicles available. If there are more vehicles available they can split the road network between each other and the work can be carried out and completed faster.

Equipment – Types of vehicles available. Not all vehicles are suitable for plowing all roads. The larger vehicles used for plowing the main streets cannot enter and plow narrow alleys. Conversely, the vehicles used for plowing alleys or pavements are inefficient at clearing wider roads.

The weather. The NPRA has specified when maintenance should be done [Norwegian Public Roads Administration, 2014], and what shape the road should be in after the maintenance is done.

The type of road. Some roads have higher priorities than other roads.

Where the snow can be stored. Snow cannot always just be showed to the side out of the road. At times, there will be sidewalks or bicycle paths there that should not be blocked by the operation just to have to be services again later. There are pre-defined areas where the snow can be deposited.

All the elements in the graph that require servicing, including nodes, also have a servicing cost besides the deadheading cost. The servicing cost is the measure of how much it costs to perform work on the required element. Besides the servicing cost, each required element has value for demand. This value is used to indicate how much work needs to be done, in the case of snow plowing it can be interpreted as for instance the amount of snow that needs to be removed. The capacity of vehicles is expressed as the amount of demand they can service in one trip.

Once the problem has been reduced to a graph G={N,E,A} where each element has these properties, calculating the fitness of a trip becomes a matter of performing a summation. The fitness can be expressed as the sum of the deadheading and the servicing costs in the trip. This sum can in turn be broken into two new sums, the sum of all the deadheading costs in the trip, and the sum of the servicing costs. Finding the later is simple, the genome consists of the required elements in the trip, so one just has to iterate over them to find the sum of the servicing cost. If the genome encodes a tour of the required elements, this sum will remain constant for all trips.

The next step then becomes finding the costs of going between the required elements in the trip. In the process on should remember to add cost of going to the depot node and the first and last element, if there is a depot the trip has to start and end in. Due to having a mixed graph, the shortest path from one element to another is not necessarily the shortest path back again. Therefore, the shortest path between each required element has to be found with a shortest path algorithm. To find and keep track of these distances using an all-pairs shortest path algorithm and storing the output in a distance matrix was chosen.

2.4.2 All-Pairs Shortest Path

To verify the output, and to be able to display it on a map, a predecessor or successor matrix should be created. Such a matrix would allow generated trips to be recreated in their entirety. The two candidates algorithm's that would accommodate the needs and be efficient were Dijkstra's algorithm [Dijkstra, 1959] iterated once for each element or the Floyd-Warshall algorithm [Floyd, 1962]. The Floyd-Warshall algorithm was chosen because the code would end up being simpler and, therefore, more maintainable and easier to modify at a later time than an iterated Dijkstra's algorithm.

There is one thing that has to be consider before the Floyd-Warshall algorithm can be used to find the shortest distance between all the elements from the graph in the genome. It must be decided how the distance between nodes and edges or arcs should be handled. Usually, the resulting shortest path matrix gives the shortest path between nodes. However, with how the genome is constructed one will have to know the distance between not only nodes but also between other elements, such as the distance from a required edge to a required arc. A solution to this is to let the algorithm treat every element as it would usually treat nodes while at the same time treating every element as an arc.

If this is to make sense when using the distance matrix in the fitness calculations, some assumptions have to be made when the matrix is initially set up. Nodes have to add zero cost when moving between elements that are inbound to it or outbound from it. Furthermore, going from a node onto and arc or edge, and conversely going off an arc or edge to a node, should cost zero. However, with these assumptions the Floyd-Warshall algorithm should break down giving all paths a cost of zero because the shortest distance between all neighbors initially is zero.

This issue can be rectified when initializing the distance matrix with the distances between each pair of connected elements, by adding the cost of the destination element. The paths are then correctly found by the algorithm, but they end up being longer than they should be by the cost of the last destination element in the path. That is easy to fix once the algorithm is done finding the paths, by traversing the distance matrix and for each entry subtracting the cost of the destination element. Because this operation has a complexity of $O(|G|)$ it does not affect the overall complexity of performing the algorithm, which is still $O(|G|^3)$. The pseudocode for the resulting modified Floyd-Warshall algorithm is shown in Algorithm 1 in Appendix A.

With the all-pairs shortest path matrix, calculating the fitness of a trip given as a set of required elements from the graph (such as a genome) becomes a matter of traversing the required elements. When doing this, one adds their servicing costs and looks up and adds the distance between them from the distance matrix.

This summation works fine for obtaining the fitness of a genome as long as the genome is to be interpreted as a single giant trip of the entire graph. However, when there are several vehicles the work should be divided between, one first has to determine how the graph should be split between them in order to calculate the cost of each vehicles trip.

2.4.3 The Split Algorithm

The Split algorithm is used to take a trip in a graph and split it into smaller sub-trips that originate and end in a depot node [Ulusoy, 1985]. It generates the sub-trips such that none of them exceed the vehicles demand capacity, they service the required elements in the same order as in the supplied trip, and the resulting sub-trips are the shortest possible combination given these constraints. The pseudocode for the Split algorithm can be seen in Algorithm 2 in Appendix A. An example graph that can be used for input to the split algorithm is shown in Figure 2.6

The way the algorithm works to find the sub-trip of least cost that services the current element is as follows. For each required element in the supplied trip, it checks whether going from the depot, servicing the element, and going back is the least cost sub-trip. If so, the algorithm updates the list of predecessors indicating that going straight to the element from the depot is the best sub-trip for servicing it. At the same time, the elements value in the array of least costs of going to each element in the sub-trips is updated. The new value becomes the sum of the shortest paths of the previous splitted sub-trips, plus the new found cost of visiting this element in its own sub-trip.

After this, the algorithm tries to add one and one more required element to the current splitted sub-trip until the next added element would exceed the vehicles capacity for servicing demand. To do the check, the algorithm first removes the cost of going from the last element in the currently splitted trip back to the depot from the end of the trip. Then the cost of going to the next required element, and from that element to the depot, is added to the cost. If this updated cost is lower than the currently lowest known cost for reaching the next element, and the demand load is manageable, the costs and predecessors arrays are updated. The next element's cost in the costs array is updated to the current cost. At the

Figure 2.6 Graph instance processed by the split algorithm. Edge traversal cost is given in the parentheses, and the required edges are marked with bold. The vehicle demand capacity is 8.

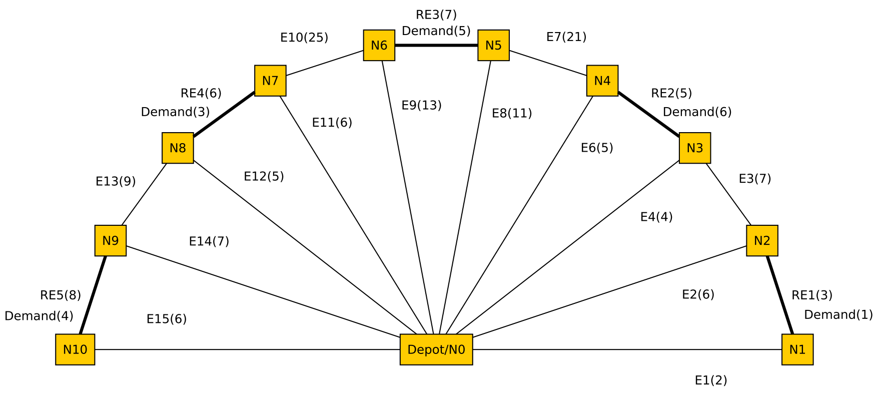


Figure 2.7 The auxiliary graph built by split to process the graph shown in Figure 2.6. The least cost of processing each required edge found is shown in parentheses in their respective nodes in the auxiliary graph. The least-cost path through all required elements is highlighted with bold arrows.

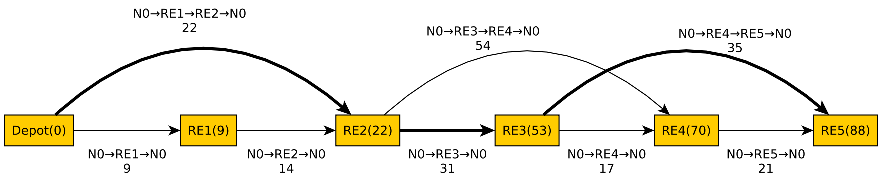
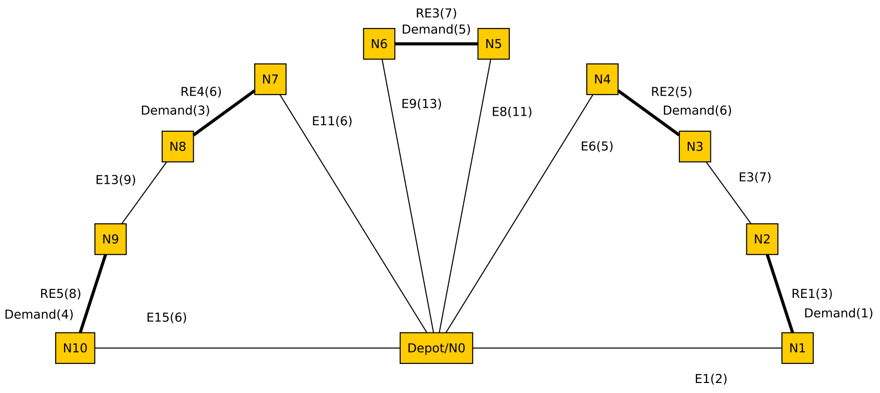


Figure 2.8 The trips generated from processing the graph in Figure 2.6 with the split algorithm.



same time, its entry in the predecessors array is set to the index of the node at the beginning of the sub-trip currently being considered.

This process of adding an element at a time to existing or new trips and checking whether the sum of demand is permissible and the length of the trip can be taught of as generating an auxiliary graph. If one uses the graph in Figure 2.6 as an example, the resulting auxiliary graph would be the one shown in Figure 2.7.

For the purposes of finding the fitness value, the algorithm makes it as simple as looking at the last entry in the costs array. It is the sum of the costs of the shortest possible sub-trips leading up to the sub-trip containing the last element. Because the cost of this sub-trip also contains the last element and its cost, it is the cost of traversing all the sub-trips.

The retrieval of the resulting trips is shown in Algorithm 3 in Appendix A.3. This algorithm works by starting with the last node of the last splitted sub-trip and iterating over the predecessor array obtained from the Split algorithm. That gives it a worst case running time of $O(|\text{required elements}|)$. However, that is still less than the asymptotical worst case of the split algorithm that is $O(|\text{required elements}|^2)$. An example of trips that would be retrieved from the graph Figure 2.6 illustrates is shown in Figure 2.8.

CHAPTER 3

ARCHITECTURE AND IMPLEMENTATION

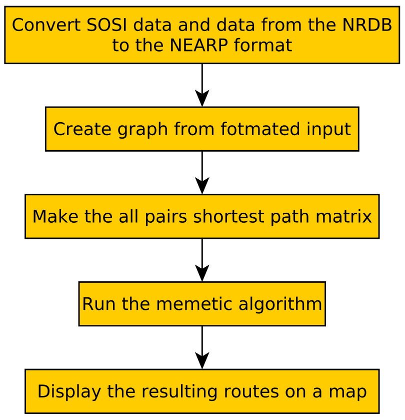
In the previous chapter, the foundation for implementing a system that does route optimization for snow plowing has been outlined. It has been shown why the NEARP has been chosen as the underlying model. Further, it was argued why MAs are a good approach to the problem. Then issues relating to the combination of the snow plowing problem and MAs solving the NEARP were discussed.

Now with the background of our implementation sorted out, the architecture should be addressed. The main flow of the system is illustrated in Figure 3.1. This chapter will consists of three main parts. It will start out by discussing the data sources used in our implementation. Then it will move on to discuss how the input and output is done. Finally, there is going to be a section that treats the implementation details of the MA itself.

3.1 Data Sources

In Figure 3.1 we can see that the first thing to happen is that input data is converted to the NEARP format [SINTEF, 2012c] for use by the MA. The data comes from two main sources. The first one is the municipality's map data, supplied in files adhering to the Norwegian SOSI-Standard for map data [The Norwegian Mapping Authority, 2013]. Most importantly, the maps contain information about what set of roads and intersections make up each existing route. These roads can then be used as the required elements of a route the MA is supposed to optimize the servicing of. The sets of roads can also be utilized for evaluating how the drivers currently drive their routes with the fitness function used by the MA. This fitness can then be used to evaluate how good the results of the MA are by comparing it to the best fitness the MA finds for the same set of roads.

Figure 3.1 Overview of the systems Workflow


Convert SOSI data and data from the NRDB to the NEARP format
Create graph from fotmated input
Make the all pairs shortest path matrix
Run the memetic algorithm
Display the resulting routes on a map

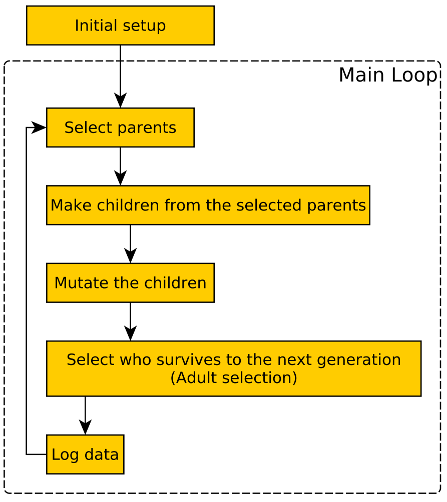
Another important piece of data that the municipality's maps contains are modifications to the infrastructure made by and maintained by the municipality. The most important example would be barriers. As shown in Figure 2.4 some of the barriers can be passed by snow plowing vehicles, while other barriers can not. The municipality's maps detail which can and which can not be passed.

What other information about the infrastructure that is required but not contained in the municipality's data set can be found in the NPRAs data. Their database of roads, the NRDB, contains highly detailed information about the road network. It covers physical features such as length, width, and curvature, as well as meta-information such as speed limits, placement of signs, and where pedestrian crossings are located.

3.2 Formatting Input and Output

The process of making an input file based on all this data is done in the following way. First the road networks from the municipality's maps and the NRDB are combined into a single dataset with PostGIS. The combination relies on matching the geometries of items in the maps, combining items into a new one if they are within a certain distance of each other, otherwise copying them in. For most parts, this works well, but there are some roads and required elements that go missing in the process. One reason for the missing pieces is how the merging of the municipality's data and the NRDB data is done. Because the merging works by combining elements closer than five meters apart in the data into one, sometimes roads that should have been separate are merged. The missing pieces are however easily found on inspection and drawn in. Nevertheless, there might be errors in the data set used further that have not been discovered at the time of writing.

Figure 3.2 Memetic Algorithm Flowchart


Initial setup
Main Loop
Select parents
Make children from the selected parents
Mutate the children
Select who survives to the next generation (Adult selection)
Log data

This set is then converted into a graph with nodes, edges, and arcs. The intersections become nodes, and the ones that are part of a route in the municipality's maps are labeled as required. One-way streets are converted to arcs and other roads to edges. When working with routes for car roads, bicycle paths, pavements, and sidewalks are ignored. The main reason for ignoring them is because they in most cases cannot be utilized by the vehicle's operation on the car roads for passing through. Also when finding routes for plowing car roads they do not have to be considered for servicing.

However, even though the other types of roads do not have to be serviced, in some cases they should be considered wile servicing the car roads. For instance, when a sidewalk and adjacent car road are both plowed the order matters. When the road is plowed, snow is going to move to the side and onto the sidewalk. Therefore, if each of the routes are to be driven only once the road should be serviced first, then the sidewalk. With that ordering, the snow will move off the road to the sidewalk and then off the sidewalk. Otherwise, the sidewalk would just end up filled with snow again after it has been cleared. It is however chosen to ignore the interaction between servicing adjacent routes in this work. The reasoning behind doing so is that it simplifies the model and allows for working on each route individually.

When making arcs and edges, the ones that are part of a route are labeled as required, just like the nodes. The arcs and edges are then assigned a traversal cost, that describes how much resources are spent passing through them without performing work on them. For car roads, this is set to be: Traversal cost $= len \times (150 - lim)$. In the equation len is the length of the road described by the arc or edge, and lim is the speed limit on the stretch. For other kinds of underlying roads or paths, the traversal cost is set to just the length. Subtracting the speed limit from 150 before multiplying is done so that faster roads get priority due to that the fitness is considered better when lower.

After the traversal costs are determined for the arcs and edges, the required elements get their demands and servicing costs set. The demand of each element, in the case of snow plowing representing the amount of snow that each vehicle should handle the removal of, is set to be the length of the element. The same is done for the servicing cost. The main reason for choosing the length for these values is that the length scales with the amount of work that needs to be done, and resources needed to carry out the work. Additionally the servicing cost amounts to the same for each trip matter what order it is processed in, so it will not affect the fitness of the individuals.

This process for converting to and from the NEARP format for the MA has been designed in collaboration with Magnus Solheim Thrap and Christoffer Viken. We all took part in deciding how the data should be mapped to the NEARP format. Thrap found how to extract certain features from the data in the NRDB. Viken has been able to combine the data from the NRDB and the municipality in PostGIS. Then the omissions from the conversion are remedied in PostGIS, exported as CSV files and converted to the NEARP format with a script written in Python. Finally, the resulting routes are converted back to CSV, and Viken has found a way to draw them on a map using PostGIS.

3.3 Memetic Algorithm Implementation

Now when the implementation of how the MA takes input has been outlined, the architecture of the MA itself can be discussed. The workflow in the MA is shown in Figure 3.2. From the figure, it can observed that the first step is generating an initial population. This population is either generated completely at random, or seeded from known routes. The seeding can be done for instance when working with existing routes, to see whether the MA can improve on them, or if routes with better fitness can be generated faster.

3.3.1 Parent Selection

Once the initial population has been generated, the MA enters its main loop. Here it will first select which individuals get to breed in the parent selection. In our implementation, there are three possible ways of doing the parent selection. Uniform selection, fitness proportionate selection, and tournament selection. All the parent selections are implemented with replacement, but with the constraint that no pair of parents can have two equal individuals. That is, one individual can have several mates each generation, but cannot mate with itself.

If using uniform selection, the parents are selected completely at random from the adult population, i.e., they have a uniform probability of being selected. The fitness proportionate selection is biased towards picking individuals with better fitness for being parents, by giving each individual a probability of being chosen that is $P(\text{selecting individual}) = \frac{\text{the fitness of the individual}}{\text{sum of fitnesses of all adult individuals}}$.

Tournament selection picks individuals for parenthood by making groups of individuals it selects from. For each parent it selects, it creates a group of fixed size that is determined by the operator on startup, which is sorted by fitness. Then, with a chosen probability P it selects the best individual. Should the best individual not be selected, it goes on to the second best and determines whether to select it with the same probability P. The tournament with the individuals stacked against each other continues until a parent is selected, or all individuals other than the worst in the group have failed. In the event that all better individuals in the group fail to be selected, the worst one is selected. This gives the best individual a probability of P of being selected, the worst one $(1-P)^{\text{tournament group size}-1}$, and the ones in between $(1-P)^n \times P$, where n is the number of better individuals in the tournament group.

3.3.2 Child Creation

When the parent selection has been done, the selected pairs of parents are used for generating children with the crossover procedure, which is illustrated in Figure 3.3. The crossover works by first selecting two random crossover points. Then, for each parent, every element in the genome that is between these points is copied to a child genome. The rest of the child genome that is not between the crossover points is then filled with the remaining required elements. They are inserted in the same order as they appear in the other parent after the second crossover point. This process ensures the child genomes maintain their properties as circular representations of all the required elements from the graph, and that they inherit traits from both parents.

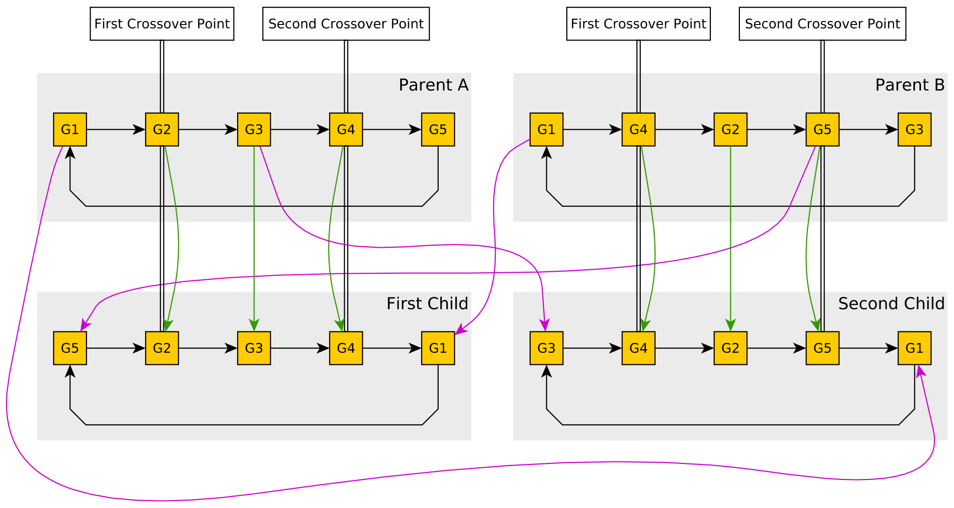
After all the children have been generated, the MA mutates them. In our implementation, this step is done by either swapping two elements in their genomes at random or performing a memetic improvement of the genomes. The memetic improvement is done by swapping two and two elements in the genome, and evaluating the fitness of the outcome. This process is repeated until the swap results in an improvement fitness, or all swaps have been tried.

3.3.3 Adult Selection

Then, when the children have been made and mutated, the adult selection is performed. It is the process where one determines what individuals from the adult population and children are kept and used as adults in the next generation. There are four kinds of adult selection methods in our implementation, full replacement, elitist mixing, random mixing, and overproduction. With full replacement, there are made exactly as many children as the size of the population is at the start of each generation. In the transition, all the previous individuals are discarded, and the children become the new population.

If doing elitist mixing, the number of children can be arbitrarily chosen. When going to the next generation, the children and adults are combined into one large pool. Then the

Figure 3.3 Crossover with two parents and two crossover points



individuals with the best fitnesses get the slots in the new population, and all the other are discarded. Random mixing works much in the same way as elitist mixing, except that the individuals are chosen from the pool at random, instead of by fitness. The approach when using overproduction on the other hand is to create more children than there are individuals at the start of the next generation. In the transition to the next generation, the best children are used, and the rest of them are discarded together with the members of the old generation.

3.3.4 Generation Transition

After the adult selection has been done, the worst individual of the new generation is removed, and the previous best is inserted in its place. Then the halting conditions for the MA are checked, and if they are met the algorithm terminates. Of the two conditions that are checked for, the first one is whether there have been no improvements in the best individual found for a number of generations which is determined on startup. The second is whether a fixed number of generations has elapsed since the MA was started. Once it has terminated, the output can be drawn on a map.

Chapter 4

Memetic Algorithm Configuration

The first thing to consider after having implemented the MA is figuring out what parameters should be used for it to perform well. Depending on the configuration, the MA's performance can be changed dramatically, and the quality of the results may vary. This chapter will look into the background for doing the configuration, outline the experimental plan for doing the tuning, show the results, and present a brief discussion on the outcome.

4.1 Parameter Tuning

Here the outline of how the parameters that should be used when running the MA will be discussed. There are several features that could be tweaked or enabled and disabled to alter how the MA works. For instance the population size and number of parents that get to reproduce each generation are values that can be adjusted. However, whether to use random mutation or memetic optimization is a choice of whether to use one of two implementations.

Some combinations of the parameters yield higher quality results faster (both in terms of actual time spent and number of generations passed before the output stabilizes, i.e., the program settles at a local optimum). Therefore, it is interesting to determine the best combination of parameters.

The following parameters are the ones that the user can set:

- How the genomes that get to produce children are selected.
- How the next generation is generated from the current set of adults and children.
- Whether the fitness should evaluate one grand tour for one vehicle or consider a case where the given area should be divided among a set of vehicles.
- The maximal number of generations the algorithm should run for before terminating.
- If the algorithm gets stuck in a local optimum, for how many generations should it continue trying to get out of there before acknowledging that it is stuck and just return the best answer it has.
- The number of individuals to have in the population at the beginning of a generation.
- How many individuals to select from the population that gets to mate each generation.
- How many pairs of parents to make from the selected parents.
- If parents are selected tournament-style, what size (in terms of number of individuals) should each tournament group be.

- If parents are selected tournament-style, what should be the probability of selecting the best individual from each tournament group each iteration of the tournament.

4.2 Experimental Plan

For the results to be as describing as possible, these tests should be run with the same input that one wishes to apply the MA to. That has the advantage of the results giving clear indications as to the computational time required, and one could come across a very good solution in the process. Also, if the structure of the search space influences the choice of parameters, it would become clear at this point. In this case, the search space is the mapping of fitnesses to different permutations of visiting required elements in the underlying graph.

In the light of the research questions (especially RQ3) that would mean running the tests on a subsection of Trondheim with data from the NRDB. In practice, this poses several challenges. First of all, at the point in time where these tests could be done, the module structuring data from the NRDB was not available. Second, if the assumption of that the structure of the search space might influence the outcome, it could be argued that different parts of Trondheim can be significantly different from each other. Which implies that for each section of the map that one wants to process, the parameters should be individually optimized. This choice would in turn lead to the dilemma of selecting the most representative part of the map if such a thing is even possible. Third, if using real world data, there would be no way of knowing whether an optimal solution has been found, or evaluating the output beyond whether it is completely outlandish or somewhat reasonable. Fourth, it should be easy for a human to verify the output, both for correctness and whether the calculated fitness values make any sense.

Hence, to tackle the challenge of configuring the MA, a new data set is produced for the tuning. It can be found as a file named *circle_test.dat* in the supplementary materials. The structure of the created graph is that of a directed cycle (as seen in Figure 4.1). All of the arcs are required elements, which has several good qualities given the criteria above. First of all it is easy for a human to verify the output. There is only one global optimum, which is readily identifiable, visiting each node and arc in order exactly once. Furthermore, its fitness is trivial to calculate, as it is the sum of the traversal and servicing costs of all the elements exactly once. Any other ordering of the elements would lead to the circle being traversed more than once. Moreover, all the possible ways to order the required elements the MA could make would have a distinct fitness.

An example of an ideal solution to the circle test shown in Figure 4.1 would be the genome which is illustrated in Figure 4.2, which has a fitness value of 60.

Figure 4.1 Circle Test Graph example with no arc labels or costs drawn in

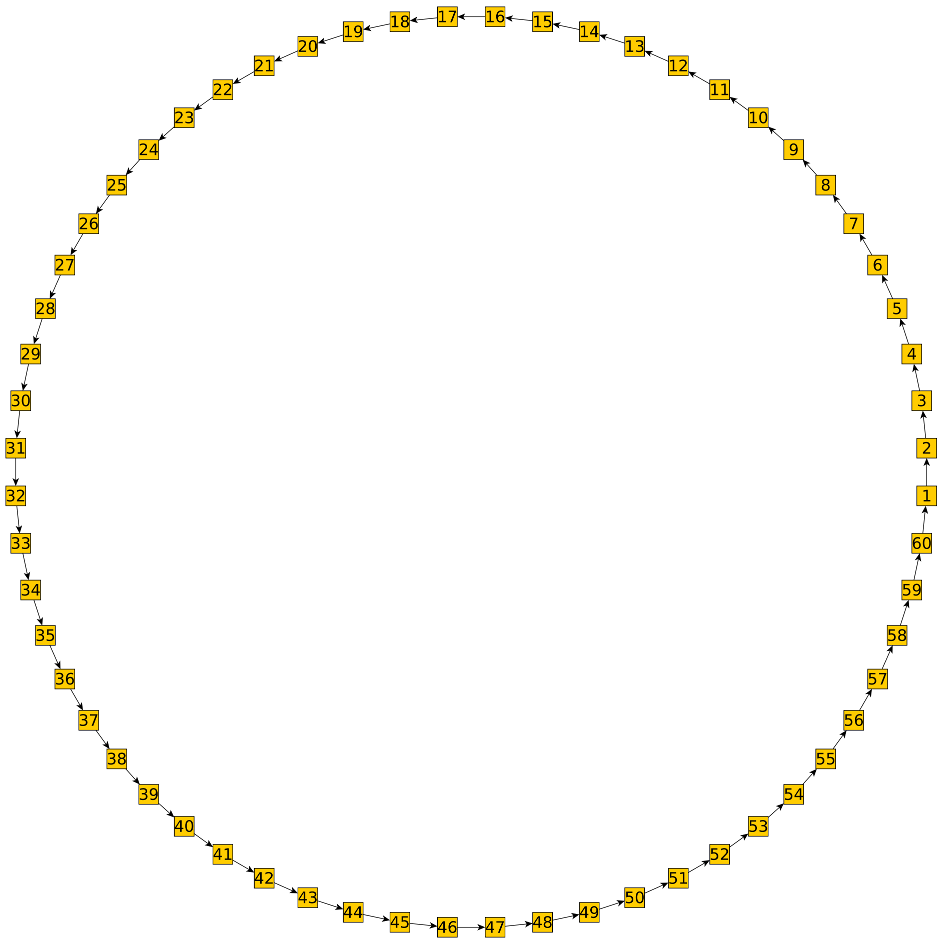
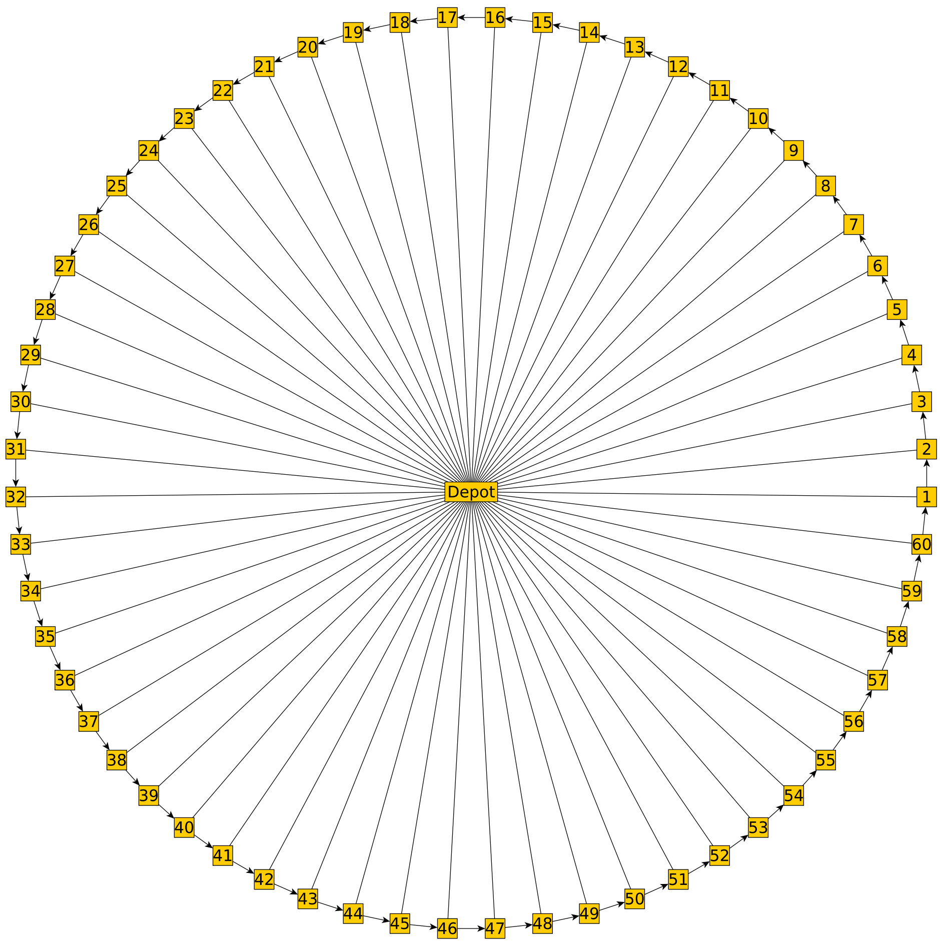


Figure 4.2 Best possible genome for the supplied Circle Test example

A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, A32, A33, A34, A35, A36, A37, A38, A39, A40, A41, A42, A43, A44, A45, A46, A47, A48, A49, A50, A51, A52, A53, A54, A55, A56, A57, A58, A59, A60

While using the circle test data set might not give an exact answer to what configuration of the parameters is optimal, it still gives a usable prediction of what settings work in a general case with our implementation. Another limitation that is not addressed at this point is that it only makes sense to evaluate the test case as giant round trip completed by one vehicle. It could be adopted to facilitate testing splitting the graph between several vehicles by adding a depot node. It would have to be in the center, and edges that are not required to service would have to be made from it to all the nodes (as illustrated in Figure 4.3). However that would add much complexity, and it would not be feasible to investigate it further at this point.

Figure 4.3 Alternative Circle Test Graph example with depot node and no arc labels or costs drawn in



4.3 Experimental Setup

The tuning of the MA is split into three parts; determining the population size, parent selection, and adult selection. For each part, a set of parameters to be tested are selected based on the author's previous experience with tuning EAs and the size of the problem at hand. Then each configuration is to be run 30 times to gather data about the best results found with the configuration and the population averages. It is decided to let each configuration run for exactly 100000 generations, to make the results as comparable as possible. For each run, all the genotypes will be initialized randomly. Each 16th generation the best fitness of the population, the average fitness of the population, and the standard deviation of the fitness of the population at that point is to be recorded.

For determining the population size, it was decided to try population sizes corresponding to 10%, 50%, 100%, and 200% of the genome length. When testing it, uniform selection is used for parent selection and full generational replacement for the adult selection.

When trying different settings for the parent selection the size of the population is set at 200% of the genome length. For the adult selection full generational replacement is used, as that allows for reuse of the results from the population size run for uniform selection.

When trying to find the best way to do the adult selection a population size of 200% of the genome length, and fitness proportionate selection as the parent selection are used. When testing overproduction it is decided to make as many pairs as there are individuals in the population, i.e., it will be produced twice as many children as there can be individuals surviving to next generation.

During the determination of whether random mutation or using hill climbing for mutation, a population size of 200% of the genome length is used. The parent selection used is fitness proportionate selection, and the adult selection method is overproduction.

4.4 Results

Now that the experimental plan has been outlined, in this section the results will be presented. First the initial results for the population size experiment will be shown. Then the outcome of using different parameters for the parent selection is going to be laid out. After that, the experimental results when testing different configurations for mutation are going to be presented. Finally, the results from a follow-up experiment on the population size will be shown. All the relevant graphs can be found in Appendix B.

4.4.1 Population Size

When observing the results from the population size test runs, it can be noticed that that a lower population size yields better results. Figure B.1 in Appendix B.1 shows that the larger populations perform noticeably worse. The largest population (120 individuals) gets a fitness value of around 700 by the end of the runs while the smallest population size (6 individuals) ends up with a fitness of about 600.

A possible reason for this is that uniform selection combined with full generational replacement is not biased in any way. The lack of steering makes it very random in terms of that the fitness of the individuals in the existing population will not affect future generations.

The population will, however, be influenced by the previous generations in terms of that their genomes will be combinations of previous genomes with mutations.

A possible explanation then could be in that the best individual from each generation is kept and is injected in place of the worst one at the end of each generation. The insertion will have an enormous impact on a tiny population of only six individuals, effectively dragging the population towards the currently best. However, as population sizes increase the best individual, and its offspring have less impact on the average of the population.

This effect should be offset by introducing bias in the population towards that the individuals with better fitness have a better chance at impacting future generations. However, to be certain that this is a quirk of combining uniform selection, full generational mixing, and the constant reintroduction of the previous best single individual at the cost of the currently worst, more testing should be done. Therefore, a control experiment for parent size using another form of parent selection and generational replacement is set up.

4.4.2 Parent Selection

In Figure B.4 in Appendix B.2 it can be seen that fitness proportionate selection is clearly better than the tournament selection, which in turn finds better results on average than uniform selection. Their final fitnesses are just below 500, a bit above 500, and around 700 respectively. The averages fitness of each populations average seems to decrease quickly before beginning to decrease more slowly for tournament and fitness proportionate selection while the uniform selection shows only a very slow improvement over time.

4.4.3 Adult Selection

When looking at the results for the adult selection in Figure B.7 in Appendix B.3, the most striking thing is the completely flat curve for the elitist mixing average best. It indicates that there on average is no improvement at all in the best solutions found when using it as adult selection. This can be explained by observing that the average standard deviation in each population is 0 (Figure B.9, Appendix B.3). Coupled with that the average fitness of the population being stuck on a single value in Figure B.8 from Appendix B.3, it indicates that all the individuals in the population have the same fitness. What seems to have happened is that the population has become stuck in a local optimum, and all the individuals it produces are exact copies of the currently best found solution.

When looking for what adult selection gives the best output according to these results both overproduction and random mixing give better results than elitist mixing and overproduction yields slightly better results than random mixing.

4.4.4 Mutation Types

As seen from figures B.10 and B.11 in Appendix B.4 the memetic improvement yields better results than when doing random mutation. Adding a memetically improved child to the children before the adult selection seems to make little difference though it reaches a somewhat better solutions a little bit earlier than when not adding it. For both memetic approaches the fitness value tends to get as good as just below 400 in the long run, as seen in Figure B.10 from Appendix B.4.

Figure B.12 in Appendix B.4 shows that the diversity of the population is somewhat lower when doing memetic improvement instead of random mutation.

4.4.5 Population Size Control

Due to the unexpected results in the population size tests (Chapter 4.4.1), it is decided to look into whether the effects of the population size that appeared could be explained by the lack of elitism. Especially because the choice of using uniform parent selection and full generational replacement removes any form of elitism that normally arises in the GA/MA process. A control test is set up, where again 30 populations are to be run, but this time with fitness scaling selection and overproduction with twice as many children as the population size.

The results obtained are according to the predictions about a population yielding better output as its size increases. Also, the results indicate that the behavior when not guiding the development of the population in any direction actively is heavily influenced by the reuse of the best individual.

A larger population gives better results, but as the populations grow very large, there are diminishing returns on expanding it further. As seen in Figure B.13 in Appendix B.5 the largest population gives only marginally better results than the second largest population. Which in turn gives somewhat better results than the third largest population, which gives a lot better results than the smallest population.

4.5 Summary of Parameter Tuning Results

The above experiments lead us to the following conclusions about the parameters for the MA. Chapter 4.4.2 indicates that the best parent selection method for the algorithm is fitness proportionate selection. As for the adult selection, Chapter 4.4.3 shows that overproduction is the adult selection type that gives the best fitnesses in our implementation. When it comes to the mutation tests in Chapter 4.4.4, it can be seen that memetic improvement gives much better results than random mutation.

Chapter 4.4.5 shows that larger populations yield better results, but that increasing the size has diminishing returns. It also shows that a population size of 200% the length of the genome is not much better than a population size equivalent to 100% the length of the genome. The findings in the chapter contradict the findings from Chapter 4.4.1, but they indicate that those findings are arising due to a specific combination of factors. A smaller population happens to be better in the following situation:

- The previous best individual is reintroduced each generation.
- The portion of the population the best individual constitutes grows larger as populations grow smaller.
- No other form of elitism or directing of the populations development is present.

None of the configurations found an ideal solution with a fitness value of 60. The best fitness values the MA got were about 400 (found when doing the mutation configurations in Chapter 4.4.4) which is not too bad. The found fitness value indicates that the MA gives a solution that goes $\frac{400}{60} = 6.\overline{66} \approx 7$ times more than it needs around the circle. Considering that this can be achieved by switching the order of 7 pairs in the circle the quality of the solutions while not optimal are still acceptable.

Overall the experiments ran smoothly, usable results were obtained, and more importantly all of the results above are 100% reproducible because all the configurations and random-seeds of each run of each experiment are kept.

There are several things that could be improved. A test that also tests splitting would be better. Using real-world test data would give indications as to running time, which would make it better in itself. Furthermore, real-world data could provide more parameters, which in turn could make the MA perform even better. Another significant limitation is that other configurations of the tournament selection are not looked into.

MA Parameter	**Best found setting**
Parent selection	Fitness proportionate selection
Adult selection	Overproduction
Mutation type	Memetic improvement
Population size	200%

Table 4.1: The tunable parameters in the MA and what settings were found to give the best results

EXPERIMENTS AND RESULTS

In this chapter obtaining results from the MA will be considered. First there will be a section that looks into the results one gets when the MA is applied to the BHW1 benchmark [SINTEF, 2012a]. Because the smaller BHW instances have known optimal solutions, this will give an idea of the MAs performance relative to known optimal solutions. Then there will be a section about obtaining results when map data from Trondheim is used as input to the MA.

5.1 The BHW Benchmark

In this section, the MAs performance on the BHW1 test case is going to be presented. Because the BHW1 benchmark has been solved optimally, and the solution is available, it can be used to verify the output of the MA. When comparing the fitness of the result from the MA with the fitness of the know solution one should be able to gauge the quality of the MAs output. Moreover, when comparing the optimal route with the one made by the MA, it should become apparent whether the MA is on the right track or not.

The section will start out by outlining the experimental plan for the tests. After the experimental plan has been given, the experimental setup will be described. Finally, the results of the tests will be presented.

5.1.1 Experimental Plan

To be able to compare the routes generated by the MA and the known solution for the BHW1 case and their fitnesses, clearly a result from running the MA on the BHW1 benchmark has to be obtained. The generated route and the known solutions' route should be displayed side by side, and the fitness the MA finds should be compared to the fitness of the solution.

5.1.2 Experimental Setup

To be able to compare the fitness and the resulting routes produced by the MA with the known solution, the MA should be set to process the BHW1 case. The result should be compared to the one found on SINTEF [2012b].

To get an idea of what fitnesses the MA generally finds when processing the BHW1 case, it will be run 30 times for 50000 generations with the parameters shown in Table 5.1. The fitness evaluation will be performed by evaluating the sum of the lengths of the splitted trips generated from each genome, due to that being how the fitness of the optimal solution is calculated. After having performed the series of runs, the result from the best run is going to be picked and compared to the known solution.

MA Parameter	Setting to be Used
Parent selection	Fitness proportionate selection
Adult selection	Overproduction
Mutation type	Memetic improvement
Population size	90 individuals

Table 5.1: The parameters used for testing the MA on the BHW1 benchmark.

5.1.3 Results

After running the MA on the BHW1 benchmark with the parameters described above, the average of the best, average, and standard deviation of fitness of the populations each generation are plotted. The charts can be seen in Appendix C. From the chart in Figure C.1, one can read that the best solutions the MA finds on average have a fitness of 368 after 50000 generations.

The best result obtained is the run labeled as *2015-06-24T01-34-22Z*, which can be found in the supplementary digital materials. It has a fitness of 360, which compared to the optimal fitness of 337 is good. The trips found by the MA and the optimal trips from SINTEF [2012b] can be seen in Table 5.2. As can be observed in the table, the trips found by the MA show resemblance to those of the optimal solution. For instance, the first part of the fourth trip obtained by the MA is very similar to the first part of the third trip of the optimal solution.

Considering these results, they clearly show that the MA has not found the optimal solution to the BHW1 benchmark. However, they also show that the output form the MA are of a good quality. Overall the results indicate that the MA produces solutions of a good enough quality for its intended application, route optimization for snow plowing.

	Result from MA	Known Optimal Solution
Trip 1	1-A5-N12-E8-N12-A9-6-E4-6-N12 1	1-A1-N2-E3-9-A11-11-E5-5 1
Trip 2	1-A1-N2-E3-9-A11-N11-E9-8-A10-N10-E10-9-E3-N2 1	1-A2-4-E2-2-E1-N3 5-E6-12 1
Trip 3	1-A2-N4-E2-N2-E3-N2-E1-N2 1	1-A4-10-E11-N11-E9-8 7-E8-12 1
Trip 4	1-A4-N10-E11-N11-E5-5-E4-6-N12 1	1 12-A9-6-E4-5-A7-3-A6-N4 1
Trip 5	1-A3-N7-E7-N7-A8-6-N12 1	1-A3-7-E7-8-A10-N10-E10-9 1
Trip 6	1-A5-N12-E6-5-A7-N3-A6-N4 1	1-A5-N12 N7-A8-6 1
Fitness	**360**	**337**

Table 5.2: The best result obtained from the MA and the known optimal solution for the BHW1 benchmark.

5.2 Map Data from Trondheim

Now that it has been established that the MA works (as shown in Chapter 4), and produces results of a reasonable quality when applied to benchmark problems (detailed in Chapter 5.1), RQ3 should be addressed. RQ3 was stated in Table 1.1 as: “Determine whether the memetic algorithm independently can find a route in Trondheim optimized for snow plowing”.

To properly test whether the MA can route in Trondheim, it should be applied to map data form Trondheim. This section will begin with exploring what data should be gathered to properly deal with RQ3. Then the experiments used to obtain the data will be detailed. Finally after that the results gathered from the experiments will be outlined.

5.2.1 Experimental Plan

To get results on the performance of the MA when trying to create snow plowing routes for Trondheim, clearly it should be fed a graph that is generated from road map data of Trondheim. To check whether the MA is consistently capable of finding solutions it should be run a number of times, and the quality of the results in terms of their fitness should be plotted.

Once the MA has generated a set of results, they should be compared to the routes currently being driven. To do this, the fitnesses of the generated routes should be compared to the fitness the implementation calculates for the routes as they are driven. This way they will both be evaluated in terms of a common reference frame, the model based on the map data from Trondheim.

Then the routes generated by the MA should be drawn up as a map. It could be argued that they should be drawn *on* a map, but due to copyright issues and the nature of the process of generating map tiles it is not feasible to do at this point. When the routes have been drawn up, they can visually be inspected for flaws and abnormalities. Additionally having the routes presented graphically is useful if they are to be applied in a real-life setting by a snowplow driver.

The drawn routes generated by the MA could also then be evaluated by the municipality’s drivers. This evaluation could bring to light aspects of the generated routes that might otherwise be overlooked, and their assessment can also be used as a way to evaluate the quality of the generated routes.

5.2.2 Experimental Setup

To produce snow plowing routes for Trondheim, the MA needs to be supplied a graph of the area with the streets that need to be plowed, as discussed in Chapter 3. An important feature of the chosen NEARP format should be considered here, namely that it requires a starting point be specified (in the “depot node”-field). Due to that the route being generated for the actual snow plowing case is a giant circle, the starting point is somewhat arbitrary. It will add to the length of the route because the MA connects it to the cycle. However, because the connection is the shortest path to the cycle, and it will be the same for all routes, the choice should not have too much influence. It can be argued that the selection of starting point will affect where the cycle begins. However, because it does not change the nature of the cycle otherwise, it should not matter, and the starting point can be chosen arbitrarily. Therefore, the starting point for each route is selected to be the first required element appearing in the input.

To generate the routes then, the MA will be configured with the parameters that are shown in Table 5.3. It will be applied to two sets of roads that need to be plowed in existing routes; route 310174_KV_B_Midtbyen, and route 310179_KV_H_Midtbyen. To get an idea about the MA's performance, in general, it will be run 30 times with this configuration. Ten of the runs are going to be 100000 generations long, to illustrate how the results converge over time. The remaining 20 runs will be going for 50000 generations, and the average performance will be graphed based on the first 50000 generations of all 30 runs combined.

MA Parameter	**Setting to be Used**
Parent selection	Fitness proportionate selection
Adult selection	Overproduction
Mutation type	Memetic improvement
Population size	200 individuals

Table 5.3: The parameters used when applying the MA to map data from Trondheim.

To get the fitness of the routes as they are currently being driven for comparison with the generated routes, the municipality's drivers will be asked to supply a list of what order they clear the roads on their routes in. The lists are then going to be processed by looking up each of the enumerated road elements in the model of the road network data from Trondheim stored in QGIS [QGIS, 2015]. There the IDs of the elements will be obtained, and written down in a file. This file will then be formatted so that it represents a route in the internal format used by the MA, by prepending each element ID with an "E", "A", or "N" to denote whether it is an edge, arc, or node respectively. Then each item will be wrapped in double quotes ("), and separated with a comma (,) so that it can be properly parsed as an array by the MA.

A similar process is going to be used when the resulting routes are drawn as maps. Instead of adding characters to denote whether an element is a node, edge, or arch, all extra characters except the numerical IDs of the elements and delimiters will be stripped. The resulting list is then going to be fed to QGIS in the form of a CSV-file, where a graphical representation of it will be made. Finally, a meeting with the municipality and the snow plow drivers will be set up so that feedback on the results from domain experts can be presented.

5.2.3 Results

The charts showing how the MA behaves when run for 100000 generations can be seen in the figures in Appendix E. From figures E.2 and E.8 it comes forth that the algorithm generally tends to stabilize after 10000 generations, and figures E.1 and E.7 show that there is little improvement in the best found solution after that point. In the charts showing the combined results of the MA running for 50000 generations and 100000 generations when the first 50000 generations are plotted, one can find the fitness at which the MA tends to converge to at 50000 generations. For route 310174_KV_B_Midtbyen the best fitness found on average has a value of 564882 which can be seen in Figure E.4, and for route 310179_KV_H_Midtbyen the average best fitness after 50000 generations is 411770, as seen in E.10.

The routes as they are driven currently as drawn up by the drivers are presented in Appendix D. The overall best fitness obtained for each route, and the fitnesses of how they are being driven at the present can be seen in Table 5.4. For route 310174_KV_B_Midtbyen, the fitness of the route generated by the MA is 35% smaller than the fitness of how the route is currently being driven, which is significantly better. When looking at route 310179_KV_H_Midtbyen, one finds that the fitness obtained by the MA is 50% smaller than the driven one. From this it can be concluded that as long as one assumes that the model is valid, the MA, even though it has not yet found a known optimal solution, is able to find solutions better than the ones made by humans.

Route	Fitness of route generated by the MA	Fitness of the route as it is driven currently
310174_KV_B_Midtbyen	444632	685933
310179_KV_H_Midtbyen	306060	617670

Table 5.4: The best result obtained from the MA and the known optimal solution for the BHW1 benchmark.

However, other than evaluating the routes by their fitness values, they should also be visually inspected to check for anomalies that might not have been caught so far in the process. A graphical representation is also useful if the generated routes are to be applied by drivers in practice. The maps drawn for the generated routes have therefore been made and are presented in figures 5.1 and 5.2, for routes 310174_KV_B_Midtbyen and 310179_KV_H_Midtbyen respectively. The route mapped for 310174_KV_B_Midtbyen had a fitness of 515610, and the one used to generate the map for the 310179_KV_H_Midtbyen route had a fitness of 472610.

In the figures, the numbers represent the order in which a road piece is visited. If one observes the maps carefully, it can be seen that there are some numbers that are missing. There are several reasons for this. First and foremost, some of the road pieces are very small, and labeling them would make the label collide with the labels of adjacent road pieces if one does not use a text size so small that it would be practically unreadable. Another factor that makes displaying the complete enumeration difficult is that some roads are traversed several times. Coupled with an already limited space to display the labels, this makes even more labels disappear. To show all the labels, one would have to print the map on A0 paper with a font-size of 4 points.

The practical solution is to print with the size and resolution used in the figures. The are humanly readable, and it is relatively easy to infer the missing numbers upon inspection. For completeness, the maps have been generated with a finer resolution (40 times 40 meters) in the style of a map book that spans several pages. For route 310174_KV_B_Midtbyen the detailed map can be found in Appendix G.1, and for the 310179_KV_H_Midtbyen route in Appendix G.2. They are also available digitally as interactive maps formatted for QGIS in the supplementary materials.

Figure 5.1 Route 310174_KV_B_Midtbyen Generated by the MA

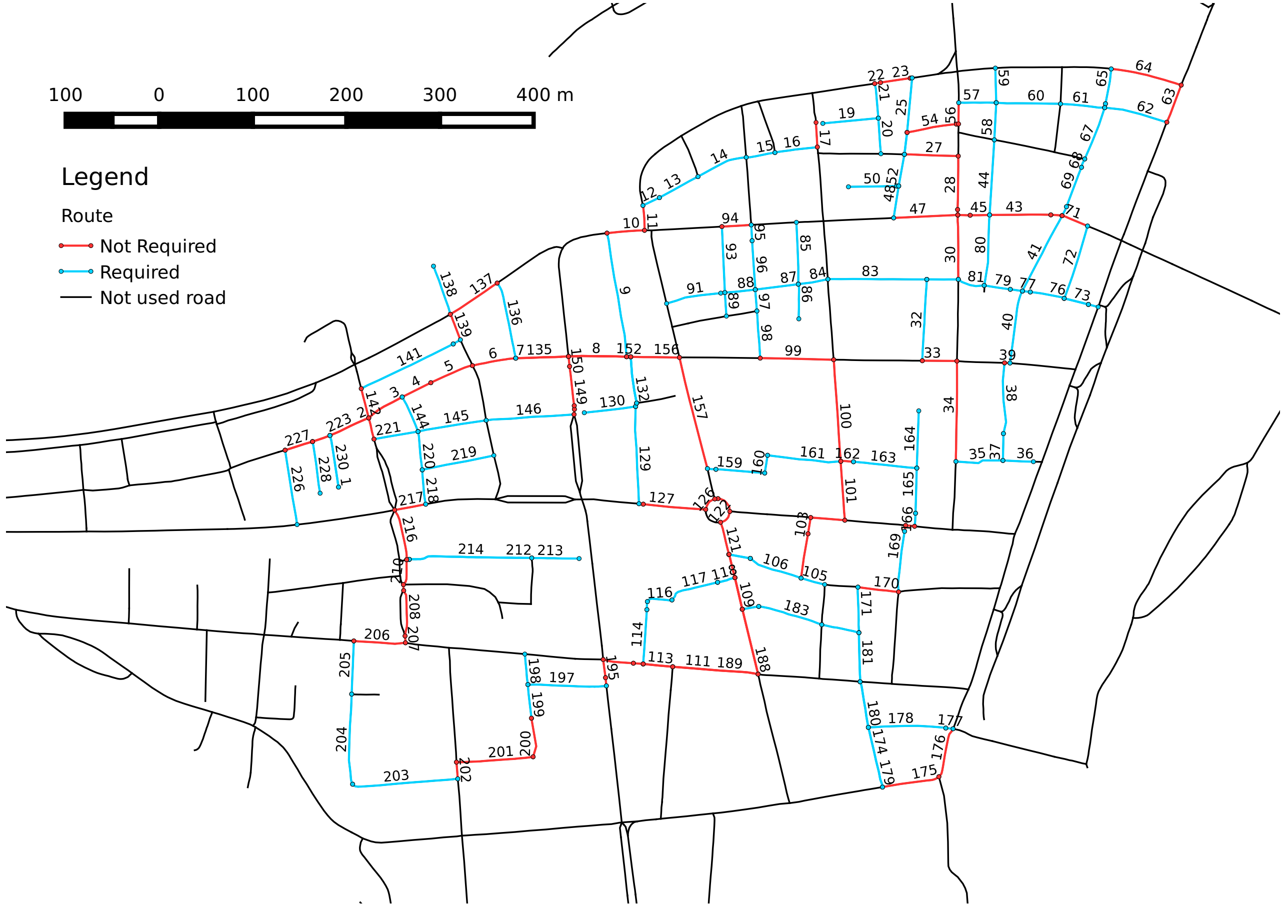
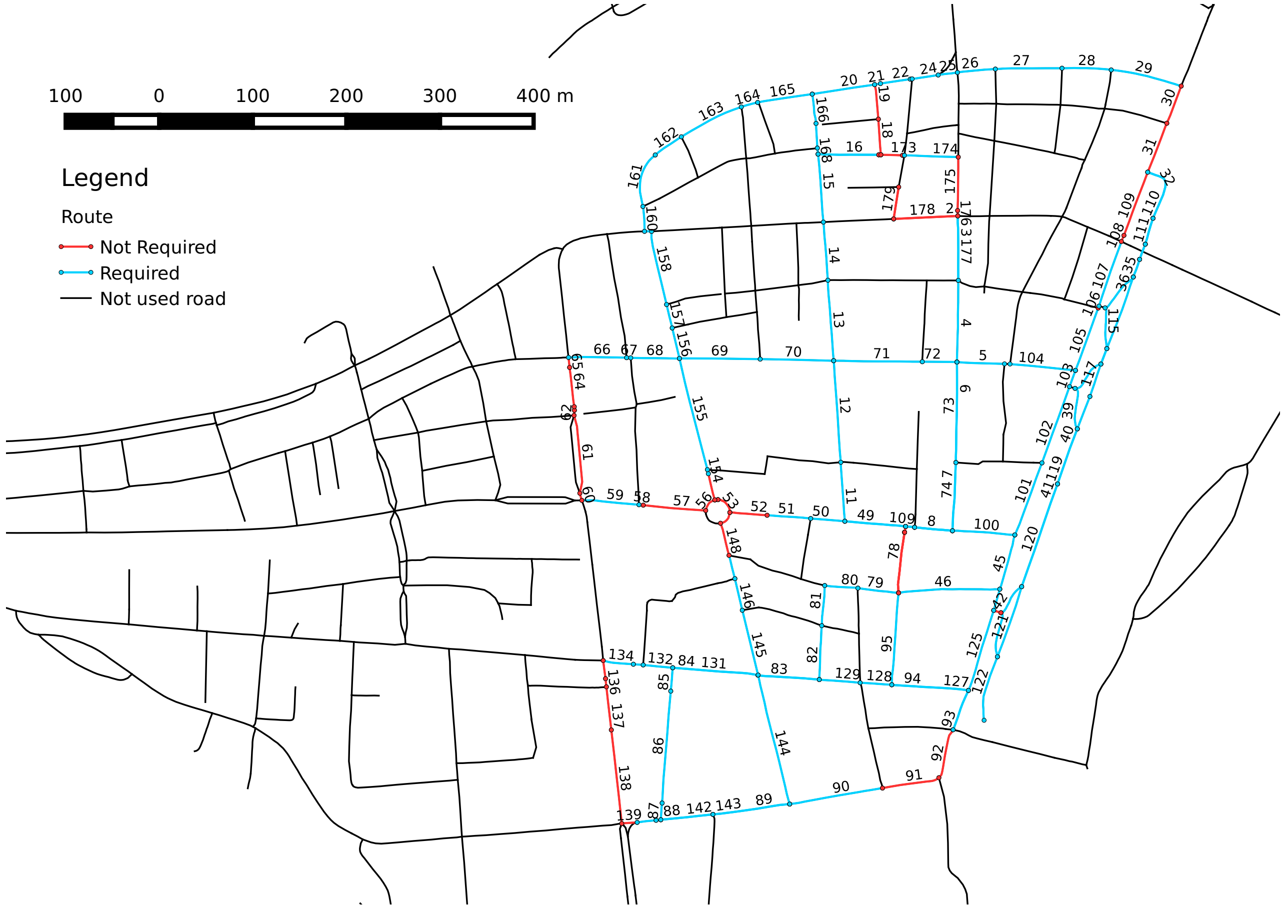


Figure 5.2 Route 310179_KV_H_Midtbyen Generated by the MA



Another interesting thing to observe is that some pieces of road appear to be missing, but still used, such as the pieces at about (9207510°N,1157400°E) and (9207120°N,1157190°E) in the route for 310174_KV_B_Midtbyen (shown in the second and twelfth tiles of the extended map in G.1 respectively). Initially, one might believe that the pieces of road are missing due to errors in the underlying data or have gone missing during the processing. However, the fact that the apparently missing pieces are the only possible entities that can hold the corresponding values that are not shown in the routes at those points indicates that the fault does not lie in the underlying data. Clearly these are road pieces that are not showing up in the QGIS interface, but are still present in the map data and the generated routes. Fortunately, it would seem that it has no effect on the readability of the routes or the correctness of the input and output of the MA.

To get a more thorough assessment of the drawn routes and some feedback on how well the visualization would work in practice, a meeting with the municipality and their snow plow drivers was set up.[1] There the maps as they were drawn in figures 5.1 and 5.2 was presented, alongside the interactive versions in QGIS to facilitate more detailed inspection of the routes.

Overall they found the representation satisfactory. It was pointed out that the drawn maps should have less detail to be more convenient to use in practice. While having each road be represented as relatively small pieces makes sense for maintaining details and integrity of the map data over time, it serves no use when the drivers are reading the maps. Their conclusion was that it would be better if each road was represented and enumerated as one piece between intersections. Although the representation in figures 5.1 and 5.2 is still completely legible and fine to use for evaluation.

Upon closer inspections of the maps, some new anomalies were also discovered. For instance, the road pieces labeled as number 211 to 215 in route 310174_KV_B_Midtbyen are driving in and out of a one-way alley, which should clearly not be done. It was discovered that the road pieces were not labeled as one-way in the map data passed to the MA. Similarly in route 310179_KV_H_Midtbyen the road pieces traversed as items number 53 to 56 and 148 to 152 in the big roundabout in the center of the town should be inaccessible to the vehicle used for servicing that route. The entire square in which the roundabout lies, although still maintained as part of the road network, and therefore in the underlying map data, has been blocked for traffic.

However, the drivers assessed that these were minor flaws that were easy to work around. In their usual work, they said, it often happens that a particular street or alley is blocked by for instance delivery vehicles. Then they have to drive out the way they came and around. Alternatively, if they are to service that end of the road they might have to go and plow other places and remember to come back later.

Another thing brought up at that point was that the route generated by the MA does not take into account that certain roads might need to be traversed several times before they are properly cleared of snow. However, this was also dismissed as a minor flaw in their eyes. Because while the need to traverse a road several times to some extent depends on the width or the road, it also varies with other factors. For instance the quality of the snow, the amount of snow, or the traffic that day.

[1]Meeting with Trondheim Municipality and Drivers. Trondheim Bydrift, 3. etage, Tempeveien 22, 7031 Trondheim, Norway. 2015-06-26 09:00.

From all this, it can be seen that the MA manages to create routes based on data from Trondheim which can be presented in a meaningful way to those that would use them. Moreover, although the routes may have flaws, they are still deemed to be usable in practice, and an interesting take on how the work can be performed by the drivers.

CHAPTER 6

CONCLUSION AND EVALUATION

So far in this thesis many topics have been covered. Chapter 2 covers theory related to snow plowing and route optimization problems. Then in Chapter 3 the implementation of the MA is discussed. In chapters 4 and 5 the performance and outputs of the MA are covered.

This chapter will be focused on rounding off the thesis. It will start off by evaluating what questions have been answered and to what extent. Then there will be a section on the contributions to the field of the work. Finally, the future work that can be done to expand upon what has been covered in this thesis will be outlined.

6.1 Conclusion

When evaluating the work in this thesis, the research questions should be considered. The first research question is stated in Chapter 1 as *"Obtain and present the related work that gives the context for this thesis"*. It is subsequently treated in Chapter 2, which covered both the background for snow plowing and the theory surrounding route optimization problems. Chapter 2.3 discusses why an MA is chosen as the approach to the route optimization problem, and later in Chapter 2.4 MAs are explained more in details. Then in Chapter 2.4.1 the theory on how to adapt an MA better to the task of snow plowing is dealt with. Finally in chapters 2.4.2 and 2.4.3 the reasoning behind the adoption of a couple of known algorithm's is laid out. Thus Chapter 2 presents related work, ties it to the setting of the thesis, and thereby answeres RQ1.

The second research question in this thesis is given as *"Configure the memetic algorithm so that it performs well"*. To do the configuration a test case that is simple for a human to verify the correctness of the solution to is made and presented in Chapter 4. The MA is then applied to the problem with a variety of configurations, and a combination of parameters that work well is found. One might think that RQ2 has been completely answered at this point, but when treating the results in Chapter 4 an interesting pattern in the MA's solutions becomes apparent. The MA, while capable of finding good solutions, has not yet found an optimal solution, even in cases where it is trivial for humans to do so. However, the results it produced are of a sufficient quality to move forwards. This is mainly because the routes found by the MA in Chapter 5.2 are better than the ones currently being driven. The MA can thus be considered to perform well, and RQ2 satisfyingly addressed by the work presented in Chapter 4.

Chapter 5.2 is also relevant when evaluating how the third research question, *"Determine whether the memetic algorithm independently can find a route in Trondheim optimized for snow plowing"*, is dealt with. There it is showed what results are obtained when the MA is applied to data from Trondheim. Moreover, when looking at the outcomes, it would seem that the MA has indeed managed to produce routes that are good in terms of length weighted by speed and with random starting points. However, for the routes generated in Chapter 5.2 it can be argued that they do not have a random starting point because it is chosen to be the first required road piece in a route that appears in the input file to the MA. Still it should be considered that which road piece is the first is rather arbitrary, and that the route is a giant circle and starting on a required point on that circle. Then it becomes clear that it does not really matter where the driver chooses to start as long as he drives the shortest possible path to the route, and that the starting point can be randomly chosen and the route be exactly the same. In that light RQ3 has been fully answered by the work presented in Chapter 5.2. It could also be argued that it shows that given a model of sufficiently high quality, route optimization for snow plowing in Trondheim, or anywhere else in Norway with similar constraints, is possible and feasible.

The question that remains at the end then is whether the goal of *"Make a system that can generate optimal routes for snow plowing in Trondheim"* has been reached. Because the work in Chapter 4 relating to RQ2 shows that the MA has not found optimal solutions, the answer has to be no. However, the work still has merits in terms of that the solutions it finds are estimated to be of a high quality by the domain experts given the input used. As such the project should be considered a success even though its goal has not been reached.

6.2 Contributions

The work carried out in this thesis does not represent an improvement upon know solutions to general route optimization problems, nor does it introduce any new techniques for solving them. In that way, it does not constitute a contribution to pushing the current state of the art of the research further. However it does explore how solutions to route optimization for snow plowing in Trondheim can be done, and in the process it has verified that MAs can be applied with success to real-world route optimization problems. Additionally it has been shown that the routes can be compared to those being driven by humans in a meaningful way.

The most significant outcome of the work for the field is that it shows that the map data required to make routes for snow plowing in Norway exists, is publicly available, and can successfully be used for route optimization. Furthermore, our implementation is a novel take on route optimization for winter road maintenance. As such, the combination of the MA and our fitness function represents a contribution to the field. Moreover, the focus on use of real-world also sets this thesis apart. The presentation of results for route optimization in Trondheim in addition to solutions to benchmark problems is an important contribution. Especially because a lot of the research on route optimization is primarily focused on improving performance and mainly perform tests on benchmark problems.

6.3 Future Work

First and foremost it should be examined why the MA is incapable of finding the optimal solution even in simple cases. Even if it happens to take an unreasonably long time, it would be nice to know that the MA is capable of finding the optimal solution if run indefinitely. On a related note, the overall performance of the MA should be improved. As the implementation is now, it only works serially on one processor core. It could benefit greatly by being optimized for running in parallel on several cores, or perhaps even on a graphics card. The creation of each child, and evaluation of its fitness could be done parallelly, greatly increasing how fast each generation is processed and the overall performance of the MA.

Another improvement that should be done is to merge as many road pieces as possible in the underlying data before it is fed to the MA, or drawn on the map. When merging them before feeding the data to the MA the number of elements in the network to process is reduced, and the performance will improve. The reasons for the improvement would be, amongst other things, that there would be fewer elements to iterate over each time the fitness is calculated. Moreover when there are fewer pieces of road that are required, the number of possible ways to make a route is reduced dramatically. It is also desirable to reduce the number of road pieces when drawing up the generated routes as maps. The reduction would make the routes easier to read for the drivers because it would remove much detail that is not necessary for their work.

In relation to how the maps are read, a nice expansion upon the work would be to make an interactive map of the routes that does not require proficiency in setting up or operating QGIS. If the maps of the routes are available, for instance, in an online format on a webpage, it would be possible for the drivers or other interested parties to view larger routes in more detail. It could also make it easier to compare routes.

However, other than pure technical improvements in how the routes are generated and represented, there are also improvements that can be made to the experiments, and also the input data itself. For instance, considering the discovery about population sizes in Chapter 4.4. There the effects of combinations of parameters reveal that in some cases a smaller population size yields better results than a bigger one. It is clear that it should be examined further to obtain a certain proof of that the only reason for the odd result that given much randomness a smaller population is better if you reinsert the best individual of the previous generation each generation.

Improvements that can be made to the heuristic are however maybe the most important for increasing the quality of the routes. First and foremost by using more of the available features of the underlying data and working them into the data that is sent to the MA. If the width of the road and the number of files for instance were included in the input data, the heuristic could be more precise in determining the amount of vehicles and passes needed for clearing a given road. Coupled with data about what types of vehicles are available and how many it could be used to determine an optimal sectoring of the road network.

With more data available, the heuristic could also be made to optimize other things than just length, or a prioritized combination of features of the routes to optimize. Considering the desire to get the traffic running as smoothly as fast as possible, optimizing for clearing the most heavily trafficked roads first and finishing the job as quickly as possible could be one thing to gear the MA towards. Another thing that could be interesting to optimize for is the fuel consumption of the plowing. This would affect the environmental impact of the plowing, but would also be important in reducing the total cost of the plowing operations.

Bibliography

Belenguer, J. M., Benavent, E., 2003. A cutting plane algorithm for the capacitated arc routing problem. Computers & Operations Research 30 (5), 705–728.

Brandão, J., Eglese, R., 2008. A deterministic tabu search algorithm for the capacitated arc routing problem. Computers & Operations Research 35 (4), 1112–1126.

Christiansen, C. H., Lysgaard, J., Wøhlk, S., 2009. A branch-and-price algorithm for the capacitated arc routing problem with stochastic demands. Operations Research Letters 37 (6), 392–398.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische mathematik 1 (1), 269–271.

Dussault, B., Golden, B., Groër, C., Wasil, E., 2013. Plowing with precedence: A variant of the windy postman problem. Computers & Operations Research 40 (4), 1047–1059.

Edmonds, J., Johnson, E. L., 1973. Matching, euler tours and the chinese postman. Mathematical programming 5 (1), 88–124.

Eglese, R. W., 1994. Routeing winter gritting vehicles. Discrete applied mathematics 48 (3), 231–244.

Eiselt, H. A., Gendreau, M., 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.

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

Frederickson, G. N., Hecht, M. S., Kim, C. E., 1976. Approximation algorithms for some routing problems. In: Foundations of Computer Science, 1976., 17th Annual Symposium on. IEEE, pp. 216–227.

Ghiani, G., Improta, G., 2000. An algorithm for the hierarchical chinese postman problem. Operations Research Letters 26 (1), 27–32.

Holland, J. H., 1975. Adaptation in natural and artificial system: an introduction with application to biology, control and artificial intelligence. Ann Arbor, University of Michigan Press.

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

Lacomme, P., Prins, C., Ramdane-Cherif, W., 2004. Competitive memetic algorithms for arc routing problems. Annals of Operations Research 131 (1-4), 159–185.

Lacomme, P., Prins, C., Ramdane-Chérif, W., 2001. A genetic algorithm for the capacitated arc routing problem and its extensions. In: Boers, E. (Ed.), Applications of Evolutionary Computing. Vol. 2037 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 473–483.
URL http://dx.doi.org/10.1007/3-540-45365-2_49

Laporte, G., 1992. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59 (3), 345–358.

Liao, Y.-H., Sun, C.-T., 2001. An educational genetic algorithms learning tool. IEEE Trans. Education 44 (2), 20.

Moscato, P., et al., 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report 826, 1989.

Norwegian Public Roads Administration, 2014. Håndbok R610 Standard for drift og vedlikehold av riksveger. Statens vegvesen Vegdirektoratet, P.O. box 8142 Dep, NO-0033 OSLO, Norway.

Pearn, W. L., Liu, C., 1995. Algorithms for the chinese postman problem on mixed networks. Computers & operations research 22 (5), 479–489.

Pearn, W. L., Wu, T., 1995. Algorithms for the rural postman problem. Computers & Operations Research 22 (8), 819–828.

Prins, C., Bouchenoua, S., 2005. A memetic algorithm solving the vrp, the carp and general routing problems with nodes, edges and arcs. In: Recent advances in memetic algorithms. Springer, pp. 65–85.

QGIS, 06 2015. QGIS Project Homepage. [Online; accessed 2015-07-03].
URL http://www.qgis.org/en/site/index.html

Randby, S., Thrap, M., 12 2014. Route optimization for winter road maintenance. Report, [Unpublished, can be found in Appendix F].

Rao, T., Mitra, S., Zollweg, J., 2011. Snow-plow route planning using ai search. In: Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on. IEEE, pp. 2791–2796.

Santos, L., Coutinho-Rodrigues, J., Current, J. R., 2010. An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transportation Research Part B: Methodological 44 (2), 246–266.

SINTEF, 06 2012a. BHW benchmark. [Online; accessed 2015-06-18].
URL http://www.sintef.no/projectweb/top/nearp/bhw-benchmark/

SINTEF, 07 2012b. BHW1 benchmark solution. [Online; accessed 2015-06-24].
URL http://www.sintef.no/globalassets/project/top/nearp/bhw-results/bhw1.txt

SINTEF, 06 2012c. NEARP Documentation. [Online; accessed 2015-05-05].
URL `http://www.sintef.no/projectweb/top/nearp/documentation/`

The Norwegian Mapping Authority, 02 2013. SOSI-Standard in English. [Online; accessed 2015-05-05].
URL `http://kartverket.no/en/SOSI-Standard-in-English/SOSI/SOSI-Standard-in-English/`

Trondheim Kommune, 02 2015. Vinterdrift 2014/15. [Online; accessed 2015-05-05].
URL `https://www.trondheim.kommune.no/content/1117746169/Vinterdrift-2014-15`

Turing, A. M., 1950. Computing machinery and intelligence. Mind, 433–460.

Ulusoy, G., 1985. The fleet size and mix problem for capacitated arc routing. European Journal of Operational Research 22 (3), 329–337.

Usberti, F. L., França, P. M., França, A. L. M., 2013. Grasp with evolutionary path-relinking for the capacitated arc routing problem. Computers & Operations Research 40 (12), 3206–3217.

Wøhlk, S., 2008. A decade of capacitated arc routing. In: The vehicle routing problem: latest advances and new challenges. Springer, pp. 29–48.

Appendix

APPENDIX A

PSEUDOCODE

A.1 Floyd-Warshall Pseudocode

Algorithm 1 Floyd-Warshall

```
procedure FLOYD-WARSHALL(Graph)
    let $numberOfElements \leftarrow \sum(|N| + |E| + |A|) \in Graph$
    let $distances$ be a $numberOfElements \times numberOfElements$ array
    let $successors$ be a $numberOfElements \times numberOfElements$ array
    set all entries in $distances$ to $\infty$
    set all entries in $successors$ to $-1$

    for each element $e \in N \cup E \cup A \in Graph$ do
        $distances[e][e] \leftarrow 0$
        $successors[e][e] \leftarrow e$
    end for

    for each element $e \in N \cup E \cup A \in Graph$ do
        let $adjacent \leftarrow$ elements adjacent to $e$ in $Graph$
        for each element $e_a \in adjacent$ do
            $distances[e][e_a] \leftarrow e_a.passThroughCost$
            $successors[e][e_a] \leftarrow e_a$
            $successors[e_a][e] \leftarrow e$
        end for
    end for

    for $k \leftarrow 0$ to $numberOfElements$ do
        for $i \leftarrow 0$ to $numberOfElements$ do
            for $j \leftarrow 0$ to $numberOfElements$ do
                if $distances[i][j] > distances[i][k] + distances[k][j]$ then
                    $distances[i][j] = distances[i][k] + distances[k][j]$
                    $successors[i][j] = successors[i][k]$
                end if
            end for
        end for
    end for

    for each element $(e_i, e_j) \in N \cup E \cup A \in Graph$ do
        if $e_i \neq e_j$ then
            $distances[e_j][e_i] \leftarrow distances[e_j][e_i] - e_i.passThroughCost$
        end if
    end for
end procedure
```

A.2 Split Pseudocode

Algorithm 2 Split

```
1: procedure SPLIT(Genotype,Graph,Distances)
2:     let $numberOfElements \leftarrow \sum(|N| + |E| + |A|) \in Graph$
3:     let $e_k$ be the k'th element from the Genotypes ordering of the set $(N \cup E \cup A) \in Graph$ in its genome
4:     let $costs$ be an array of length $numberOfElements + 1$
5:     let $predecessors$ be an array of length $numberOfElements + 1$
6:     set all entries in $costs$ to $\infty$
7:     $costs[0] \leftarrow 0$
8:     $predecessors[0] \leftarrow 0$

9:     for $i \leftarrow 1$ to $numberOfElements + 1$ do
10:        $j \leftarrow i$
11:        $load \leftarrow 0$
12:        $cost \leftarrow 0$
13:        do
14:            $load \leftarrow load + e_j.demand$

15:            if $i = j$ then
16:                $cost \leftarrow$ ( distance from depot to $e_{i-1}$) $+ e_{i-1}.servicing\_cost +$ ( distance from $e_{i-1}$ to the depot )
17:            else
18:                $cost \leftarrow cost -$ ( distance from $e_{j-2}$ to depot ) $+$ (distance from $e_{j-2}$ to $e_{j-1}$) $+ e_{j-1}.servicing\_cost +$ ( distance from $e_{j-1}$ to the depot )
19:            end if

20:            if $(load < vehicle\_capacity) \wedge (costs[i-1] + cost < costs[j])$ then
21:                $costs[j] \leftarrow costs[i-1] + cost$
22:                $predecessors[j] \leftarrow i - 1$
23:            end if

24:            $j \leftarrow j + 1$
25:        while $(j < numberOfElements + 1) \wedge (load < vehicle\_capacity)$
26:    end for

27:    let $Genotype.fitness \leftarrow costs[last\_index]$
28: end procedure
```

A.3 Split Retrieve Trips Pseudocode

Algorithm 3 Retrieve Trips from Split

```
1:  procedure RETRIEVE TRIPS FROM SPLIT(predecessors, genotype)
2:      let $list\_of\_trips$ be an empty list
3:      $j \leftarrow$ number of tasks
4:      do
5:          $i \leftarrow predecessors[j]$
6:          let $current\_trip$ be a new array of size $j - i$
7:          for $k \leftarrow i + 1$ to $k \leq j, k \leftarrow k + 1$ do
8:              $current\_trip[k - (i + 1)] \leftarrow genotype.genome[k - 1]$
9:          end for
10:         add $current\_trip$ to the beginning of $list\_of\_trips$
11:         $j \leftarrow i$
12:     while $i \neq 0$
13: end procedure
```

Appendix B

Memetic Algorithm Tuning Results

B.1 Population Size Testing Results

Figure B.1 Population Size - Average Best

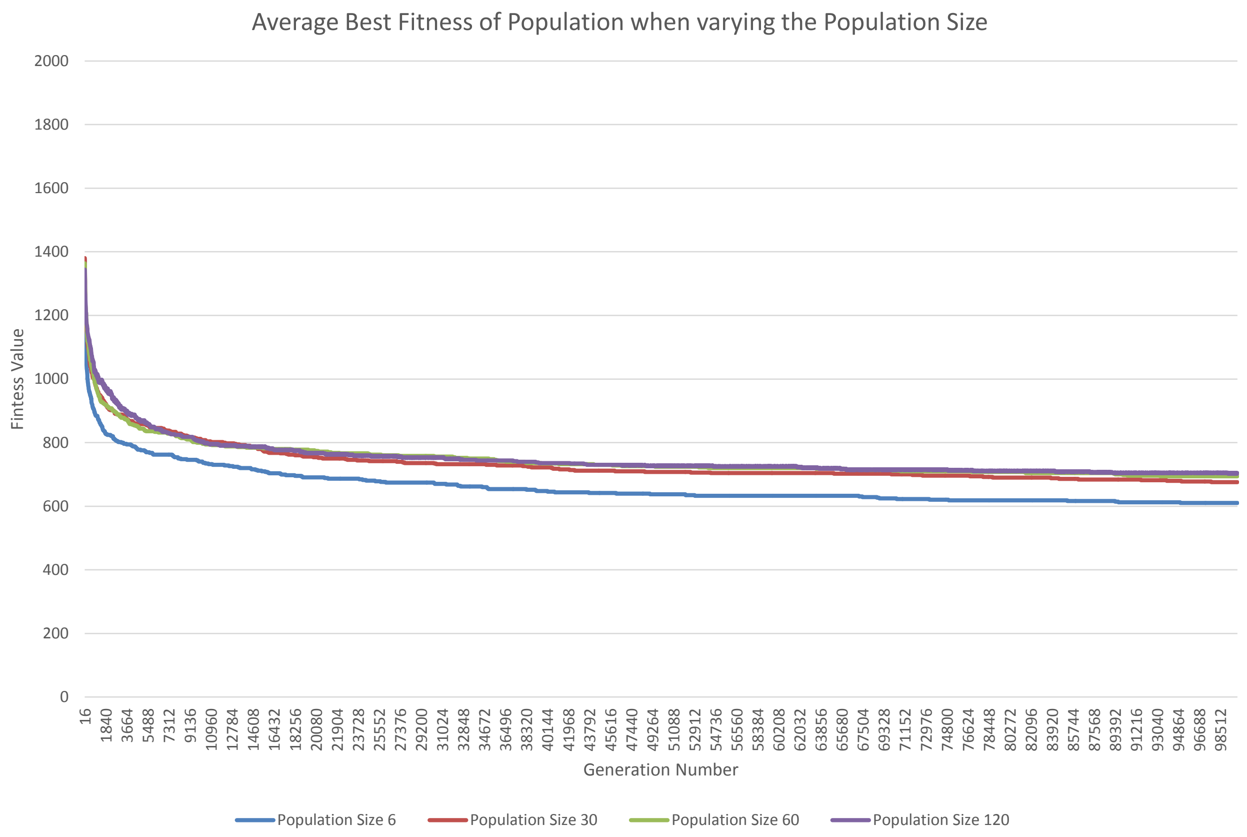


Figure B.2 Population Size - Average Average

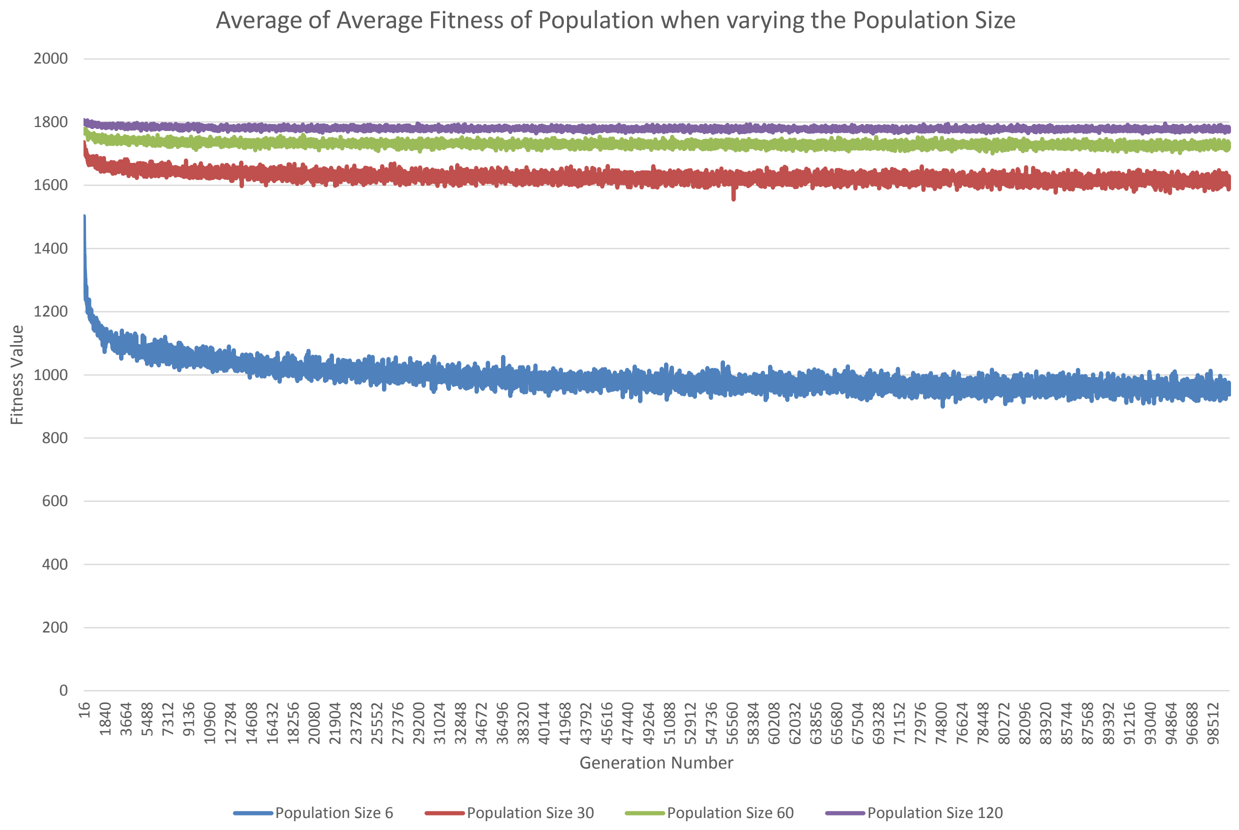
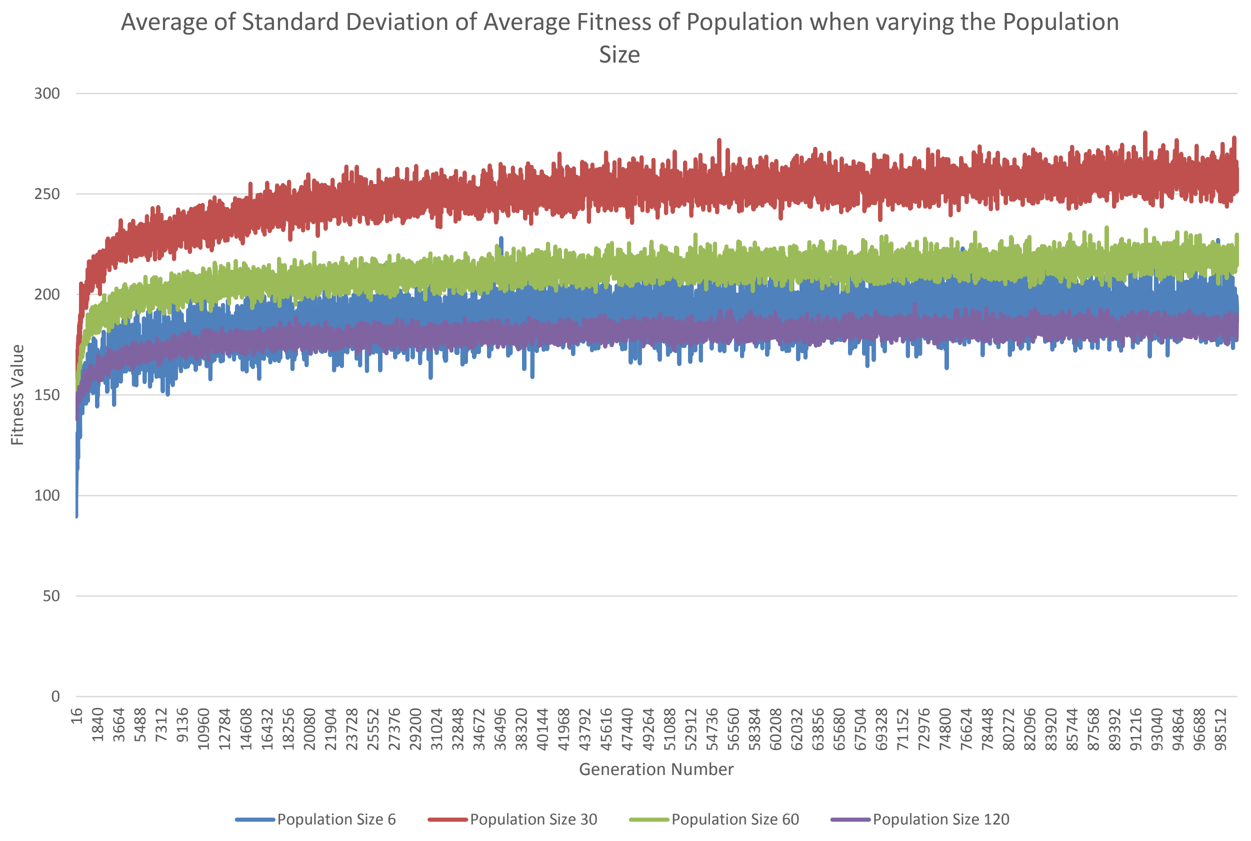


Figure B.3 Population Size - Average Standard Deviation



B.2 Parent Selection Results

Figure B.4 Parent Selection - Average Best

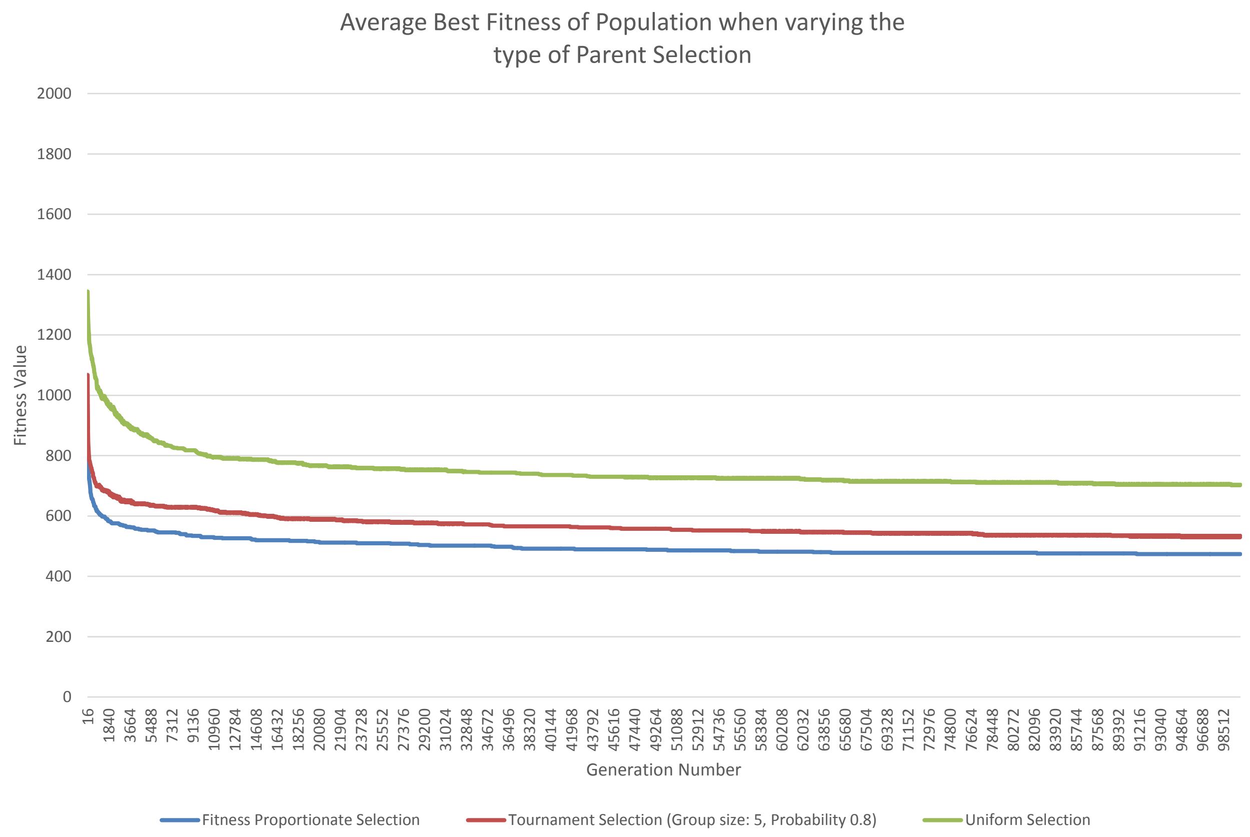


Figure B.5 Parent Selection - Average Average

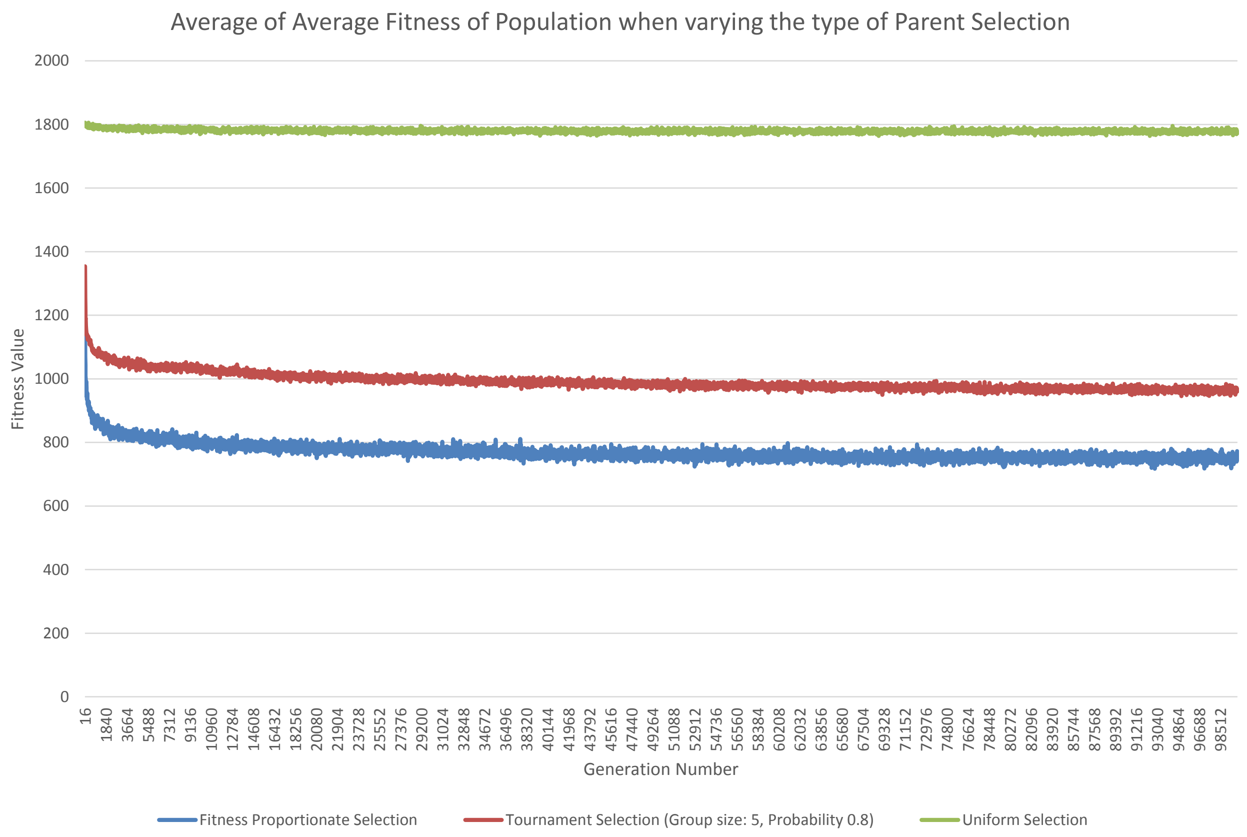
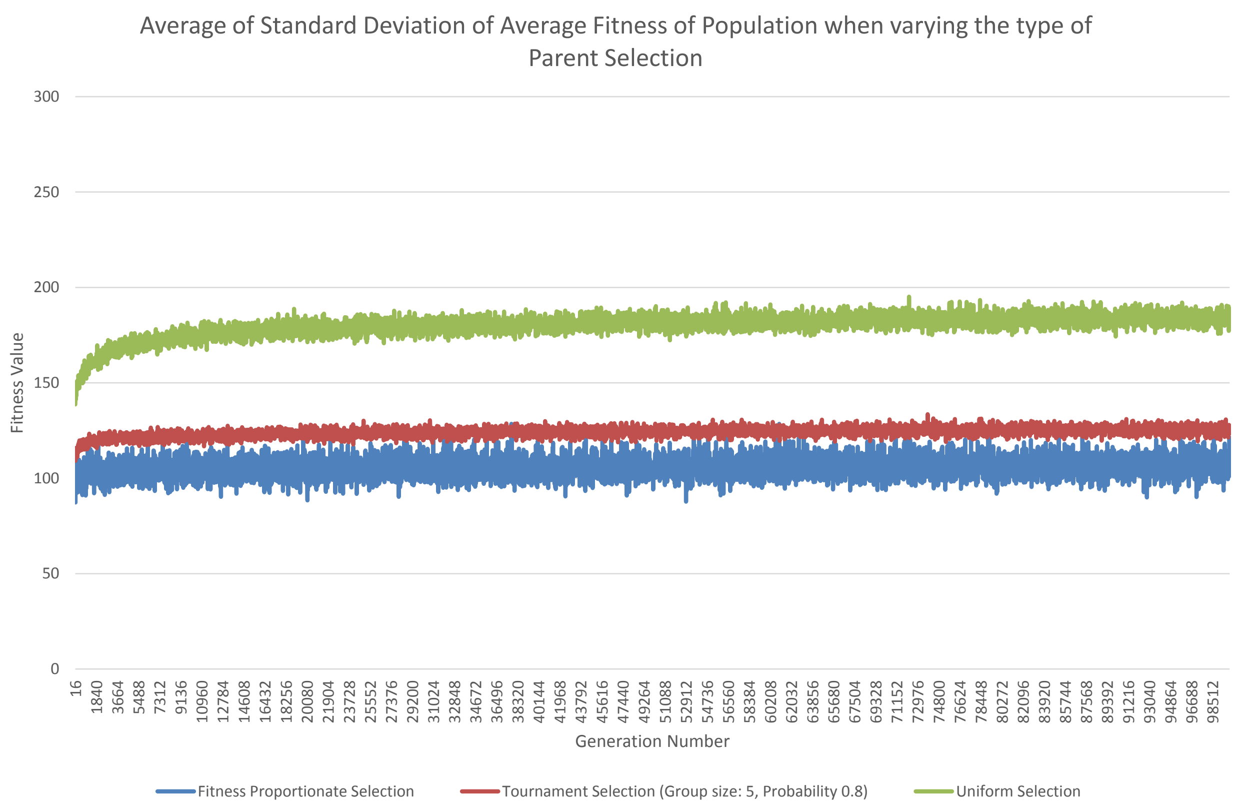


Figure B.6 Parent Selection - Average Standard Deviation



B.3 Adult Selection Results

Figure B.7 Adult Selection - Average Best

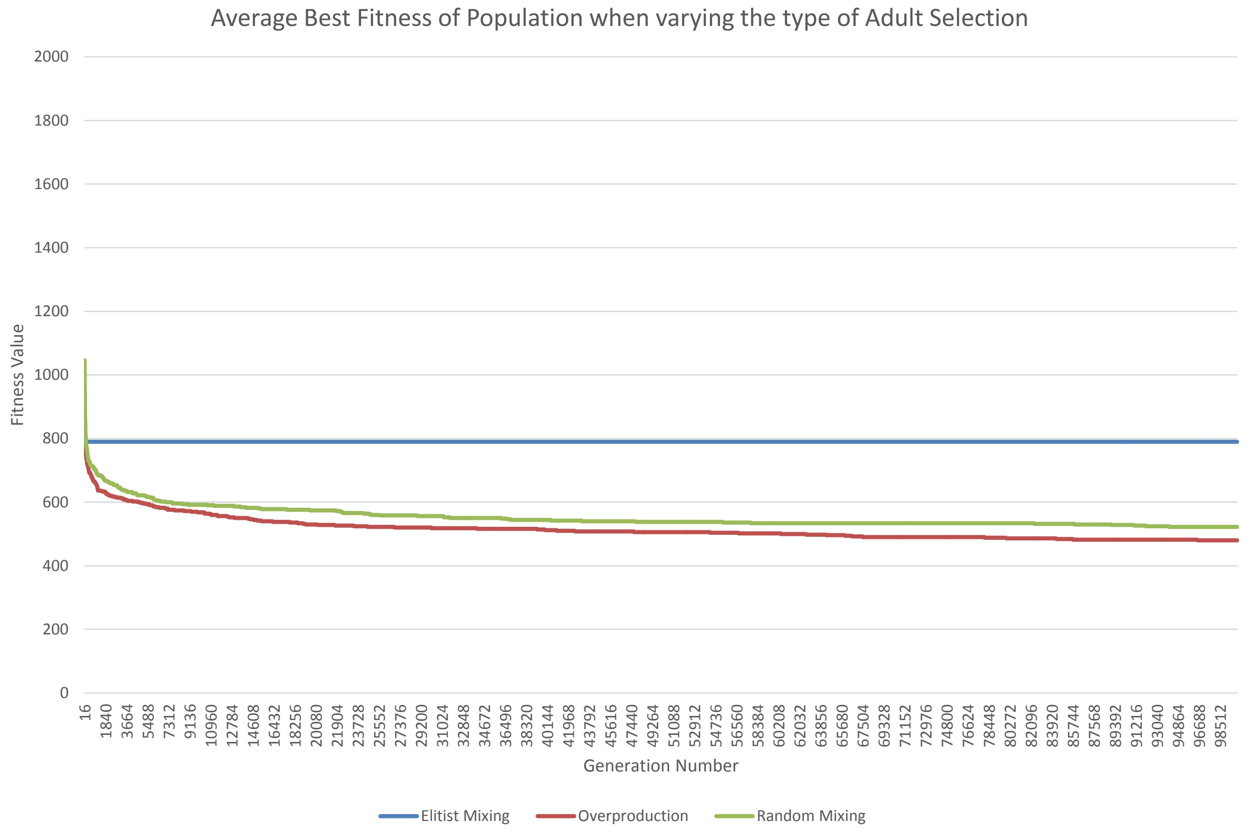


Figure B.8 Adult Selection - Average Average

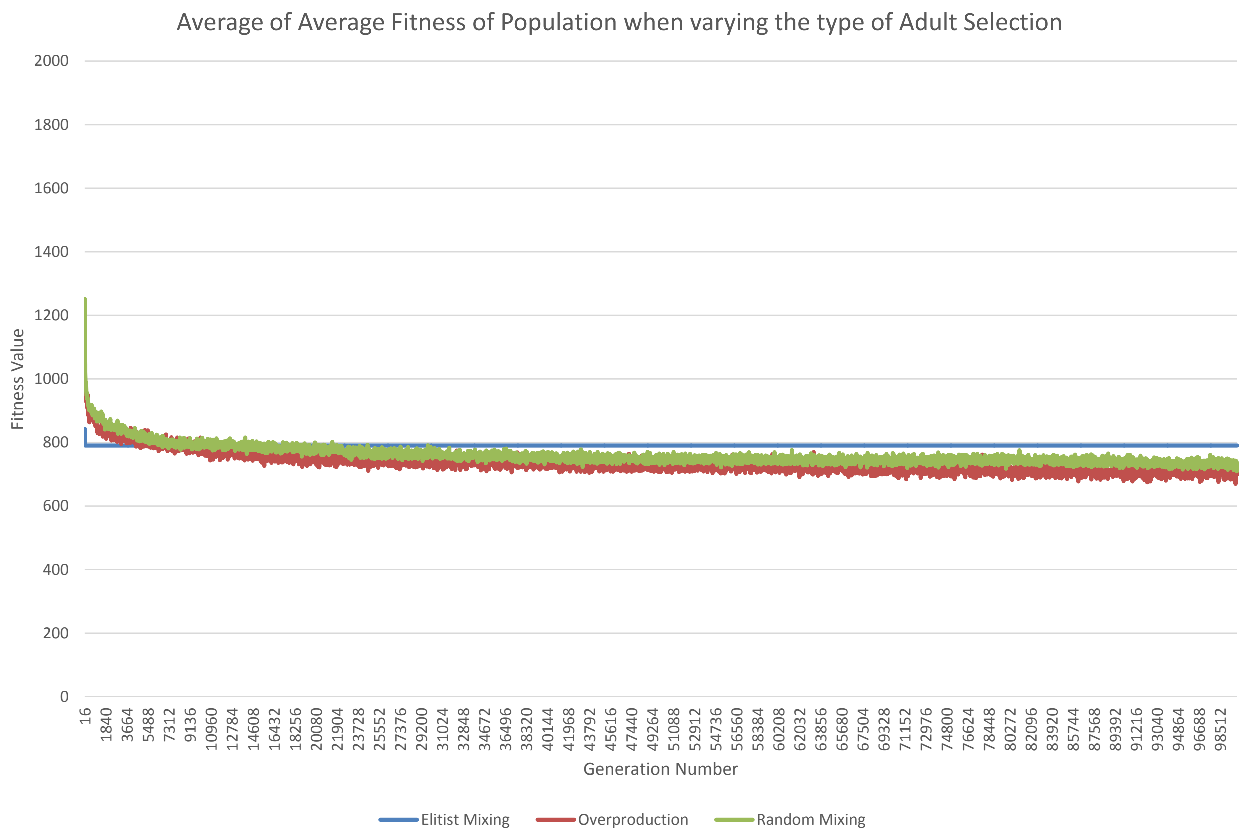
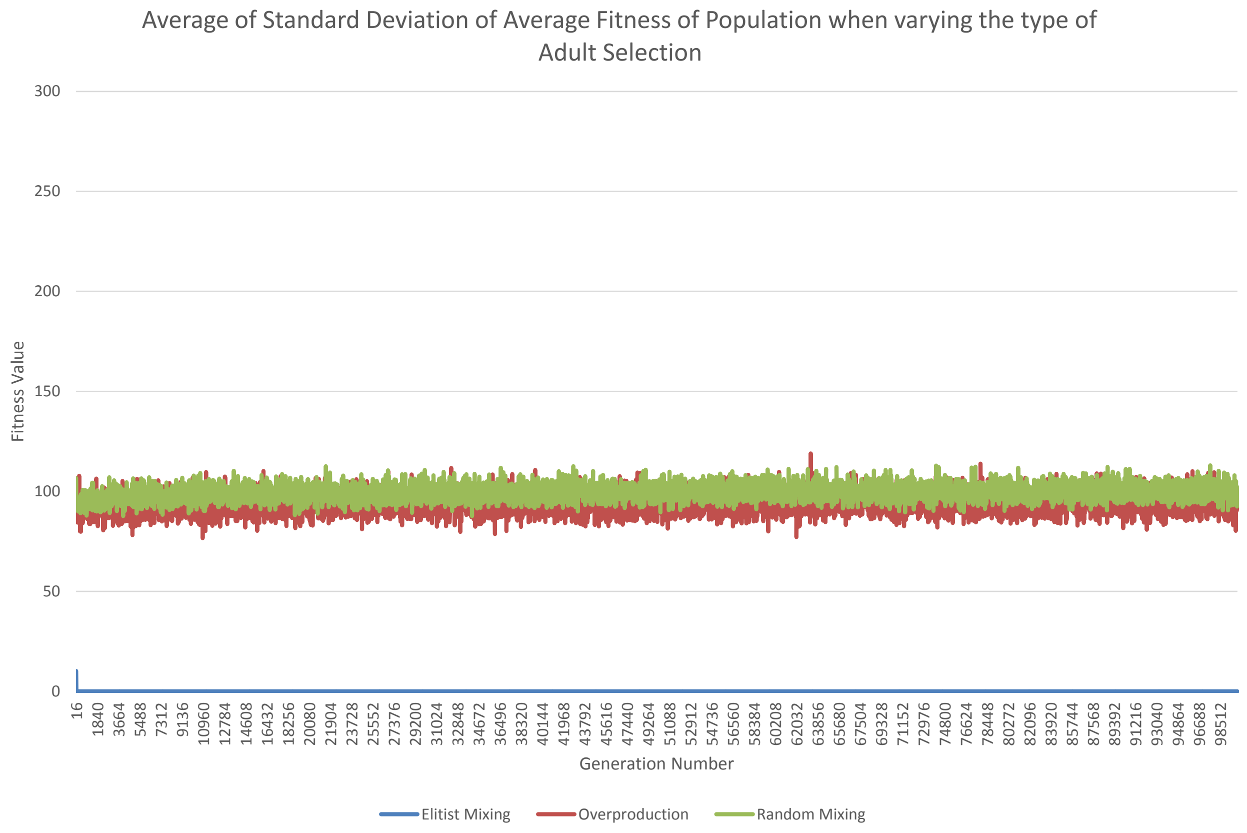


Figure B.9 Adult Selection - Average Standard Deviation



B.4 Mutation Types Results

Figure B.10 Mutation - Average Best

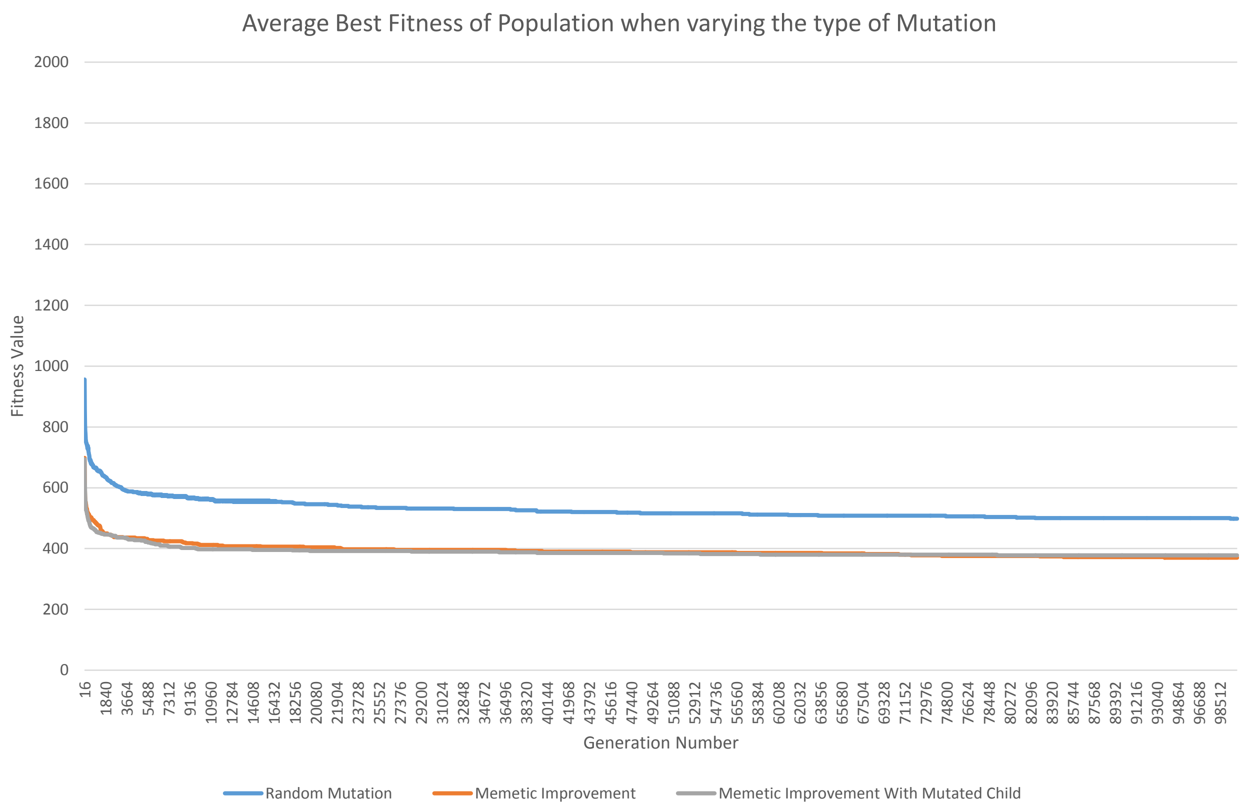


Figure B.11 Mutation - Average Average

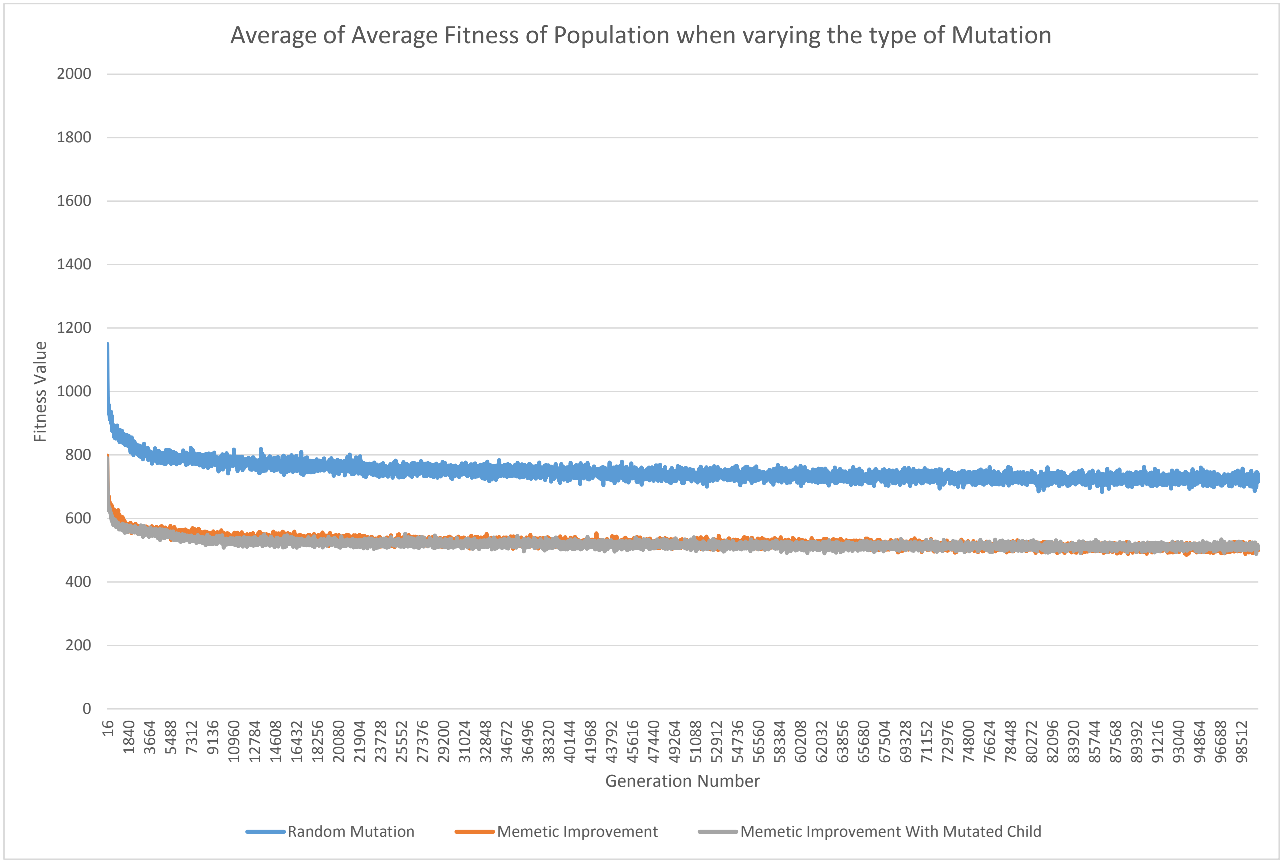
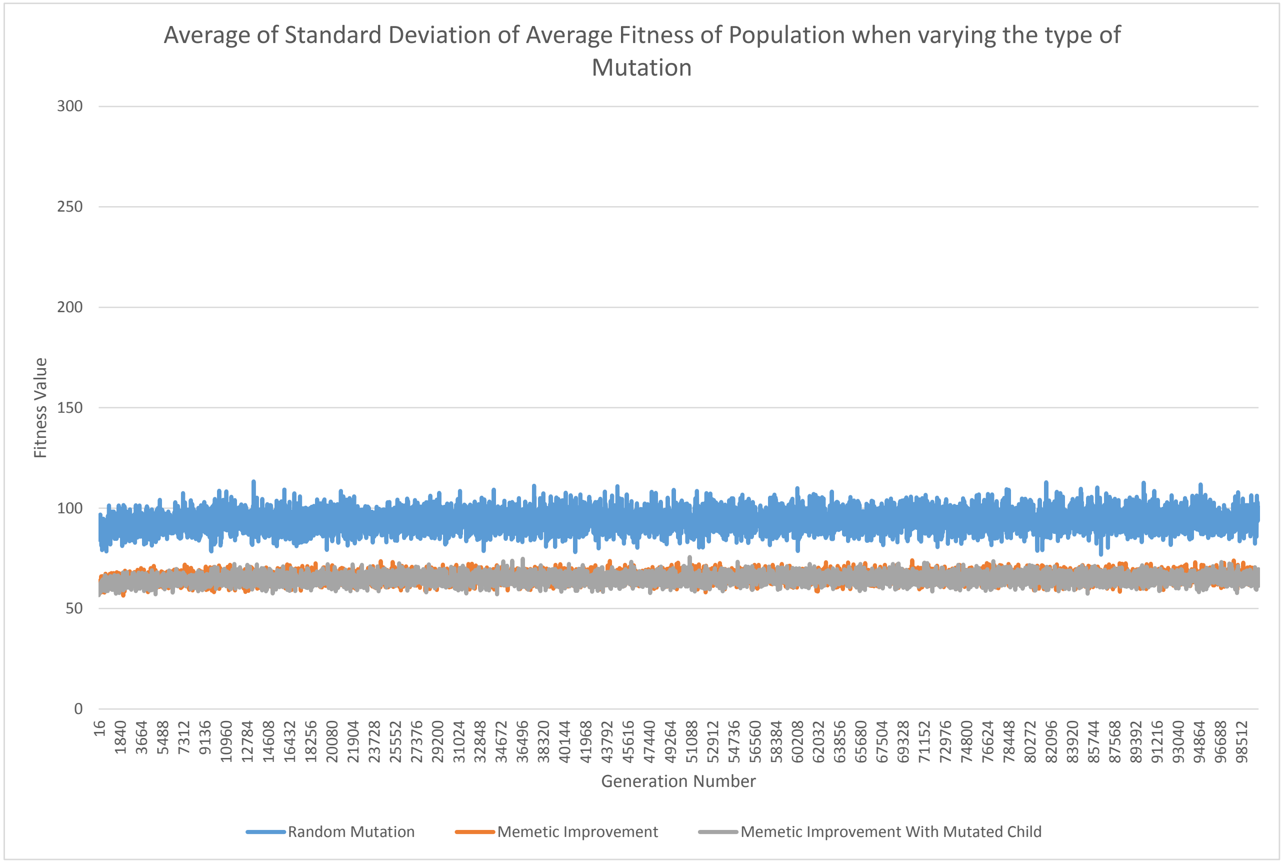


Figure B.12 Mutation - Average Standard Deviation



B.5 Population Size Control Results

Figure B.13 Population Size Control - Average Best

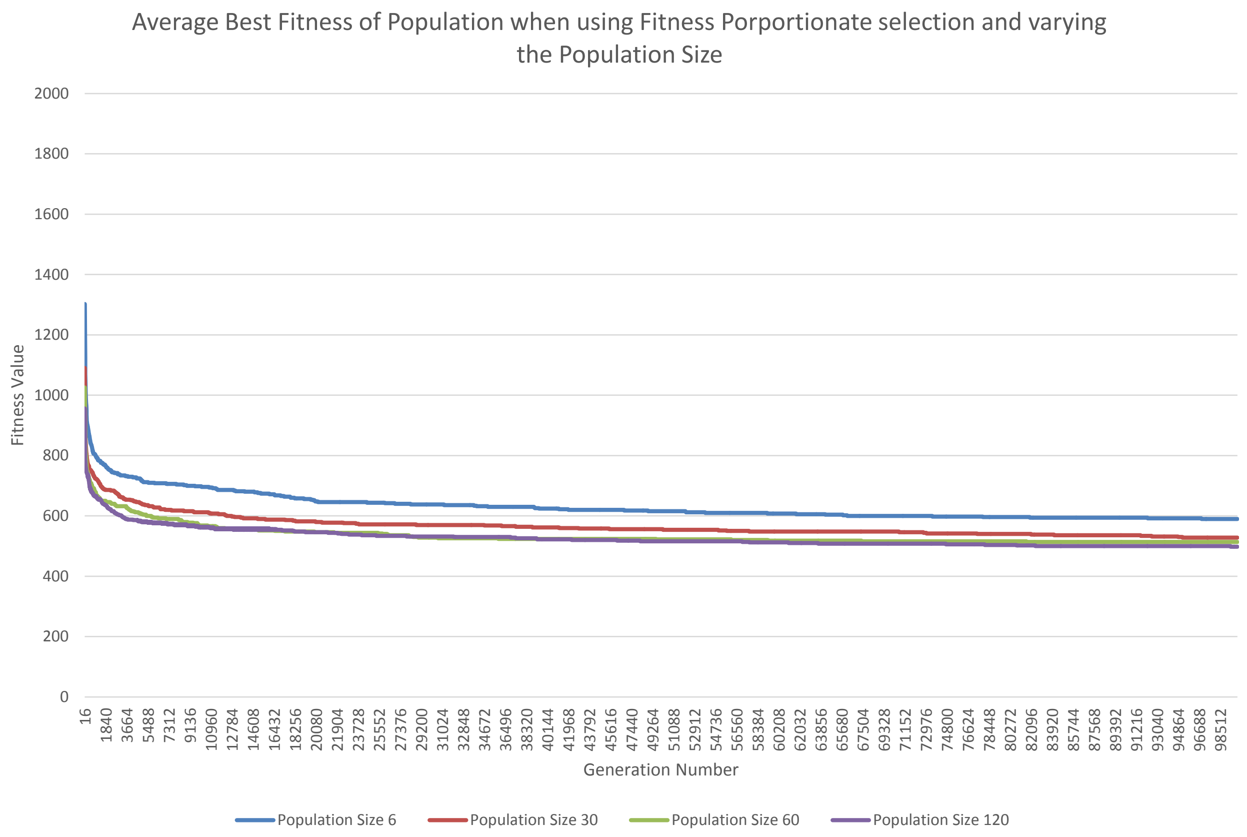


Figure B.14 Population Size Control - Average Average

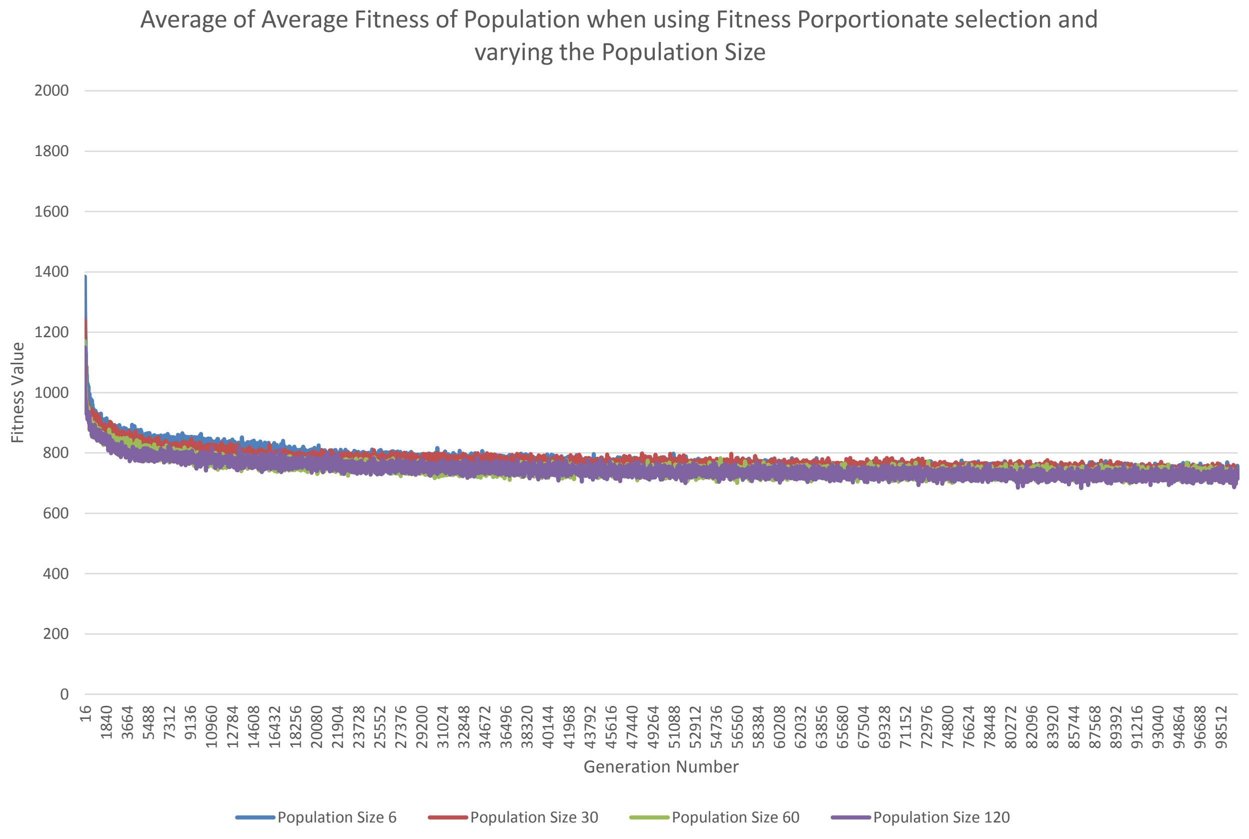
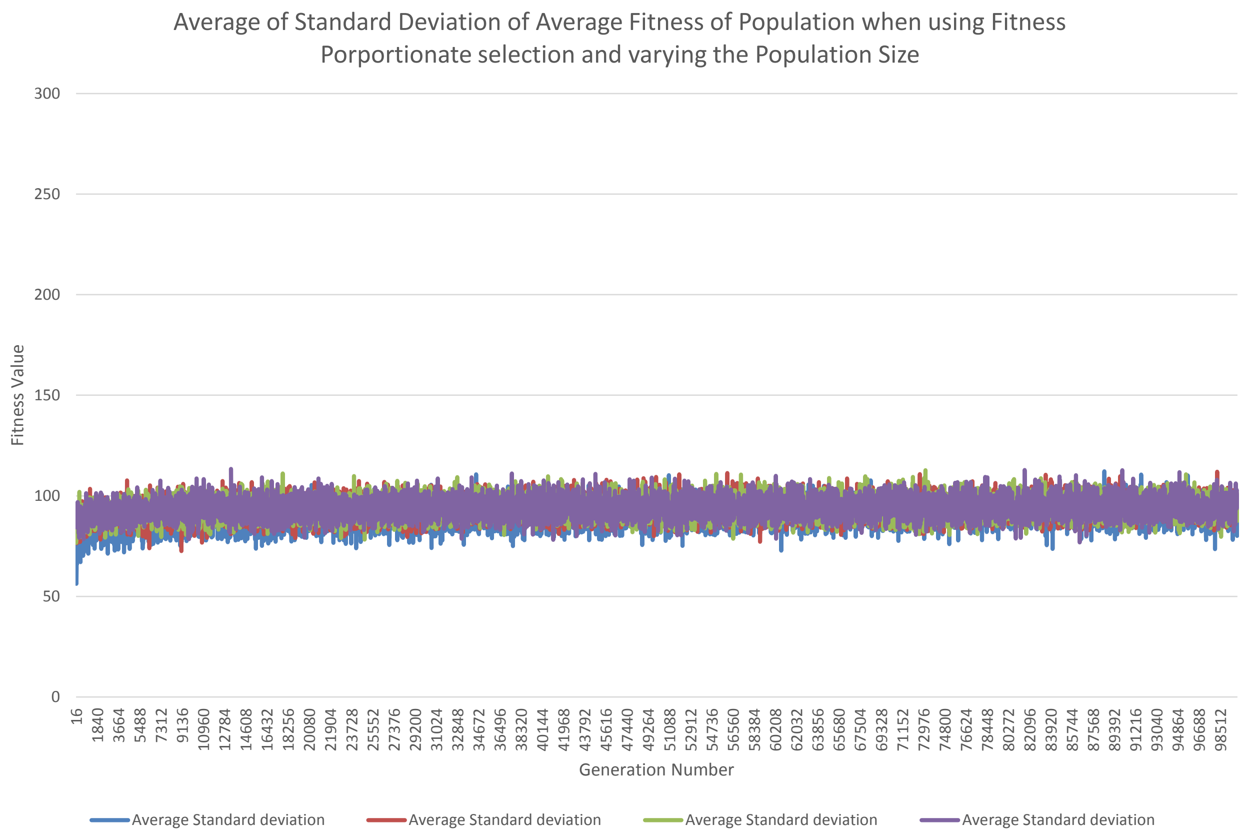


Figure B.15 Population Size Control - Average Standard Deviation



Appendix C

BHW1 Benchmark Results

Figure C.1 BHW1 - Average Best

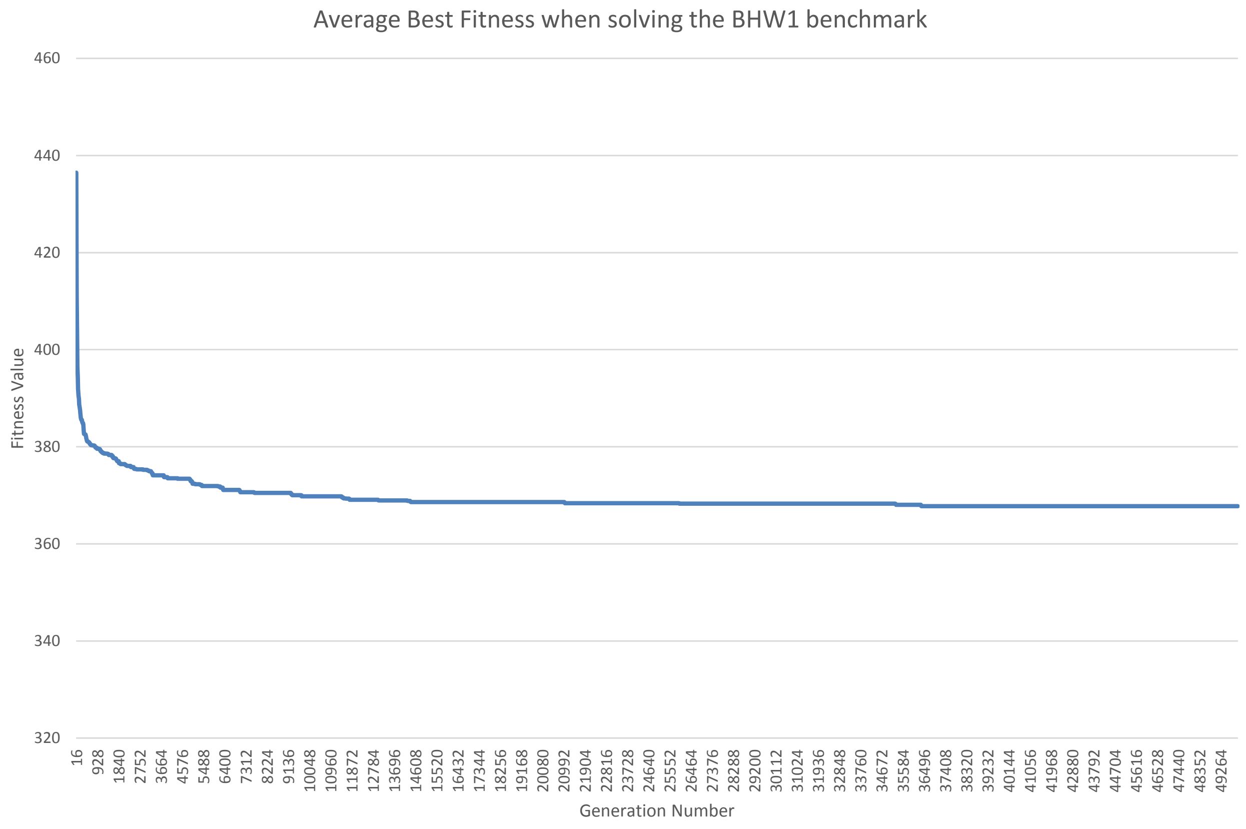


Figure C.2 Population Size Control - Average Average

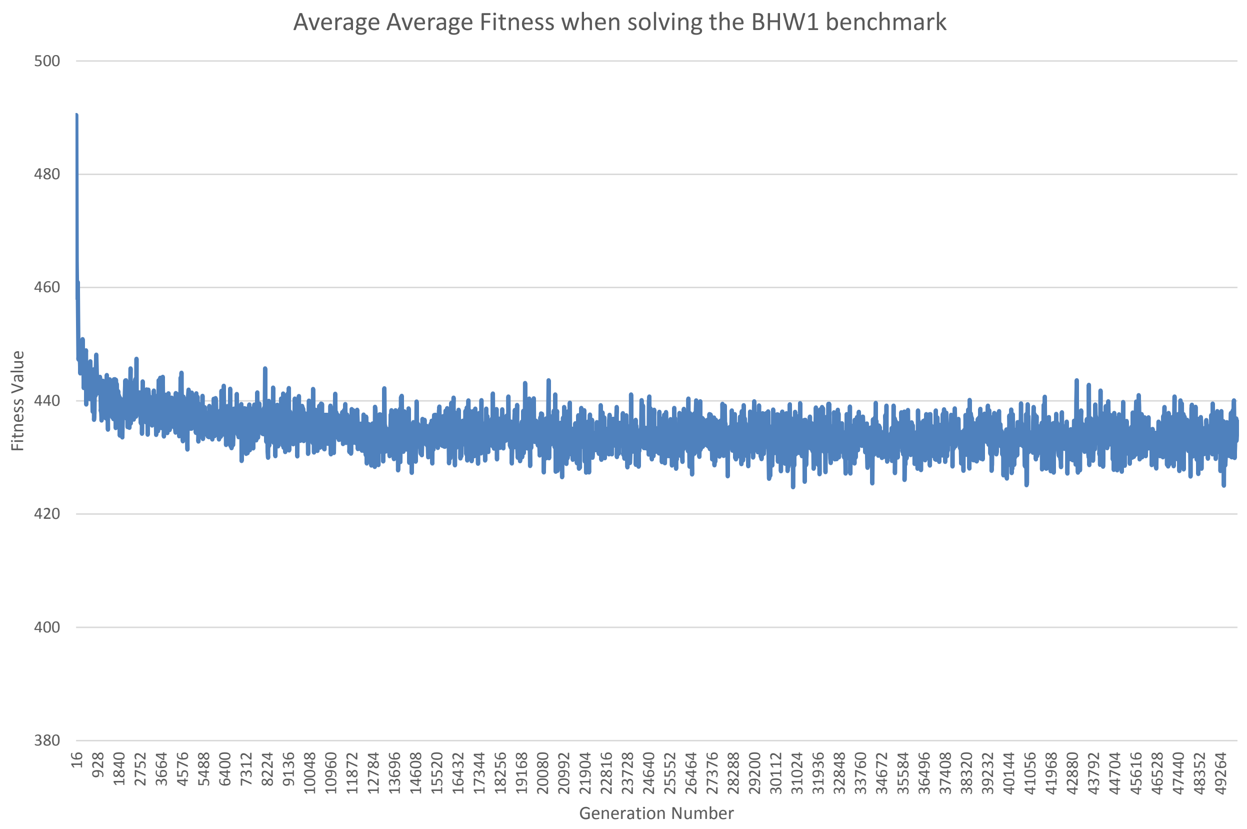
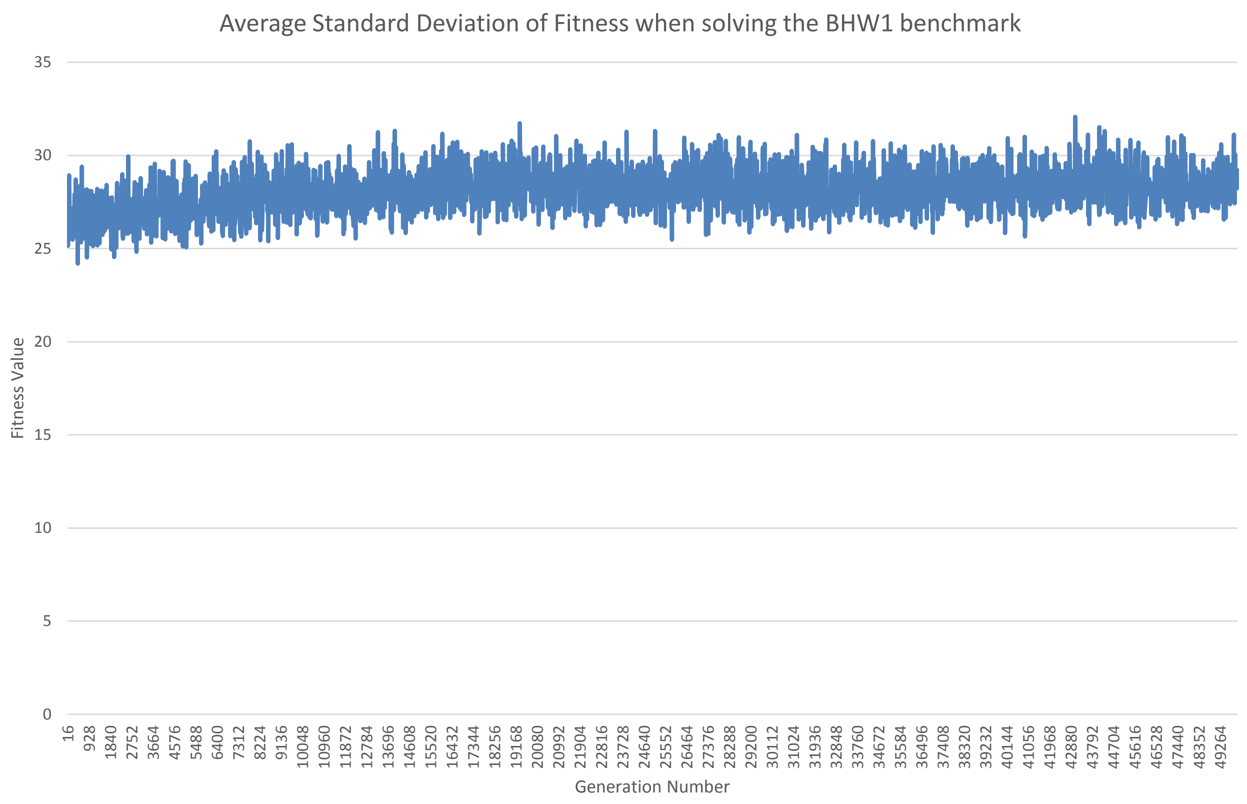


Figure C.3 BHW1 - Average Standard Deviation



Appendix D

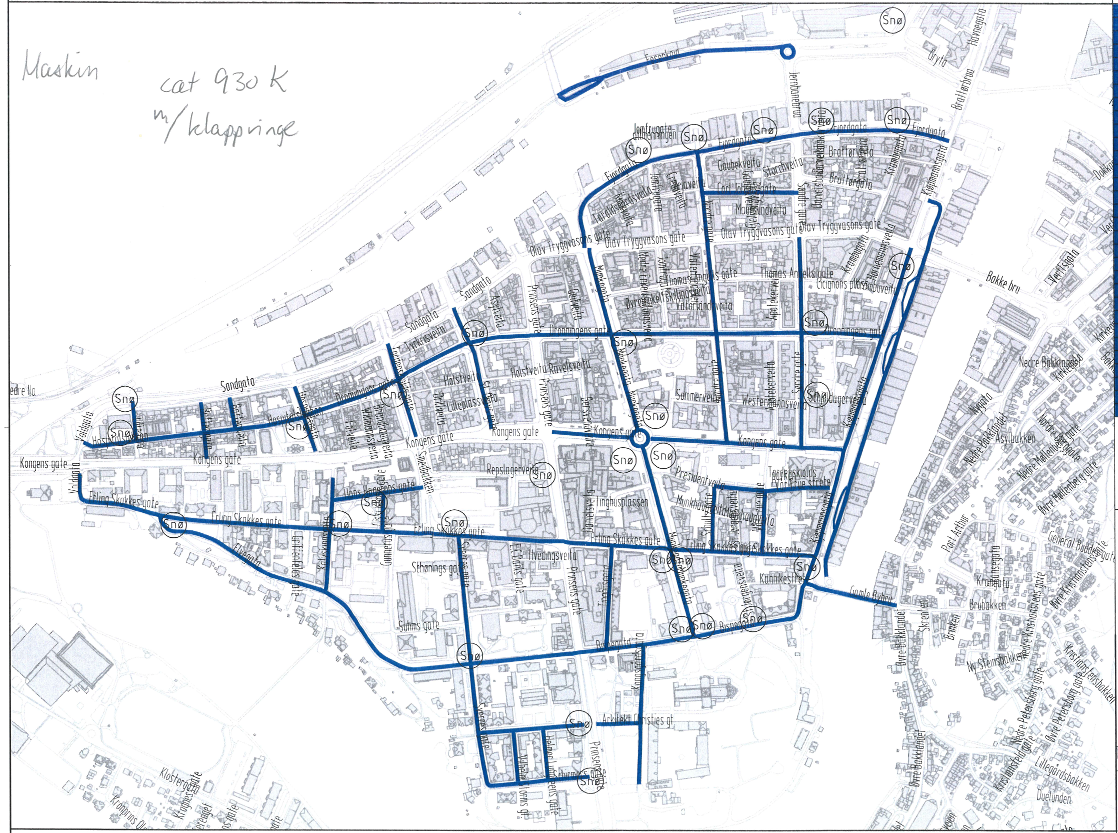
How Routes are Driven as Drawn by the Drivers

VEST - Sone TRONDHEIM SENTRUM

310179_KV_H_Midtbyen

Maskin

cat 930 K
m/klappvinge



	Gatenavn		Lengde [km]
🟦	Arkitekt Christies gate		0,16
🟦	Bispegata	1	0,62
🟦	Batteriveita		0,05
🟦	Batterigata		0,09
🟦	Carl Johans gate	12	0,15
🟦	Ciss Kleins gate		0,06
🟦	Dronningens gate	5	0,93
🟦	Elvegata		0,46
🟦	Erling skakkes gate	2	1,14
🟦	Fjordgata	9	0,67
🟦	Fosenkaia	10	0,43
🟦	Gamle bybru		0,15
🟦	Hans Hagerups gate		0,14
🟦	Helmer lundgreens gate		0,08
🟦	Hospitalsgata		0,11
🟦	Hospitalsløkkan		0,35
🟦	Kalvskinnsgata		0,18
🟦	Kjøpmannsgata	4	1,13
🟦	Kongens gate	7	0,42
🟦	Kongsgårdsgata		0,29
🟦	Lillegata		0,06
🟦	Munkegata	8	0,70
🟦	Nordre gate	11	0,46
🟦	Schirmers gate		0,16
🟦	St. Olavs gate		0,21
🟦	Sverres gate		0,39
🟦	Søndre gate	6	0,33
🟦	Tordenskiolds gate		0,15
🟦	Vilhelm Storms gate		0,08
🟦	Voldgata		0,05
🟦	Vår Frue gate	14	0,10
	Tinghusgt.	3	
	Vår frues strete	13	
	Schultzgt.	15	
	Bryggegt.	16	
	Total lengde		**10,3 km**

TEGNFORKLARING

- Fylkesveg
- Kommunalveg
- Boliggate
- GS-veg
- Sykkelfelt
- Snø Snøopplager
- Sand Strøsandlager

Målestokk: 1:5000 (A3)

Tegnet av: asplan viak

Trondheim bydrift
for miljø og trivsel

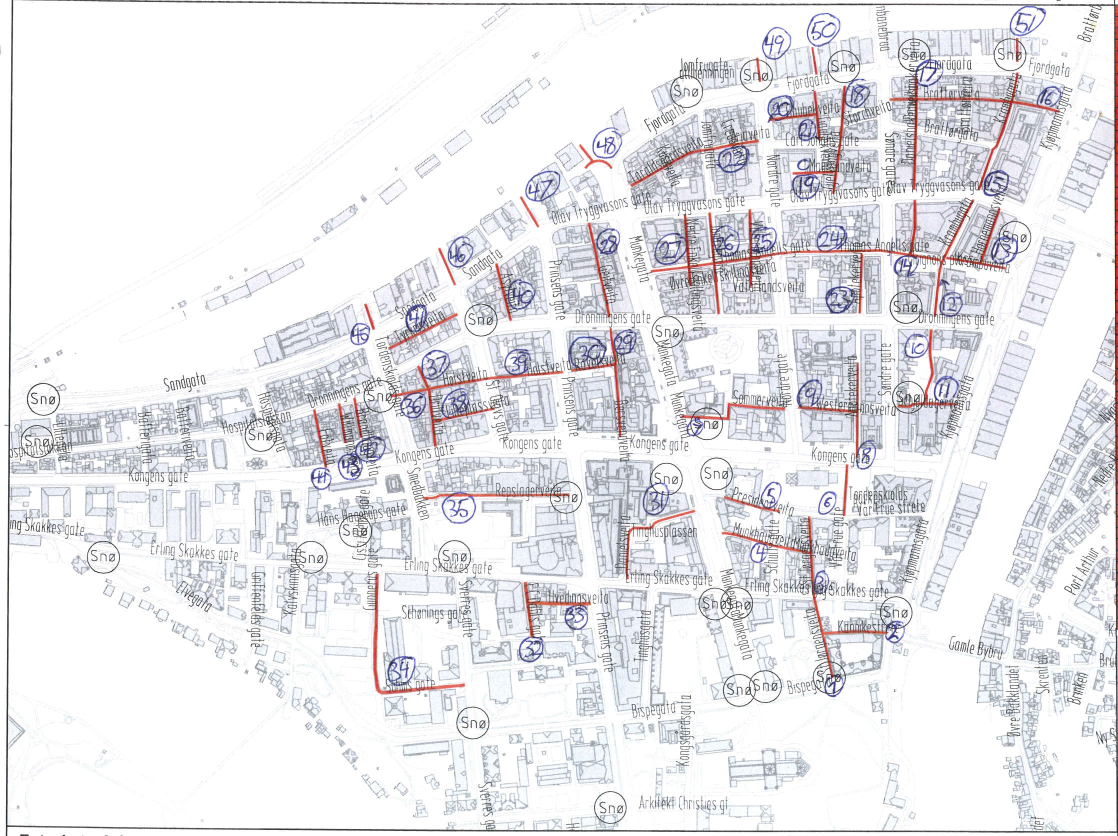
Koordinatsystem: EUREF89 UTM- SONE 32 (22)
Høydereferanse: NN 2000
Kartuttrekk pr dato: 06122013
Kilde: Trondheim kommune
Revideringsansvarlig: Anders Svanekil

Ved snødeponi i kryss bør siktforholdene vurderes.
Driftsklasse GsB: Kongsgårdsgata.

Revisjonsdato	**01.10.2014**
Revisjonsnr.	**02**
Tegningsnr.	**310179**

VEST - Sone TRONDHEIM SENTRUM

310174_KV_B_Midtbyen



Gatenavn		Lengde [km]
🟥	Apotekerveita	0,12
🟥	Apotekerveita fra Dronningens gt.	0,08
🟥	Asylveita	0,07
🟥	Bersvendveita	0,10
🟥	Brandhaugveita	0,05
🟥	Brattørveita	0,21
🟥	Danielsbakerveita	0,14
🟥	Danielveita	0,06
🟥	Drillveita	0,12
🟥	E.C Dahls gate	0,08
🟥	**Fjordgata 12-16**	0,03
🟥	**Fjordgata 42-46**	0,03
🟥	**Fjordgata 54-56**	0,03
🟥	**Fotveita**	0,07
🟥	**Gaubekveita**	0,08
🟥	**Gaubekveita til Nordre gate**	0,06
🟥	**Geitveita**	0,12
🟥	Gjelvangveita	0,14
🟥	Gunnerus gate	0,15
🟥	Holstveita	0,22
🟥	Hornemannsveita	0,08
🟥	Hvedningsveita	0,08
🟥	Jomfrugata	0,13
2	Kannikestrete	0,08
🟥	Kattveita	0,07
🟥	Kongens gate 16-18	0,06
🟥	Krambugata fra Dronningens gt.	0,32
🟥	Krambugata til Dronningens gt.	0,06
🟥	Krambuveita	0,07
🟥	Lilleplassveita	0,08
🟥	Moursundveita	0,05
🟥	Munkhaugveita	0,11
🟥	Nedre Enkeltskillingsveita	0,09
🟥	Peter Egges plass	0,03
🟥	Presidentveita	0,09
🟥	Prinsens gate 67	0,04
🟥	Ravelsveita	0,11
🟥	Ravnkloa	0,07
🟥	Repslagerveita	0,18
🟥	Sandgata 10-12	0,05
🟥	Sangata 24B	0,03
🟥	Schjoldagerveita	0,08
🟥	Sommerveita	0,09
1	St. Jørgensveita	0,21
🟥	Suhms gate	0,11
🟥	Taraldsgårdveita	0,10
🟥	Thomas Angells gate	0,37
🟥	Tinghusplassen	0,09
🟥	Tyrkrisveita	0,11
🟥	Vaterlandsveita	0,10
🟥	Vår Frue strete	0,17
🟥	Vår Frues gate	0,06
🟥	Westermannsveita	0,07
🟥	Willmannsveita	0,06
🟥	Ørjaveita	0,08

Total lengde **5,56 km**

Tegnet av: asplan viak

Trondheim bydrift *for miljø og trivsel*

Koordinatsystem: EUREF89 UTM- SONE 32 (22)
Høydereferanse: NN 2000
Kartuttrekk pr dato: 06122013
Kilde: Trondheim kommune
Revideringsansvarlig: Anders Svanekil

Evt. ekstra info.

MASKIN: CAT ~~T~~ 906

TEGNFORKLARING

🟩	Fylkesveg	🟥	Boliggate	Snø	Snøopplager
🟦	Kommunalveg	🟦	GS-veg	Sand	Strøsandlager
		🟧	Sykkelfelt		

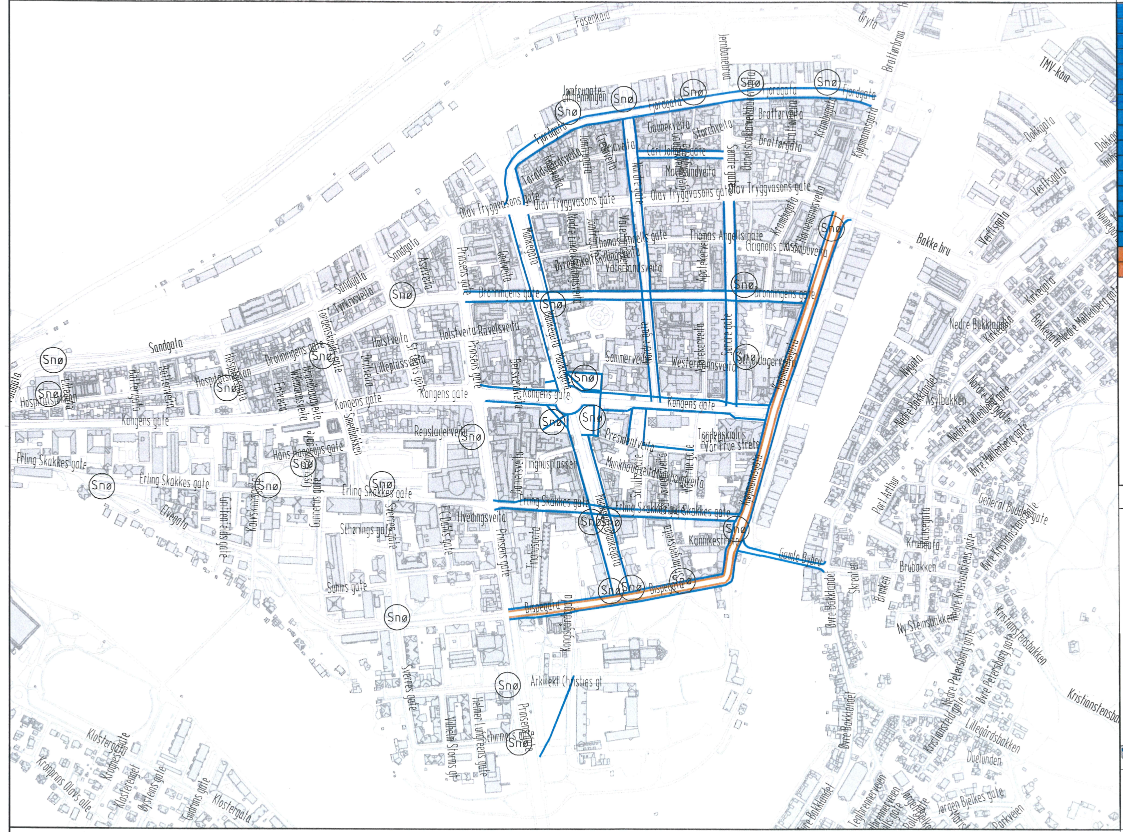
Målestokk: 1:4000 (A3)

[illegible]

Revisjonsdato	**01.10.2014**
Revisjonsnr.	**02**
Tegningsnr.	**310174**

VEST - Sone TRONDHEIM SENTRUM

310116_KV_GS_Midtbyen øst



	Gatenavn		Lengde [km]
🟦	Bispegata	2	0,35
🟦	Carl Johans gate	11	0,15
🟦	Dronningens gate	8	0,54
🟦	Erling skakkes gate	3	0,38
🟦	Fjordgata	12	0,67
🟦	Gamle Bybru	14	0,14
🟦	~~GS, Ilen kirke~~		~~0,16~~
🟦	Kjøpmannsgata	7	0,61
🟦	kongsgårdsplassen	1	0,14
🟦	Kongens gate	6	0,32
🟦	Munkegata	4	0,70
🟦	Nordre gate	10	0,46
🟦	Kongens gate 5-7		0,12
🟦	Søndre gate	9	0,33
🟦	Trondheim Torg	5	0,22
🟦	Vår Frue Strete	13	0,08
🟧	Bispegata		0,35
🟧	Kjøpmannsgata		0,59

Total lengde **6,30 km**

TEGNFORKLARING

🟩	Fylkesveg
🟦	Kommunalveg
🟥	Boliggate
🟦	GS-veg
🟧	Sykkelfelt
Snø	Snøopplager
Sand	Strøsandlager

Målestokk: 1:5000 (A3)

Tegnet av: asplan viak

Trondheim bydrift
for miljø og trivsel

Koordinatsystem: EUREF89 UTM- SONE 32 (22)
Høydereferanse: NN 2000
Kartuttrekk pr dato: 06122013
Kilde: Trondheim kommune
Revideringsansvarlig: Anders Svanekil

Ved snødeponi i kryss bør siktforholdene vurderes.
Driftsklasse GsB: Kongsgårdsplassen, Bispegata og Kjøpmannsgata.

Revisjonsdato	**01.10.2014**
Revisjonsnr.	**02**
Tegningsnr.	**310116**

GRAPHS OF THE PERFORMANCE OF THE MA WHEN APPLIED TO MAP DATA FROM TRONDHEIM

E.1 Route 310174_KV_B_Midtbyen

E.1.1 Graphs of Performance when Run for 100000 Generations

Figure E.1 Route 310174_KV_B_Midtbyen 100000 Generations - Average Best

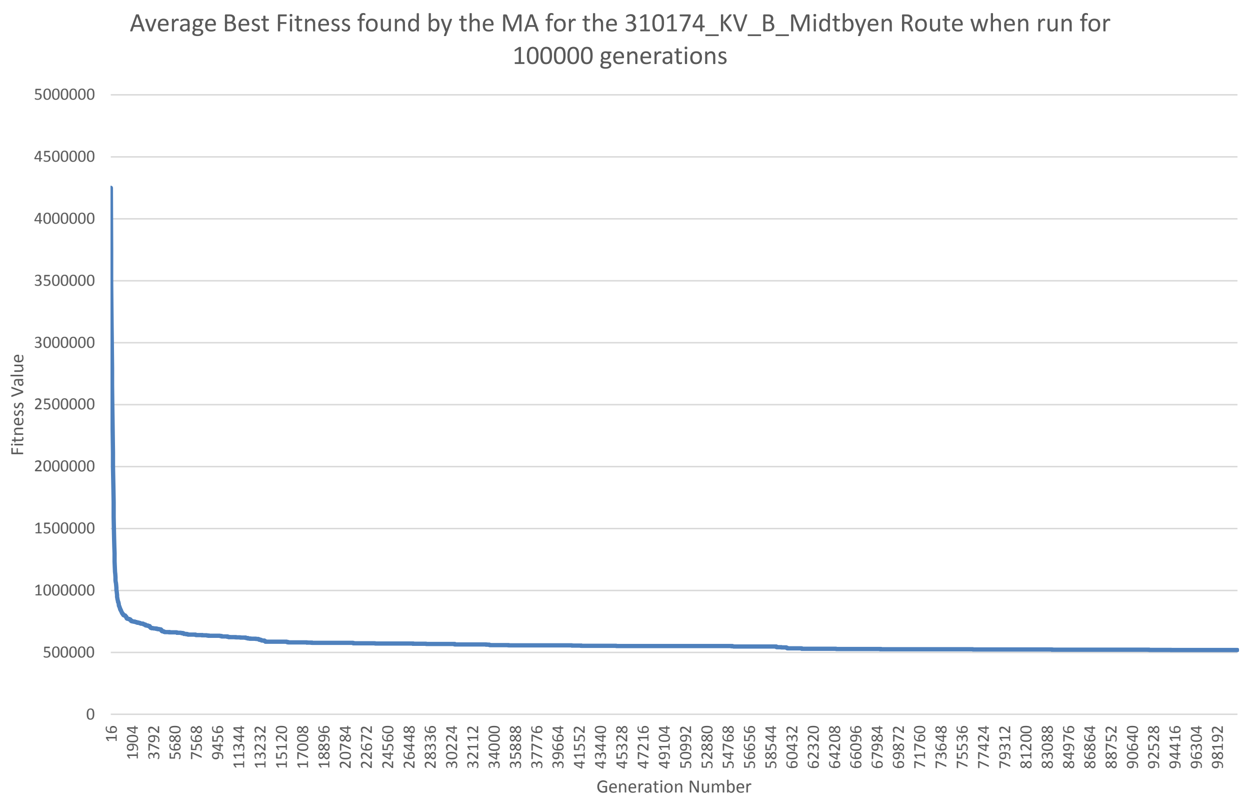


Figure E.2 Route 310174_KV_B_Midtbyen 100000 Generations - Average Average

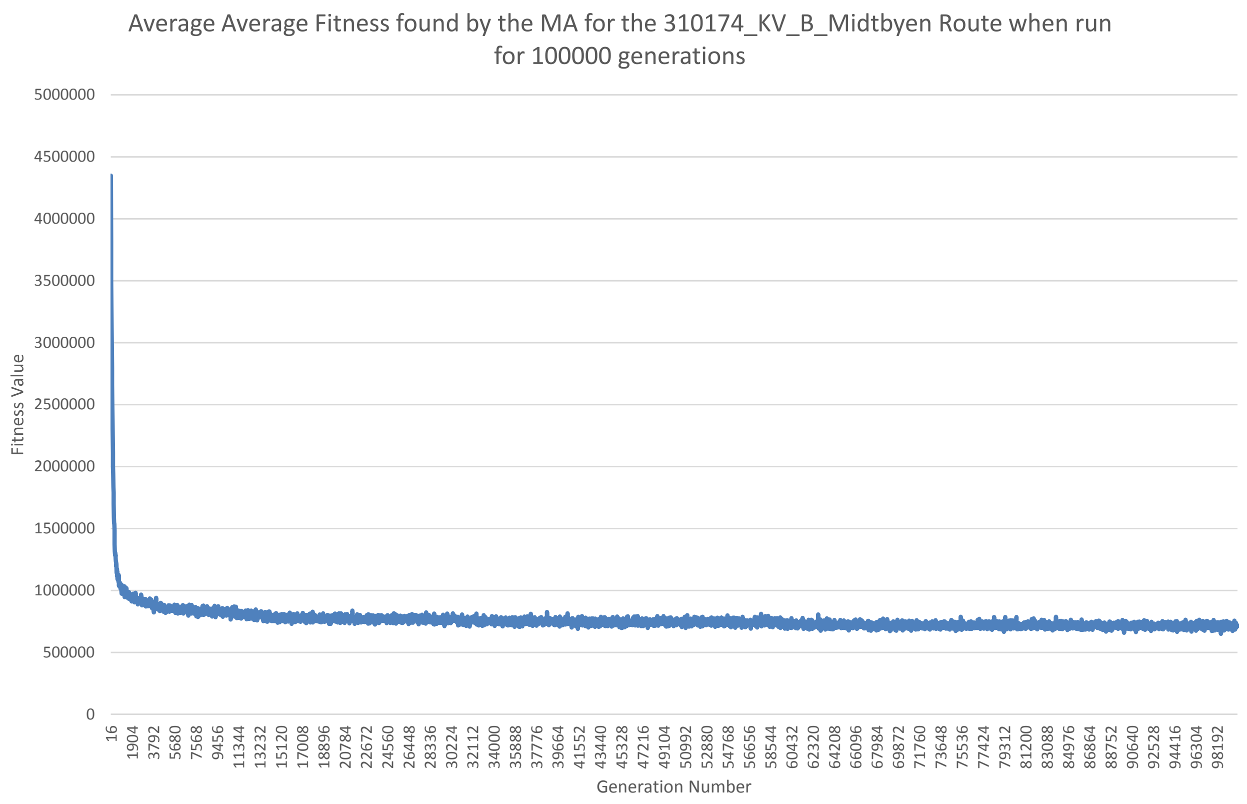
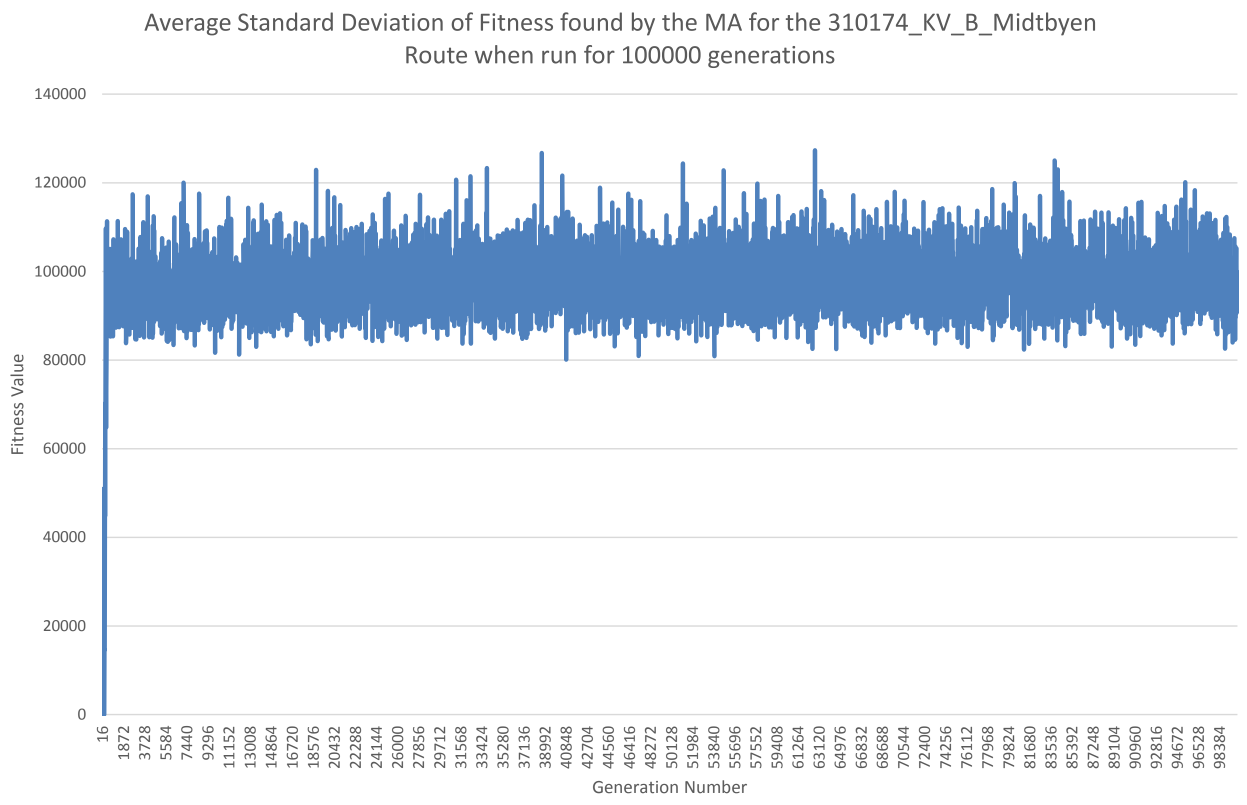


Figure E.3 Route 310174_KV_B_Midtbyen 100000 Generations - Average Standard Deviation



E.1.2 Graphs of Performance when Run for 50000 Generations

Figure E.4 Route 310174_KV_B_Midtbyen 50000 Generations - Average Best

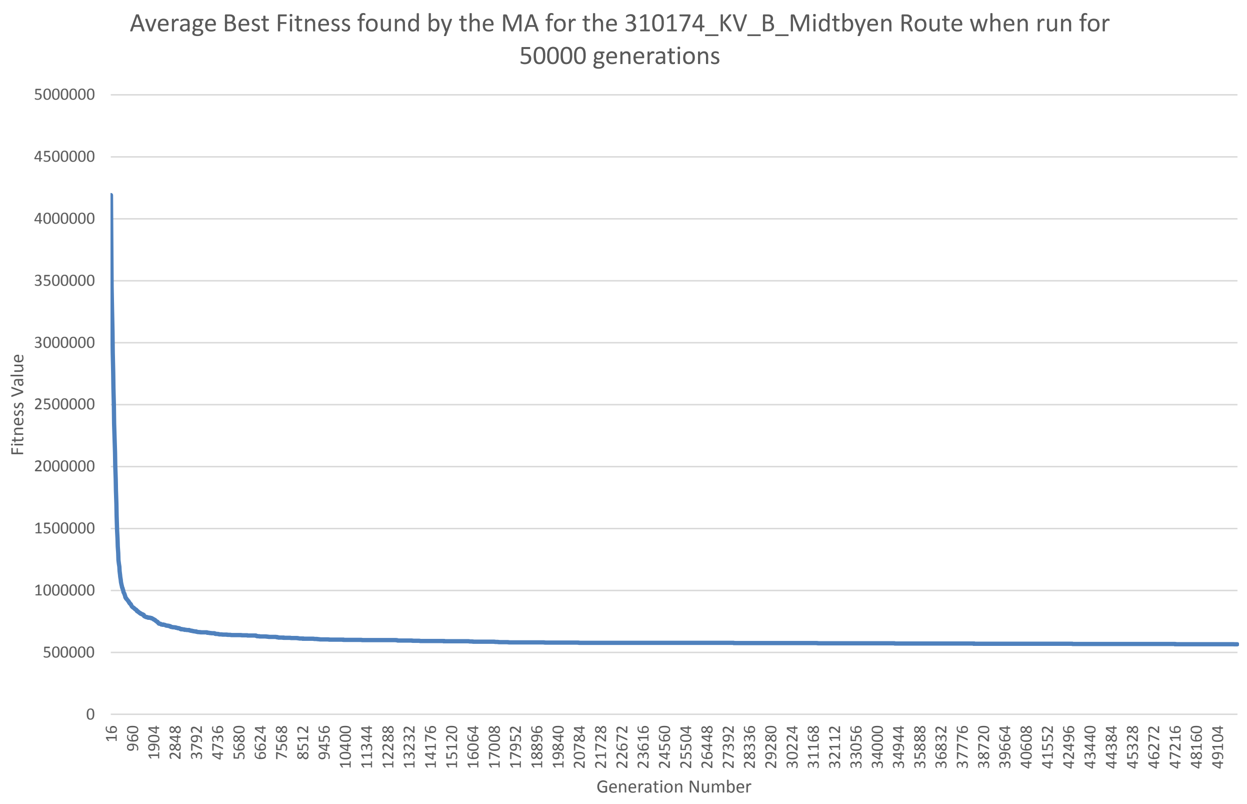


Figure E.5 Route 310174_KV_B_Midtbyen 50000 Generations - Average Average

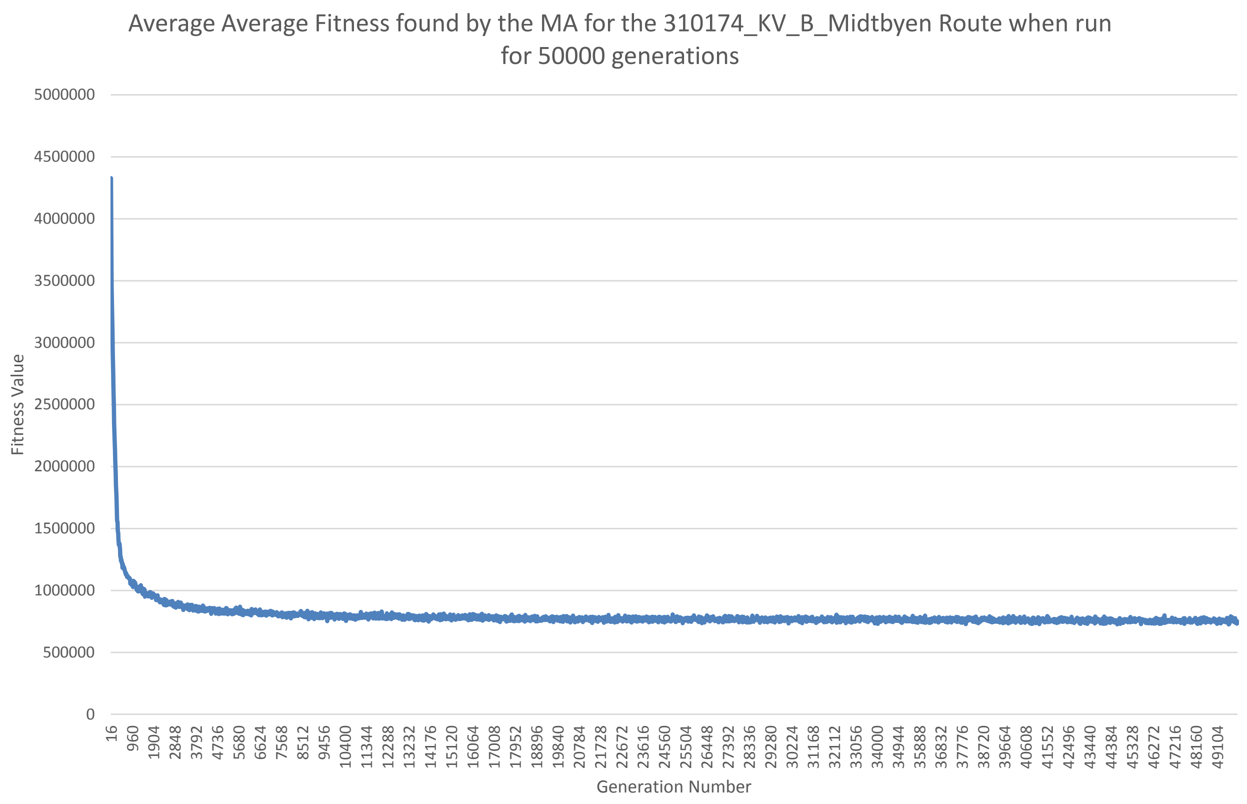
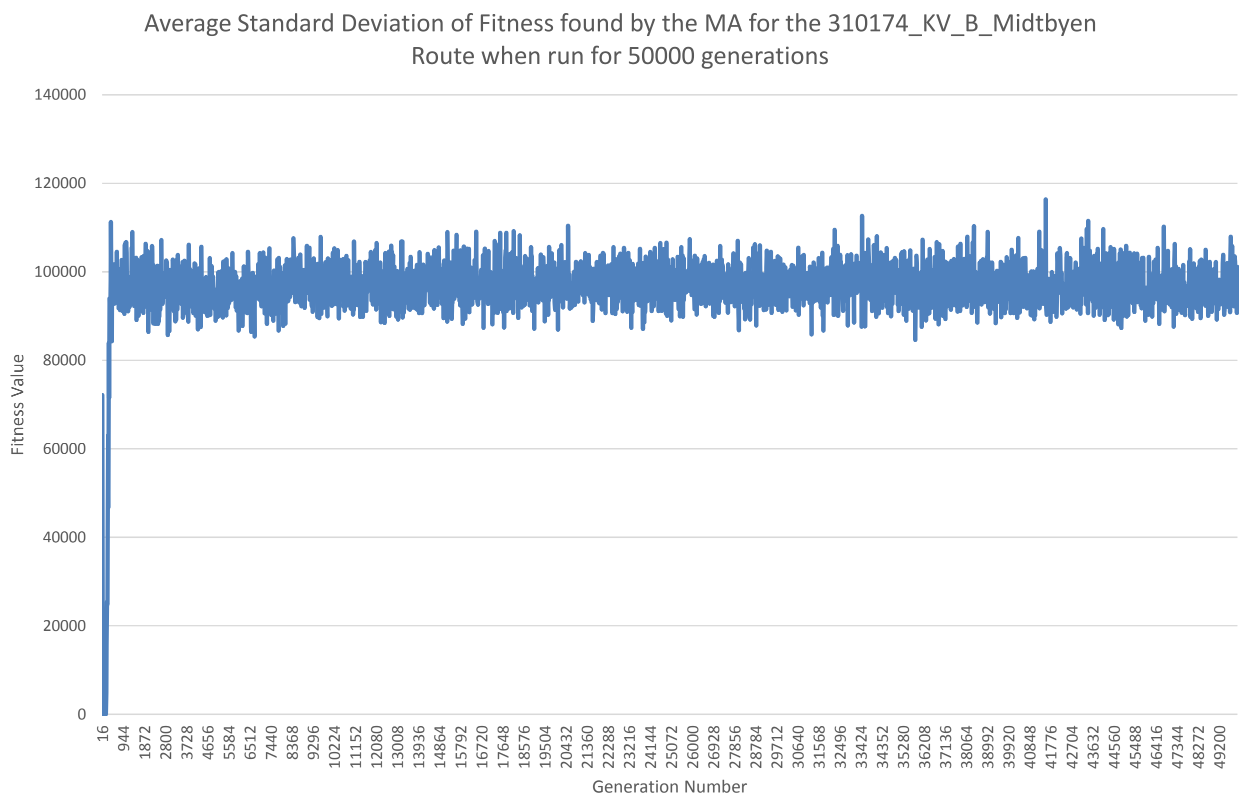


Figure E.6 Route 310174_KV_B_Midtbyen 50000 Generations - Average Standard Deviation



E.2 Route 310179_KV_H_Midtbyen

E.2.1 Graphs of Performance when Run for 100000 Generations

Figure E.7 Route 310179_KV_H_Midtbyen 100000 Generations - Average Best

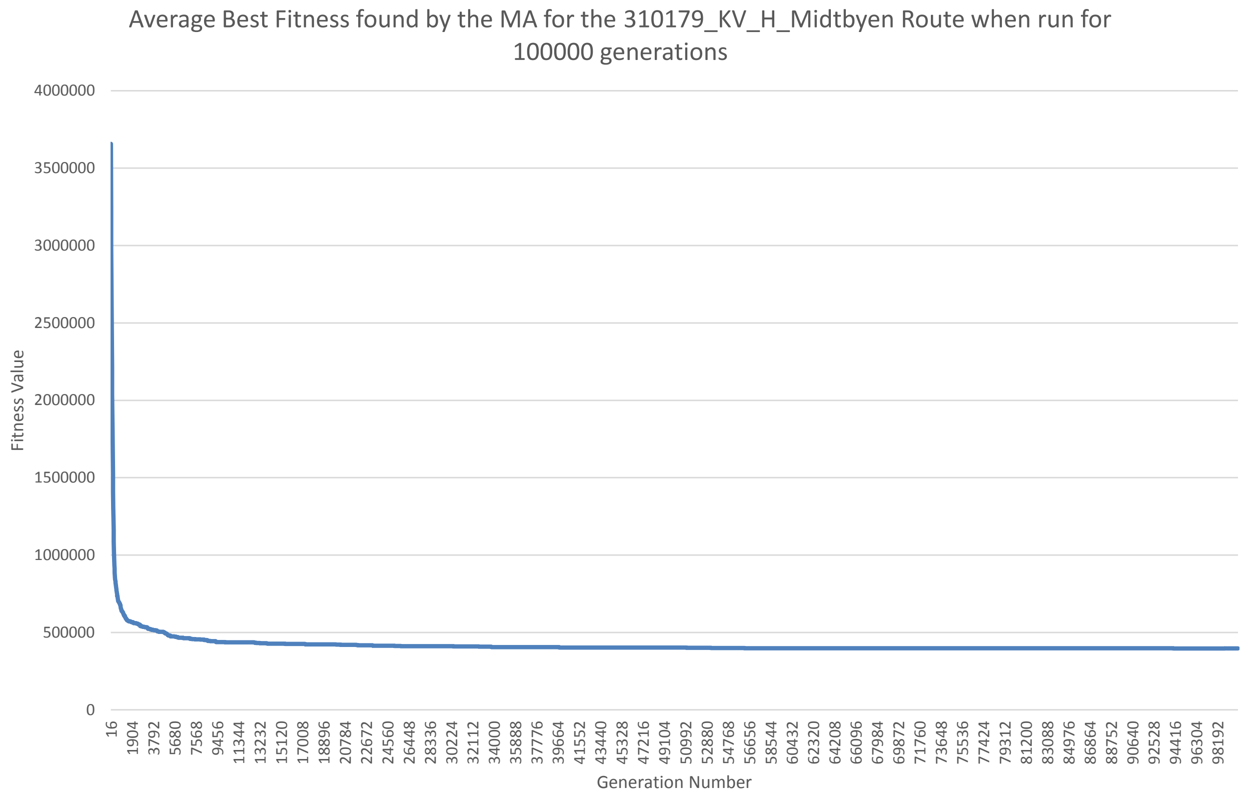


Figure E.8 Route 310179_KV_H_Midtbyen 100000 Generations - Average Average

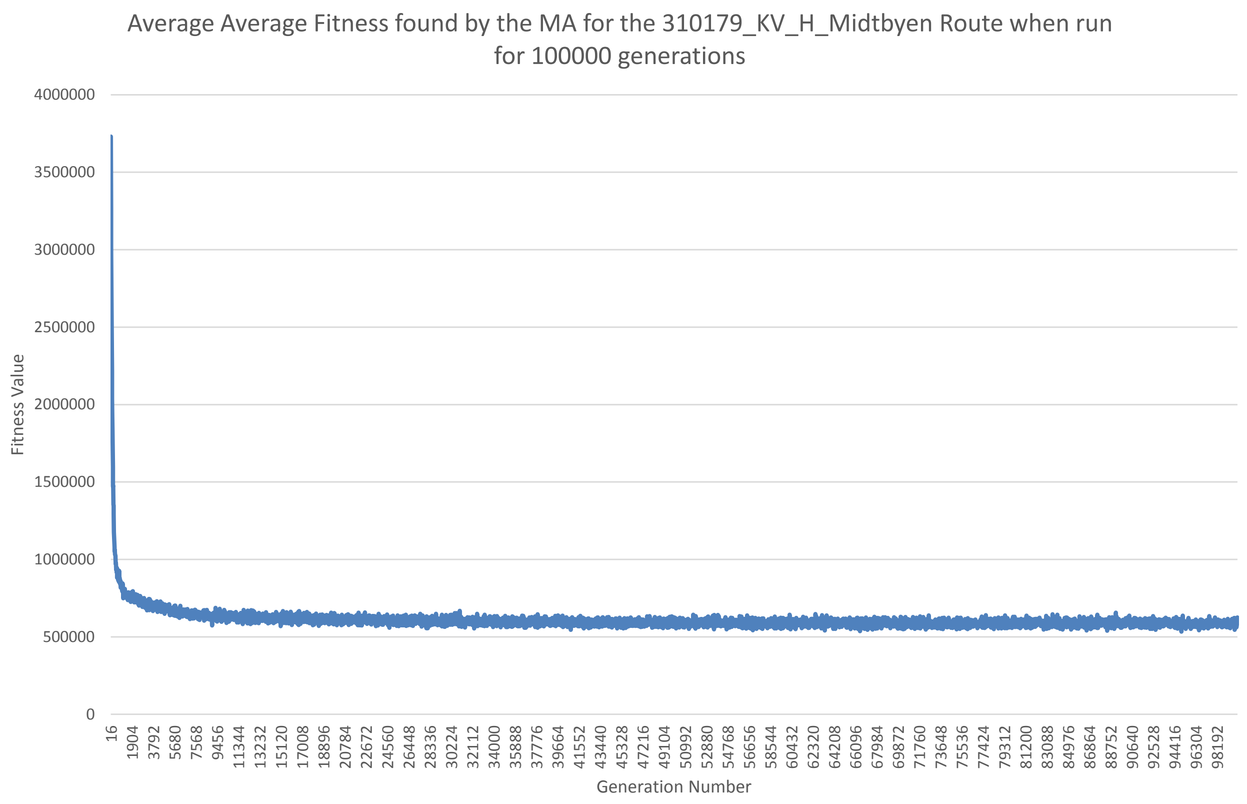
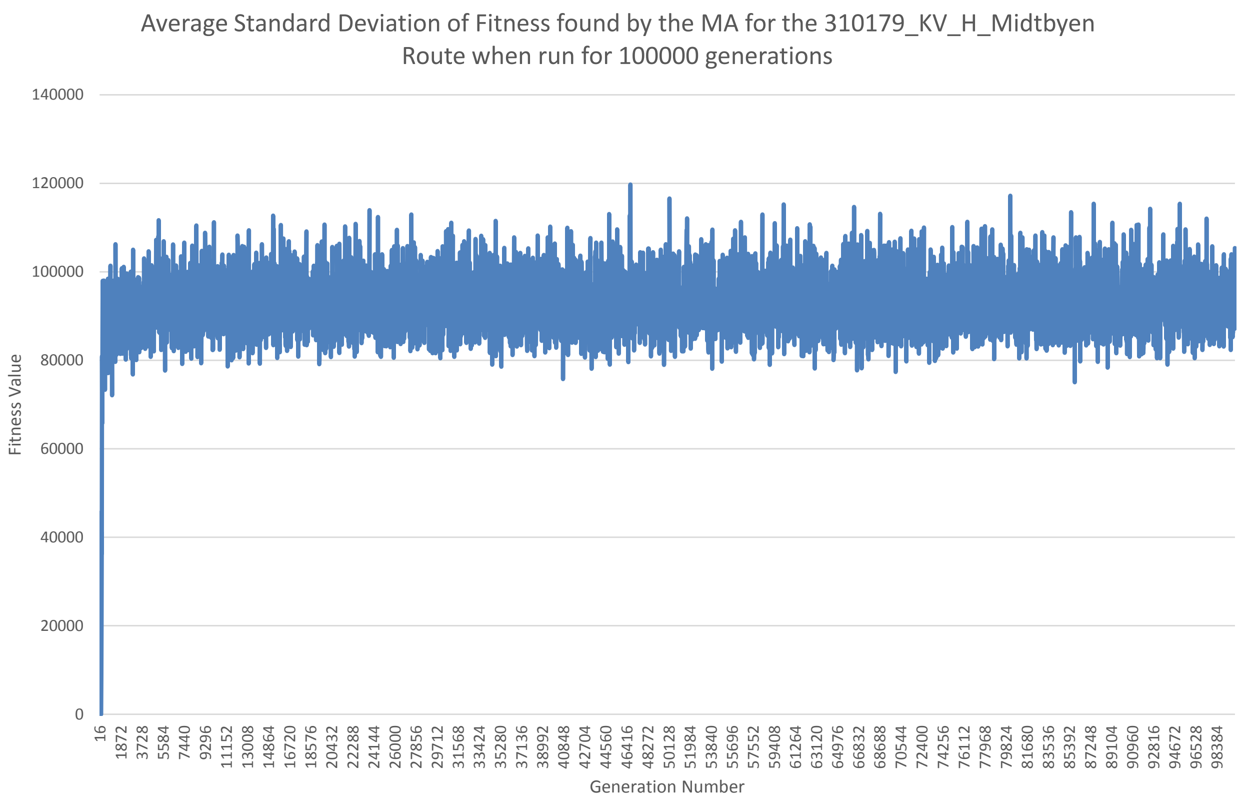


Figure E.9 Route 310179_KV_H_Midtbyen 100000 Generations - Average Standard Deviation



E.2.2 Graphs of Performance when Run for 50000 Generations

Figure E.10 Route 310179_KV_B_Midtbyen 50000 Generations - Average Best

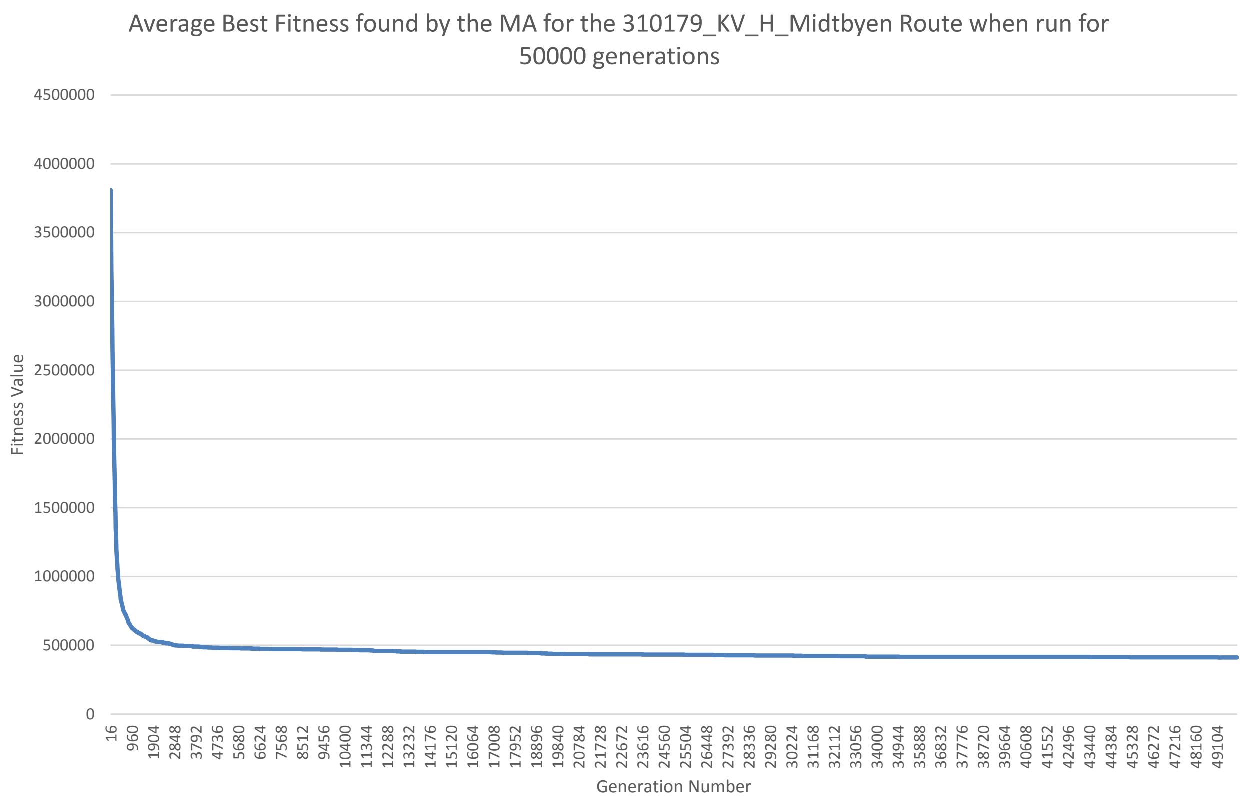


Figure E.11 Route 310179_KV_B_Midtbyen 50000 Generations - Average Average

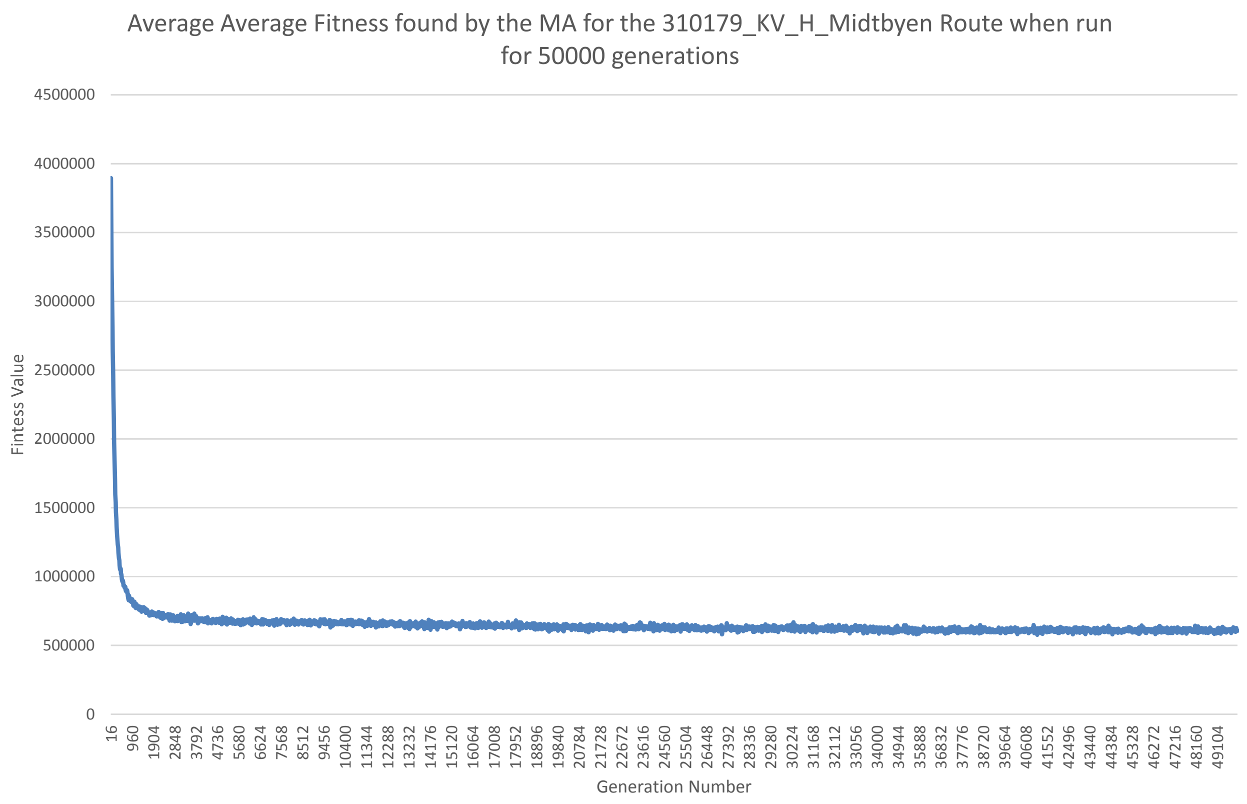
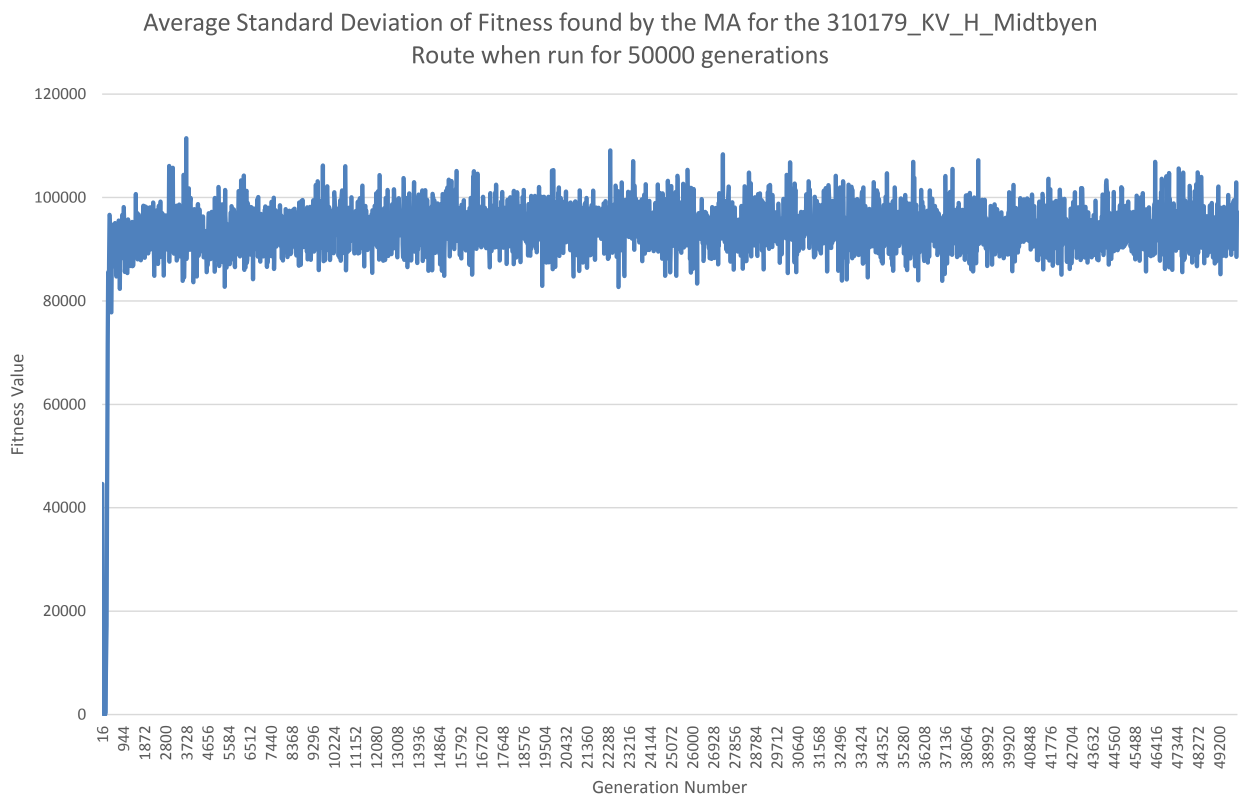


Figure E.12 Route 310179_KV_B_Midtbyen 50000 Generations - Average Standard Deviation



Appendix F

Pre-Project: Route Optimization for Winter Road Maintenance

Route Optimization for Winter Road Maintenance

Simon Glisic Randby
Magnus Solheim Thrap

Supervisor: Anders Kofod-Petersen

TDT4501 - Datateknologi, fordypningsprosjekt

December 17, 2014

Abstract

In this paper we present the current state of the art in route optimization for winter road maintenance. We found that an evolutionary approach is known to give good results for this kind of problem. Because the problem is NP-hard with the naturally occurring instances being of a vast size, tailoring the approaches to each instance produces the best results in the most time efficient matter.

Contents

List of Tables

1 Introduction

This paper is the result of a pre-project for a masters thesis in computer science at the Norwegian University of Science and Technology (NTNU). The goal is to lay the foundation for our further work, and obtain the necessary background material.

What we want to do is to explore how route optimization can help improve the work with winter road maintenance in the municipality of Trondheim in Norway. The problem is highly complex as the routing problem in itself is NP-hard and the relevant road network is huge.

1.1 Structure of this paper

After some initial work we found out that optimization of routes is a problem that there has been done a lot of work on, and exists in many different forms. There will first be given a brief historical background of the problem. As there is a lot of available material on the subject, so we did a structured literature review to help us find the most relevant information. We will present our literature search protocol along with what we found to be the state of the art on the subject. At the end of the paper we will give our suggestions for future work, that summarizes our most important findings on the subject and makes suggestions for what questions would be interesting to see more research on.

1.2 History

In the early 18th century the inhabitants of Königsberg debated whether it was possible to form a closed walk across all the seven bridges over the river Pregel, without crossing the same bridge twice. The problem is know as the “Seven bridges of Königsberg Problem”, and has historical importance both in mathematics and the study of arc routing problems.

In 1735, the Swiss mathematician Leonhard Euler presented his solution to the problem. He had generalized the problem and formulated it mathematically. The generalization is now known as the Eulerian path problem, and can be written as follows.

Given a graph G=(N,E), determine if there exists a cycle that passes through every edge in E exactly once.

A cycle that passes through every edge of a graph exactly once is called an Euler tour. Euler proved that an Euler tour exists if and only if every node of the graph has even degree (Wøhlk (2008) [30]).

The next big happening in arc routing was over 200 years later. In 1962 the chinese mathematician, and former postman, Mei-Ko Kwan extended Euler's problem to what is now known as the Chinese Postman Problem (CPP). It can be formulated as follows:

Given a graph G = (N,E,C), where C is the distance matrix for the edges, find a tour with minimal total distance that passes through each edge in E at least once (Wøhlk (2008) [30]).

Both Euler's problem and the CPP looks at graphs with respect to the edges, but in contrast to Euler's problem the CPP asks to find find the tour rather than determining whether it exists or not.

The CPP can be solved in polynomial time if the edges are either exclusively directed or undirected, but if the graph contains both directed and undirected edges the problem becomes NP-hard (Wøhlk (2008) [30]).

Many variants of the CPP have later been studied, and the field of arc routing has arisen.

2 Structured Literature Review Protocol

The following literature review is based on the method outlined in Kofod-Petersen (2014) [14].

2.1 Search Terms

We were considering our problem to be a variant of the Arc Routing problem. Based on this and the fact that our assignment is to be solved using artificial intelligence methods, the search terms given in Table 2 and Table 1 were chosen.

The reason two search term matrices were chosen was that we wanted material focusing specifically on route optimization concerning winter road maintenance (table 2) and literature regarding general arc routing problems from an AI standpoint (table 1).

	T1	**T2**	**T3**	**T4**
Search term	route	optimization	arc routing	Artificial intelligence
Synonym 1	path	finding	capacitated arc routing	ai
Synonym 2	navigation	optimal	undirected capacitated arc routing	genetic algorithm
Synonym 3			multi-depot arc routing	ga
Synonym 4			multi depot arc routing	local search
Synonym 5			chinese postman problem	global search
Synonym 6			CPP	
Synonym 7			rural postman	
Synonym 8			vehicle routing	
Synonym 9			multi-depot routing	
Synonym 10			multi depot routing	
Synonym 11			periodic vehicle routing	

Table 1: First search term matrix

	T1	**T2**	**T3**	**T4**
Search term	route	optimization	winter road maintenance	chinese post-man problem
Synonym 1	path	finding	snow	CPP
Synonym 2	navigation	optimal	plowing	arc-based
Synonym 3				vehicle rout-ing problem

Table 2: Second search term matrix

We constructed one search string for both of the term matrices above, using the following procedure:

1. Create a group for every term by combining the term and its synonyms with the OR operator.
2. Construct the final search query by combining the groups using the AND operator.

The resulting search strings were as follows.

Search string 1

```
(route OR path OR navigation) AND (optimization OR finding OR optimal)
AND (arc routing OR capacitated arc routing OR undirected capacitated
arc routing OR multi-depot arc routing OR multi depot arc routing OR
chinese postman problem OR CPP OR rural postman OR vehicle routing
OR multi-depot routing OR multi depot routing OR periodic vehicle routing)
AND (Artificial intelligence OR ai OR genetic algorithm OR ga OR local
search OR global search)
```

Search string 2

```
(route OR path OR navigation) AND (optimization OR finding OR optimal)
AND (winter road maintenance OR snow OR plowing) AND (chinese postman
problem OR CPP OR arc-based OR vehicle routing problem)
```

2.2 Search engines

Based on the recommendations of our supervisors, peers and Kofod-Petersen (2014) [14], we selected the search engines listed in table 3.

Engineering Village
IEEE
Web of Science
ScienceDirect
CiteSeer
Springer Link

Table 3: Search engines

Despite their recommendation, we chose not to use Google Scholar, ACM or Wiley, due to issues with reproducibility and lacking support for of searches consisting of logical operators (making it hard to create comparable and reproducible searches across the engines).

2.3 Inclusion and Quality criteria

To remove irrelevant papers, we used the inclusion and quality criteria listed in table 4 and table 5 respectively. We noticed that our search results contained a lot of papers focusing on unrelated types of route optimization (e.g. routing in discrete computer networks, routing in telecommunication networks and route finding in open environments without movement restrictions), so we focused on removing them as early as possible in the process.

Inclusion Criteria	Criteria
IC1	Paper must be in English
IC2	Paper must be finished before conclusion of the review (October 2014)
IC3	If there are duplicates from several search engines, only one instance of the paper is included
IC4	At least some of the search terms have to be in the title (doesn't necessarily have to have both routing and snow in title, but should have at least one)
IC5	Title must not be about (internet/telecom) network routing
IC6	Paper must be available to us digitally

Table 4: Inclusion criteria

Quality Criteria	Criteria
QC1	Paper should clearly manage to convey what it is about.
QC2	Paper should mention which algorithms were used if it mentions route optimization.
QC3	If the paper is about general route optimization, applications of what is discussed for winter road maintenance should be at least suggested.
QC4	If the paper is about winter road maintenance, optimization of routing should be mentioned.
QC5	Paper must to some extent cover route optimization (eg. no papers about how absence of winter road maintenance leads to less safety)
QC6	If paper is on routing in networks, the networks must resemble road networks. E.g. no routing in open planes, computer networks, etc.
QC7	When paper is on routing in traffic: Real time optimization of route to handle traffic flow, and trying to minimize congestion/take congestion into account is not relevant

Table 5: Quality criteria

2.3.1 Results

For details on the complete list of papers that were found, and what papers came through to each iteration of the work, see appendix A. After processing what remained, the following summary of the current state of the art and suggestion for future work was created.

6

3 State of the art

3.1 Edge routing problems

3.1.1 CPP

The Chinese Postman Problem (CPP) asks to find a closed route that goes through all the edges of a graph. When the graph is either completely directed or completely undirected, there exists polynomial time algorithms that solves the problem, however if the graph contains both directed and undirected edges the problem becomes NP-hard (Wøhlk (2008) [30]) (this is discussed in the section about Mixed CPP).

3.1.2 RPP

The rural postman problem (RRP) is a variation of the CPP, where not all the edges have to be serviced (Algorithms for the rural postman problem, 1995). In the RPP there can be both edges and arcs (undirected/directed connections between the nodes), and the nonrequired edges can be traversed (usually at a cost) to reach the edges that have to be services.

Although algorithms to solve the RPP optimally exist, the problem has been shown to be NP-complete if the required set of edges do not form a weakly connected network, which makes finding the exact solutions not feasible for practical applications (Pearn et. al (1995) [22]). However, it is possible to obtain good approximate solutions to the RRP using heuristic algorithms.

In practice, the RPP can be used to model for an instance snow removal on secondary roads, such as sidewalks or bicycle paths (Holmberg et. al (2010) [13]).

3.1.3 HCPP

When the edges of the graph is partitioned into a hierarchy of clusters $(E_1, .., E_n)$, where every single edge in E_p must be served before E_{p+1} is started, we get the hierarchical chinese postman problem (HCPP). Looking at snow plowing, the HCPP approach is relevant if a set of streets has a higher priority than others.

For the special case of HCPP, where all the clusters are connected and there exists a linear relation specifying the order of which the clusters are to be traversed, Ghiani et. al (2000) [11] introduces a polynomial time algorithm to get the optimal solution.

For general HCPPs, Cabral et. al (2004) [7] introduces procedures for transforming the HCCP into a RPP. The RPP can be solved using approximation algorithms as discussed in the RPP section above, and the paper shows that a solution to the RPP can then be transformed back to a solution of the HCCP.

HCPP can be approximated directly using heuristic algorithms. Perrier et. al (2008) [25] presents a model and two heuristic approaches based on mathematical optimization to solve the problem. The paper discusses topics such as turn restrictions, load balancing, tandem service and deadheading time, which are realistic constraints in a real world HCPP problem.

3.1.4 MCPP

The mixed chinese postman problem (MCPP) is when you want to solve the CPP in a graph with both directed and undirected edges (Pearn et.al (1995) [23]). MCPP is of interest because route optimization problems in practice often take place in mixed networks, e.g. road networks where there exists roads that can be traversed in any direction and roads that are one way only.

Pearn et.al (1995) [23] presents the MIXED algorithms, that convert MCPP into graphs with undirected edges and even degree to obtain approximate solutions. They suggest modifications that significantly improve the quality of the solutions, however they do not take into account some factors that might affect the goodness of the routes (such as fleet sizing, different vehicles, etc).

Yaoyuenyong et. al (2002) [31] used a heuristic based on Minimum Cost Flow to improve on Pearn et.al (1995) [23] versions of the MIXED algorithms, which they called Shortest Additional Path Heuristic (SAPH). They found that SAPH performed better than all other heuristic improvements of MIXED that they tested against.

3.1.5 WPP

The windy postman problem (WPP) is concerned with graphs where the same edge can have varying cost for each time you traverse it (Dussault et. al 2013 [10]). This is seen in problems where servicing a road takes a lot more capacity than simply passing through it (which might be done on the way back, or to access other roads in the network) for an instance.

Dussault et. al (2013) [10] proposed an algorithm that found near-optimal solutions to the WPP in networks with a relatively low number of nodes.

The Min-Max k-Chinese Postman Problem looks at connected undirected graphs, where one of the nodes of the graph serves as a depot node. The task is to find the k tours that starts and ends in the depot node, such that all the edges in the graph are served and the length of the longest tour is minimized (Wøhlk (2008) [30]).

Ahr et. al (2001) [1] looks at this variation of the CPP. They introduce two heuristics for the problem, giving lower bounds to help further increase of heuristic quality.

3.1.6 ARP

The term arc routing problem (ARP), is used to describe problems where the main goal is to service arcs in graphs modelling networks used for transport. It is strongly related to the CPP (some argue that variations of the CPP fall in under ARPs, other that ARPs are special cases of the CPP). But in general as ARPs focus more on transportation networks and vehicles, it therefore also has close ties to the vehicle routing problem (VRP). The terminology when describing ARPs tend to be more focused on vehicles and roads than what is the case with the special cases of the CPP that often refer to edges and postmen (Hertz (2005) [12], Assad et. al (1995) [2]).

3.1.7 CARP

A Capacitated Arc Routing Problem (CARP) is a problem where the demands are placed on the edges of the graph. The edges are to be serviced by a given number of identical vehicles with a given capacity, initially located at a node that serves as a depot. The problem's objective is to find the tours that minimizes the total cost, under the conditions that each arc with positive demand is served by one, and only one, vehicle and no vehicle serves a larger demand than it's capacity (Wøhlk (2008) [30]).

Since CARP has been proven to be NP-hard (Wøhlk (2008) [30]), only small problem instances can be solved optimally in a reasonable amount of time. Belenguer et. al (1998) [3] gives an approach to finding optimal solutions for problem instances consisting of 7 or less vehicles. The algorithm performs best when the numbers of vertices is between 24 to 41. An algorithm later introduced by Belenguer et. al (2003) [5] further increased the number of vertices to 50 and edges to 97.

As it is infeasible to find optimal solutions for large problem instances, several metaheuristic algorithms has been introduced. Brandão et. al (2008) [6] proposes a tabu search algorithm to approximate a solution to CARP. The paper shows that a pretty straightforward implementation of a tabu search algorithm can give high quality solutions to CARP in an efficient manner. Santos (2010) [29] presents an ant colony optimization based metaheuristic to solve the CARP. The presented algorithm produced good solutions, in terms of both solution quality and running time, compared to state of the art metaheuristic approaches. Other metaheuristics are reported to produce solutions of similar quality, as for instance the Greedy Randomized Adaptive Search Procedure with path-relinking presented by Luiz et. al (2013) [19], which gives solutions comparable to solutions of tabu search and ant-colony optimization, but unfortunately the algorithm is outperformed when it comes to execution speed. Rao et. al (2011) [28] shows that A* can be used find approximate solutions to CARPs. The paper used A* to find routes for snow-plows, and routes found by the algorithm performed much better than the routes that were already in place in the municipalities they worked with. However they had a significant tradeoff between their implementations ability to find solutions fast, and finding good solutions. Test results were also only presented for relatively small graphs (less than 25 vertices and less than 75 nodes).

Christiansen et. al (2009) [9] approached CARP with a probabilistic approach. The demand on each edge is modelled by a random variable and the goal is to find the routes with the least expected cost, under the constraint that the vehicle capacity is not exceeded. The paper formulates the problem as a set partitioning problem and solves it with a branch-and-price algorithm.

Evolutionary computation approaches have been reported to perform even better than the approaches mentioned above. Lacomme et. al (2001) [18] looks at how to solve the CARP with a genetic algorithm (GA). They give a good overview of the data structure they used, and details on the implementation of the algorithm. At the time the paper was published, their implementation performed better than the current best known approach (which was tabu-search), and improved some of the best known solutions. An important factor for their performance was that they used local search for the mutation operator. When doing the mutation of each child, they did a local search to determine which of the possible mutations gave the best child, and used this mutation. Lacomme et. al (2006) [16] looks into optimizing both shortest total distance and also minimizing total spent time the longest route.

A variation of CARP worth mentioning is one where an edge can be serviced by several vehicles. That is, if a vehicle runs out of capacity while serving an edge, another vehicle can take over and finish the rest of it. Belenguer et. al (2010) [4] looks at this variant of CARP and introduces a cutting-plane algorithm to find the lower bounds, and presents an evolutionary local search approach to find the upper bounds.

3.1.8 ECARP

The extended capacitated arc routing problem (ECARP) looks at the same problem as the CARP, but with additional constraints. It takes into account that the graph can have both directed and undirected edges (that can be parallel), that servicing and passing through edges have different costs, that some some turns in nodes are penalized and others are forbidden (such as turning left and doing u-turns), and that each trip can not be longer than a certain threshold (because the servicing vehicles can not perform unlimited servicing per trip) (Lacomme et. al (2004) [17]).

Lacomme et. al (2004) [17] approached the ECARP with a Memetic Algorithm (MA), which they found performed really well. At the time of its publication the algorithm found all the already best known solutions, improved several of them, and found one tight lower bound that had not yet been found.

Lacomme et. al (2004) [17] is also a very useful paper for future work because they have a good documentation of relevant implementation details such as the genetic encoding, the calculation of the heuristics, how the local-search mutation component was done, and thorough documentation of the performance.

3.1.9 VRP

The Vehicle routing problem (VRP) is the converse of the ARP, where vehicles service the nodes of a transport network instead of the edges as they do in ARPs. One can find applications where a problem can be expressed as either an ARP or VRP (such as snow plowing), and where problems have been viewed as a combination of ARPs and VRPs (Prins et. al (2005) [27]).

Prins et. al (2005) [27] wanted to model garbage collection, and found that it is necessary to service both edges (undirected and directed) and nodes. Therefore they use a more general problem category that they call Node, Edge, and Arc Routing Problem (NEARP), which essentially is the combination of the VRP and ARP. To solve it they used a MA, which they argue performs comparable to other heuristic methods used for the ARP and VRP.

3.2 Sectoring

Sectoring is concerned with partitioning a graph into smaller subgraphs, to reduce problem sizes, and make management practical (Mourão et. al (2009) [20]). When doing this, it is important to keep in mind that it should be possible to solve the ARP in each sector (i.e. each sector should be a connected graph), and that it might be important to ensure that each sector has certain facilities, such as depots or garages for an instance (Labelle et. al (2002) [15]). The Sectoring Arc Routing Problem (SARP) a mix of CARP and Sectoring problems, and its goal is to partition a graph into smaller subgraphs, called sectors, in such a way that the total cost of all the sectors are minimized and that each sector can be serviced by a single vehicle (Heuristic methods for the sectoring arc routing problem). To find the cost of a sector a CARP must be solved, and the problem is thus reducible to CARP when only one sector is used. SARP can be approximated using heuristics, and Mourão et. al (2009) [20] proposes three heuristics to aid this problem.

In winter road maintenance the existence of e.g. snow disposal sites and vehicle depots, and the fact that some parts of the road network can be serviced by private companies makes sectoring desirable. Perrier et. al (2006) [26] and Perrier et. al (2006) [24] focuses on the system design process related to snow plowing operations. Amongst the processes introduced in the papers, they present algorithms to partition the road network into strategic sectors for snow disposal sites and vehicle depots. Methods considering road network partitioning, utilizing depot locations as the main concern, is also introduced in Muyldermans et. al (2002) [21]. Labelle et. al (2002) [15] discusses techniques for snow removal after the snow plowing vehicles have completed their tasks. It presents a decision support system for designing sectors for snow removal operations with various relevant constraints.

3.3 Winter Road Maintenance

Winter road maintenance is at its core about servicing infrastructure. There are many things to take into account, such as equipment, storage, and road safety concerns. However, due to that what being serviced are roads, and that they can very effectively be modelled as graphs (and often already are), the CPP, ARP, and their derivatives are good approaches to improve current several aspects of winter road maintenance.

Campbell et. al (2005) [8] offers a good and concise overview of concerns in winter road maintenance beyond just routing, and gives examples from work done in Montreal, in 1994.

4 Future Work

Because ARPs are NP-hard, it is not feasible to find exact/optimal solutions for large (realistic) problems instances. Winter road maintenance is usually done on road networks of significant size, therefore when we model winter road maintenance as an ARP finding approximate solutions with heuristic algorithms seem the best choice for finding good solutions within reasonable time.

To be able to use algorithms based on heuristics as efficiently as possible, we need good heuristic functions, both for estimating the goodness of partial solutions, and to give an idea of how far off a given solution is of an ideal solution. Finding good estimates that are computationally cheap to obtain, and simple to implement, is therefore of vital importance to the field, and efforts should be made to improve what are currently the best known approaches.

For practical reasons, both to reduce the problem size, and to facilitate administration when solutions are applied to cases, sectoring is an important issue. If one can improve how the problems are sectored one can make better and more efficient routes.

There are various methods for finding good solutions to ARP, but the ones that have shown the most promising results are evolutionary algorithms, especially GAs and MAs. When trying to solve new problem instances or improve on existing solutions using GAs or MAs should be strongly considered.

There has been done a lot of work on finding general solutions to ARPs, and good attempts at using this to solve instances of ARPs in fields where it is applicable, such as winter road maintenance. However, as solutions tailored to a specific instance do not translate to work ideally for other instances, there is a lot of potential in making systems that are easy to tailor to concrete instances, or creating new solutions for cases that have not yet been attempted optimized, or cases solutions have been found for.

References

[1] Dino Ahr and Gerhard Reinelt. *New Heuristics and Lower Bounds for the Min-Max k-Chinese Postman Problem*, volume 2461 of *Lecture Notes in Computer Science*, book section 10, pages 64–74. Springer Berlin Heidelberg, 2002.

[2] Arjang A. Assad and Bruce L. Golden. *Chapter 5 Arc routing methods and applications*, volume Volume 8, pages 375–483. Elsevier, 1995.

[3] J. M. Belenguer and E. Benavent. The capacitated arc routing problem: Valid inequalities and facets. *Computational Optimization and Applications*, 10(2):165–187, 1998.

[4] Jose-Manuel Belenguer, Enrique Benavent, Nacima Labadi, Christian Prins, and Mohamed Reghioui. Split-delivery capacitated arc-routing problem: Lower bound and metaheuristic. *Transportation Science*, 44(2):206–220, 2010. Times Cited: 8 Belenguer, Jose/L-3049-2014 0 9.

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

[6] José Brandão and Richard Eglese. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112–1126, 2008.

[7] Edgar Alberto Cabral, Michel Gendreau, Gianpaolo Ghiani, and Gilbert Laporte. Solving the hierarchical chinese postman problem as a rural postman problem. *European Journal of Operational Research*, 155(1):44–50, 2004.

[8] James F. Campbell and André Langevin. Operations management for urban snow removal and disposal. *Transportation Research Part A: Policy and Practice*, 29(5):359–370, 1995.

[9] Christian H. Christiansen, Jens Lysgaard, and Sanne Wøhlk. A branch-and-price algorithm for the capacitated arc routing problem with stochastic demands. *Operations Research Letters*, 37(6):392–398, 2009.

[10] Benjamin Dussault, Bruce Golden, Chris Groër, and Edward Wasil. Plowing with precedence: A variant of the windy postman problem. *Computers & Operations Research*, 40(4):1047–1059, 2013.

[11] G. Ghiani and G. Improta. An algorithm for the hierarchical chinese postman problem. *Operations Research Letters*, 26(1):27–32, 2000. 6484385 hierarchical Chinese postman problem arc partitioning connected arc clusters linear precedence relation road snow control road ice control optimal torch path flame cutting NP-hard problem polynomial-time solution computational effort arc routing.

[12] Alain Hertz. *Recent Trends in Arc Routing*, volume 34 of *Operations Research/Computer Science Interfaces Series*, book section 9, pages 215–236. Springer US, 2005.

[13] Kaj Holmberg. Heuristics for the rural postman problem. *Computers & Operations Research*, 37(5):981–990, 2010.

[14] Anders Kofod-Petersen. How to do a structured literature review in computer science, 2014-08-13 2014.

[15] A. Labelle, A. Langevin, and J. F. Campbell. Sector design for snow removal and disposal in urban areas. *Socio-Economic Planning Sciences*, 36(3):183–202, 2002.

[16] P. Lacomme, C. Prins, and M. Sevaux. A genetic algorithm for a bi-objective capacitated arc routing problem. *Computers & Operations Research*, 33(12):3473–3493, 2006. Times Cited: 31 0 33.

[17] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1-4):159–185, 2004.

[18] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Chérif. *A Genetic Algorithm for the Capacitated Arc Routing Problem and Its Extensions*, volume 2037 of *Lecture Notes in Computer Science*, book section 49, pages 473–483. Springer Berlin Heidelberg, 2001.

[19] Fabio Luiz Usberti, Paulo Morelato Franca, and Andre Luiz Morelato Franca. Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers and Operations Research*, 40(12):3206–3217, 2013. Compilation and indexing terms, Copyright 2014 Elsevier Inc. 20134316901714 Arc routing Infeasible solutions Meta heuristics Path relinking Reactive parameters.

[20] Maria Cândida Mourão, Ana Catarina Nunes, and Christian Prins. Heuristic methods for the sectoring arc routing problem. *European Journal of Operational Research*, 196(3):856–868, 2009.

[21] L. Muyldermans, D. Cattrysse, D. Van Oudheusden, and T. Lotan. Districting for salt spreading operations. *European Journal of Operational Research*, 139(3):521–532, 2002.

[22] W. L. Pearn and T. C. Wu. Algorithms for the rural postman problem. *Computers & Operations Research*, 22(8):819–828, 1995.

[23] Wen Lea Pearn and C. M. Liu. Algorithms for the chinese postman problem on mixed networks. *Computers & Operations Research*, 22(5):479–489, 1995.

[24] N. Perrier, A. Langevin, and J. E. Campbell. A survey of models and algorithms for winter road maintenance. part ii: system design for snow

disposal. *Computers & Operations Research*, 33(1):239–262, 2006. Times Cited: 10 0 10.

[25] Nathalie Perrier, Andre Langevin, and Ciro-Alberto Amaya. Vehicle routing for urban snow plowing operations. *Transportation Science*, 42(1):44–56, 2008. Compilation and indexing terms, Copyright 2014 Elsevier Inc. 20094812494881 Arc routing Chinese postman problem Different services Disposal sites Heuristic solutions Load-Balancing Makespan objective Mathematical optimizations Multicommodity network flow Operational constraints Precedence relations Real-life applications Road segments Service requirements Snow plowing Snow removal Solution strategy Urban areas Winter road maintenance.

[26] Nathalie Perrier, André Langevin, and James F. Campbell. A survey of models and algorithms for winter road maintenance. part i: system design for spreading and plowing. *Computers & Operations Research*, 33(1):209–238, 2006.

[27] Christian Prins and Samir Bouchenoua. *A Memetic Algorithm Solving the VRP, the CARP and General Routing Problems with Nodes, Edges and Arcs*, volume 166 of *Studies in Fuzziness and Soft Computing*, book section 4, pages 65–85. Springer Berlin Heidelberg, 2005.

[28] T. M. Rao, Sandeep Mitra, James Zollweg, and Ieee. *Snow-Plow Route Planning using AI Search*, pages 2791–2796. IEEE International Conference on Systems Man and Cybernetics Conference Proceedings. 2011. Times Cited: 1 SMC IEEE International Conference on Systems, Man and Cybernetics (SMC) OCT 09-12, 2011 Anchorage, AK IEEE; IEEE Syst, Man & Cybernet Soc; IEEE Circuits & Syst Soc (CAS); IEEE Engn, Med & Biol Soc (EMB) 0 1.

[29] Luís Santos, João Coutinho-Rodrigues, and John R. Current. An improved ant colony optimization based algorithm for the capacitated arc routing problem. *Transportation Research Part B: Methodological*, 44(2):246–266, 2010.

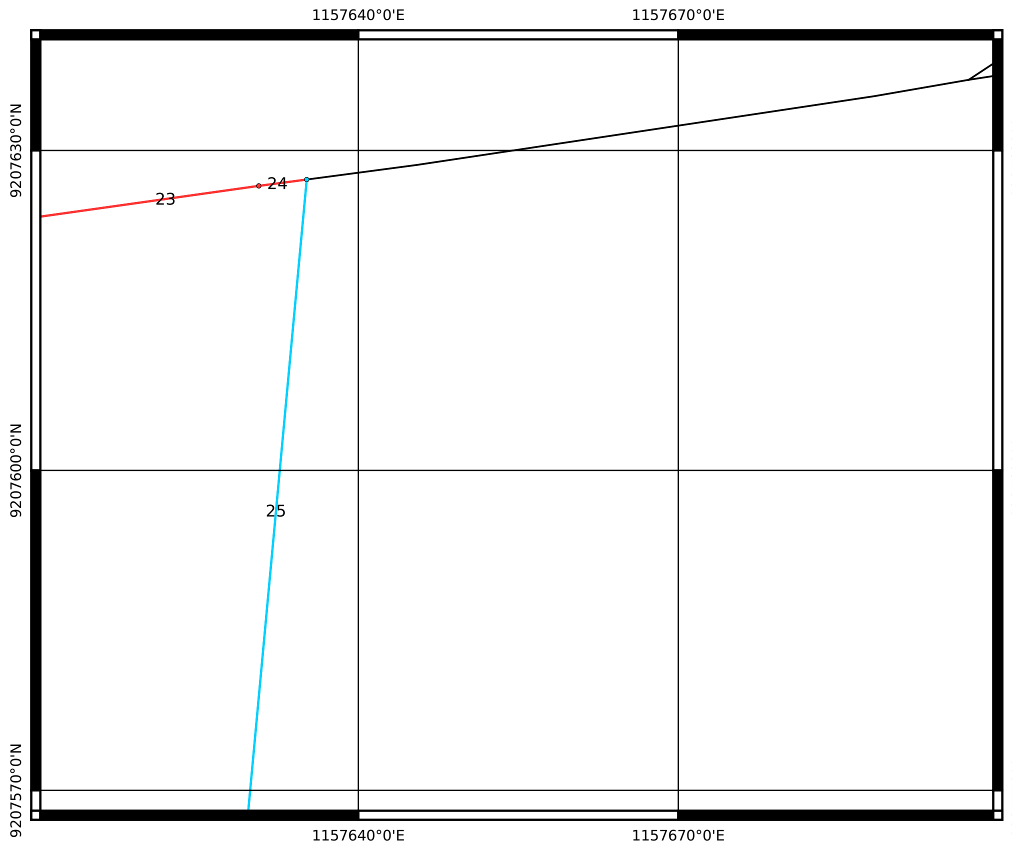
[30] Sanne Wøhlk. *A Decade of Capacitated Arc Routing*, volume 43 of *Operations Research/Computer Science Interfaces*, book section 2, pages 29–48. Springer US, 2008.

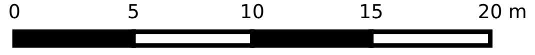
[31] Kriangchai Yaoyuenyong, Peerayuth Charnsethikul, and Vira Chankong. A heuristic algorithm for the mixed chinese postman problem. *Optimization and Engineering*, 3(2):157–187, 2002.

The appendix of the report has not been included here. To access it contact the author’s.

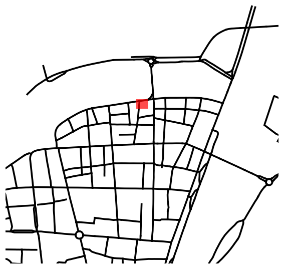
DETAILED MAPS FOR THE GENERATED ROUTES

G.1 Detailed Generated Map for Route 310174_KV_B



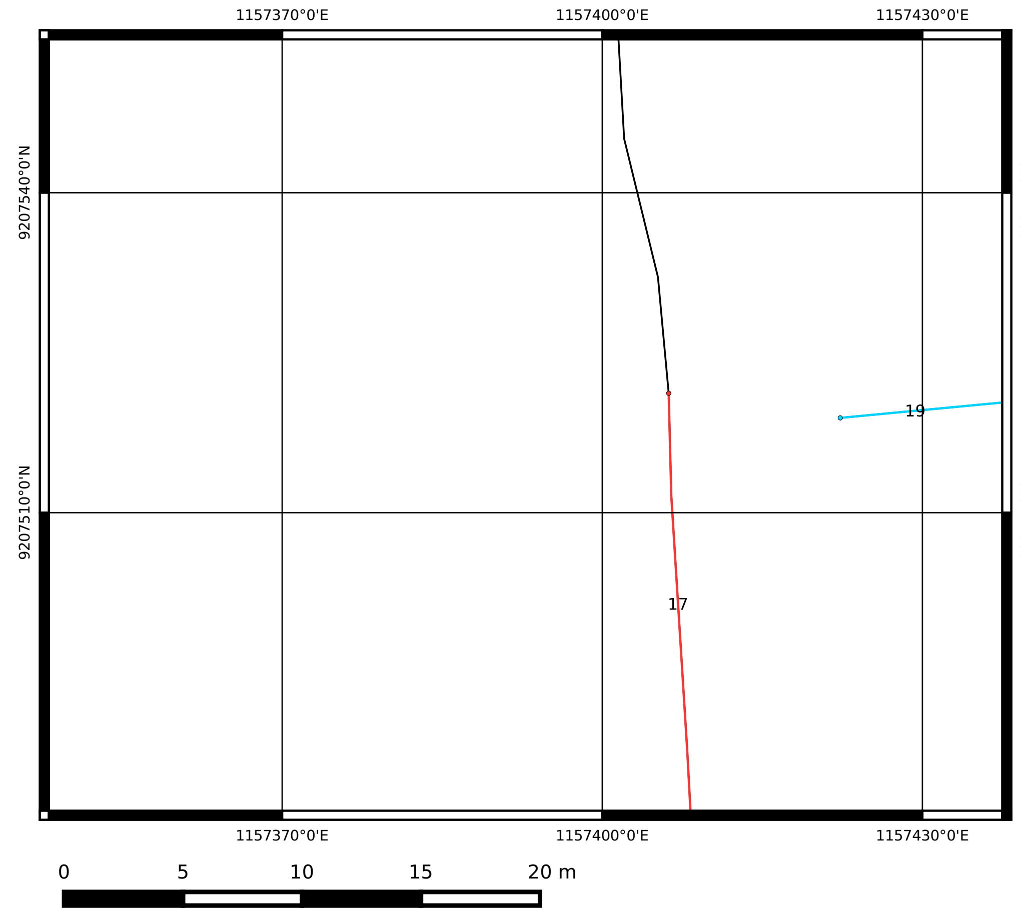


Details for the generated route 310174_KV_B_Midtbyen

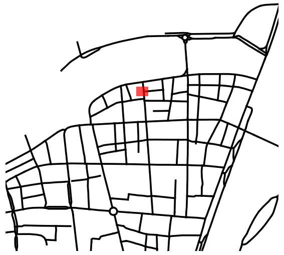


Legend

🟥	Not Required
🟦	Required
—	Not used road

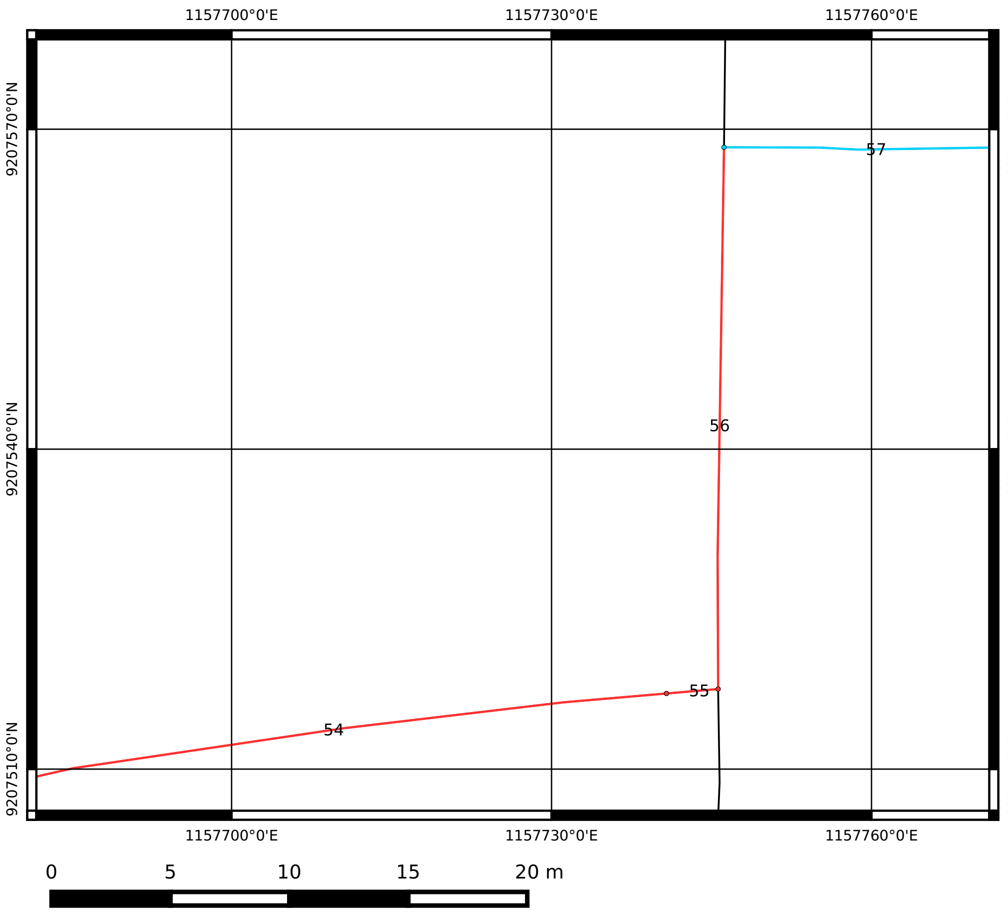


Details for the generated route 310174_KV_B_Midtbyen

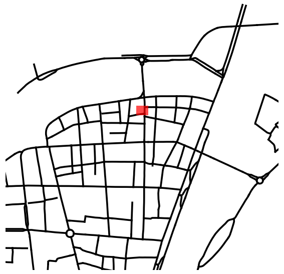


Legend

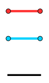
🟥	Not Required
🟦	Required
—	Not used road

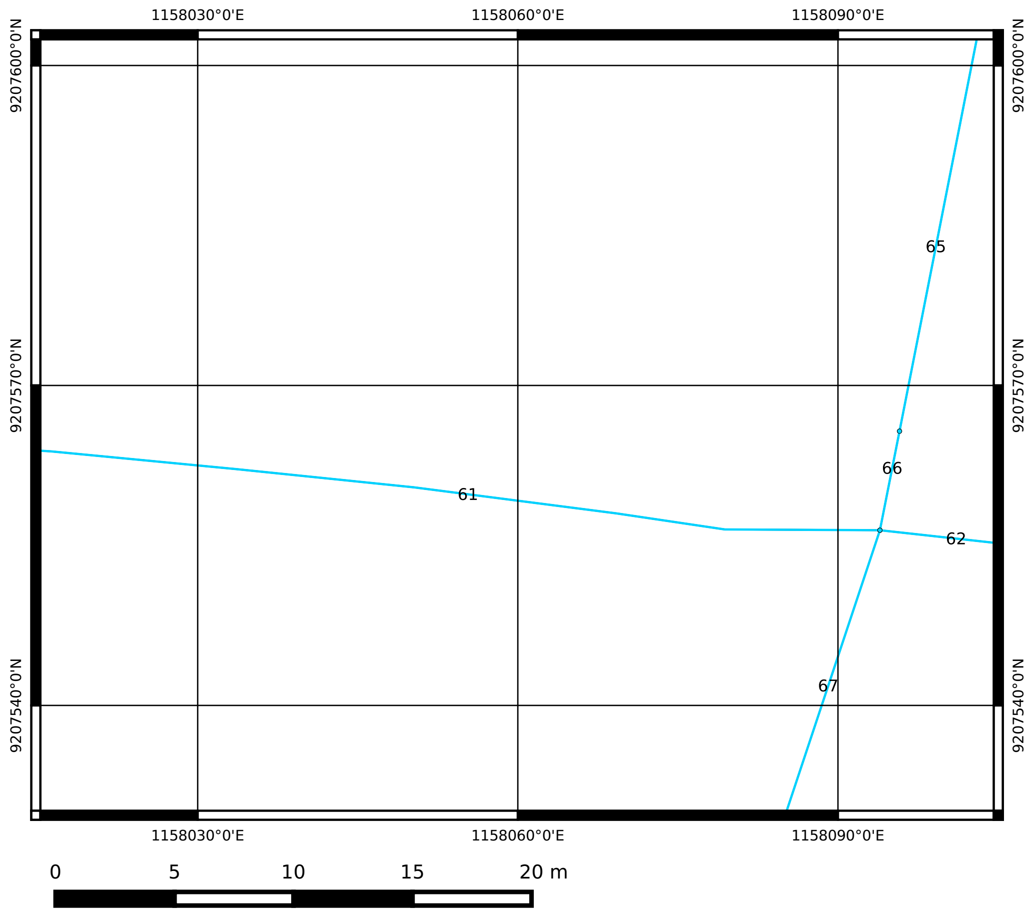


Details for the generated route 310174_KV_B_Midtbyen

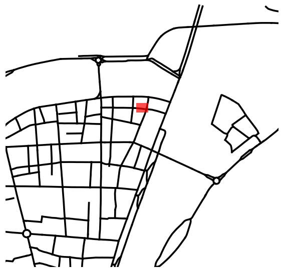


Legend



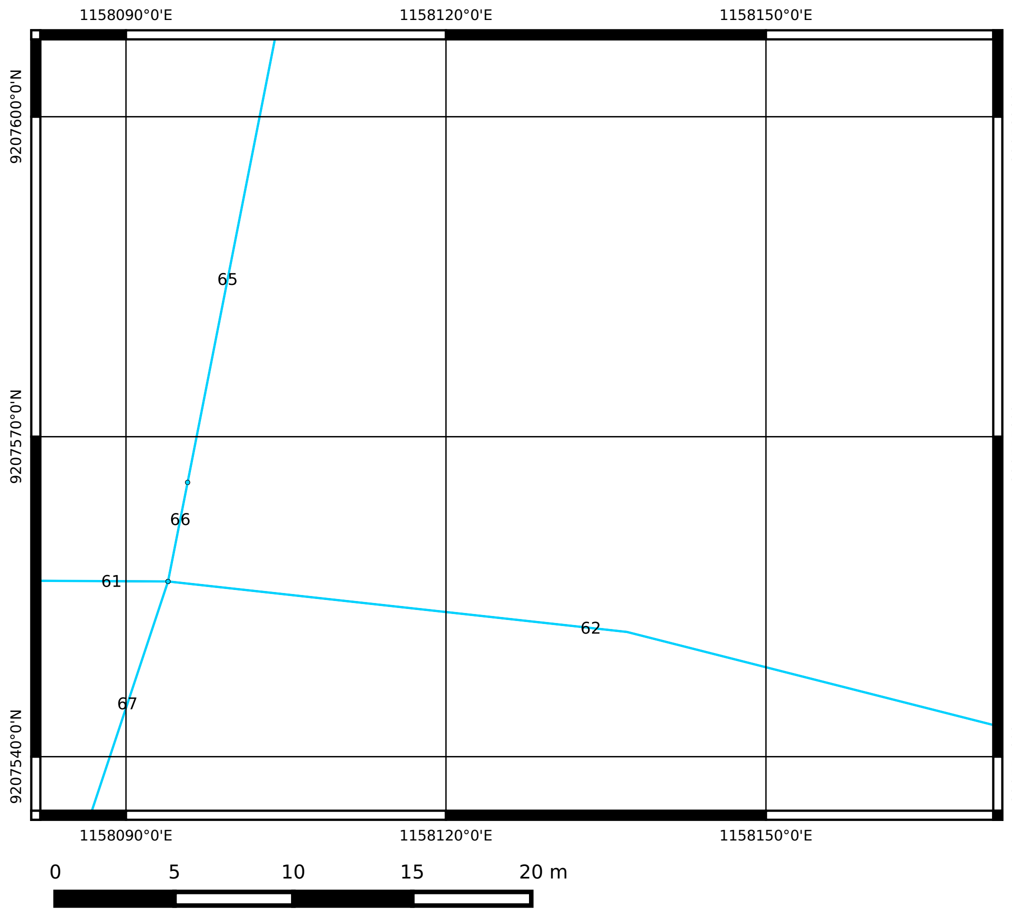


Details for the generated route 310174_KV_B_Midtbyen

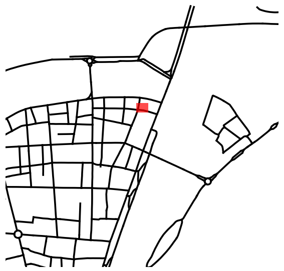


Legend

- 🟥 Not Required
- 🟦 Required
- — Not used road

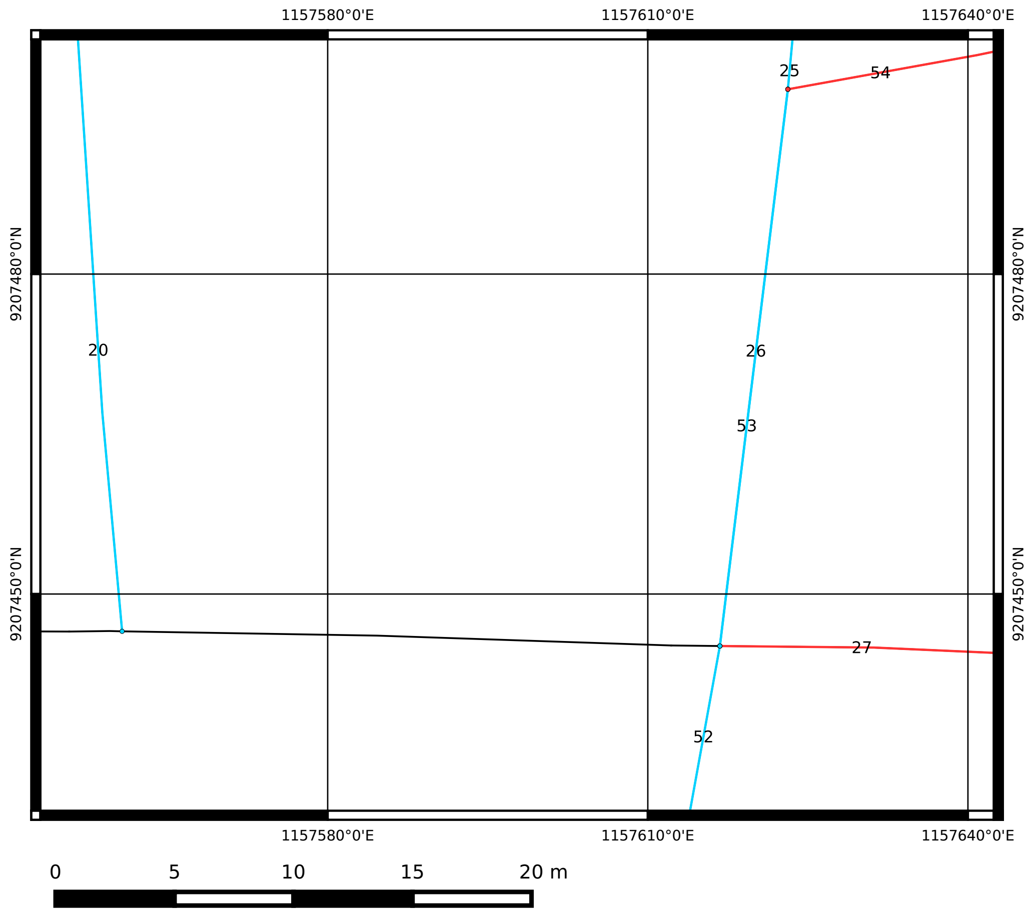


Details for the generated route 310174_KV_B_Midtbyen

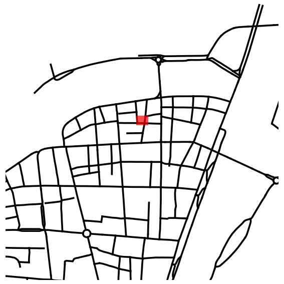


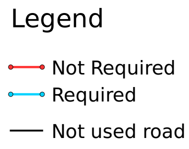
Legend

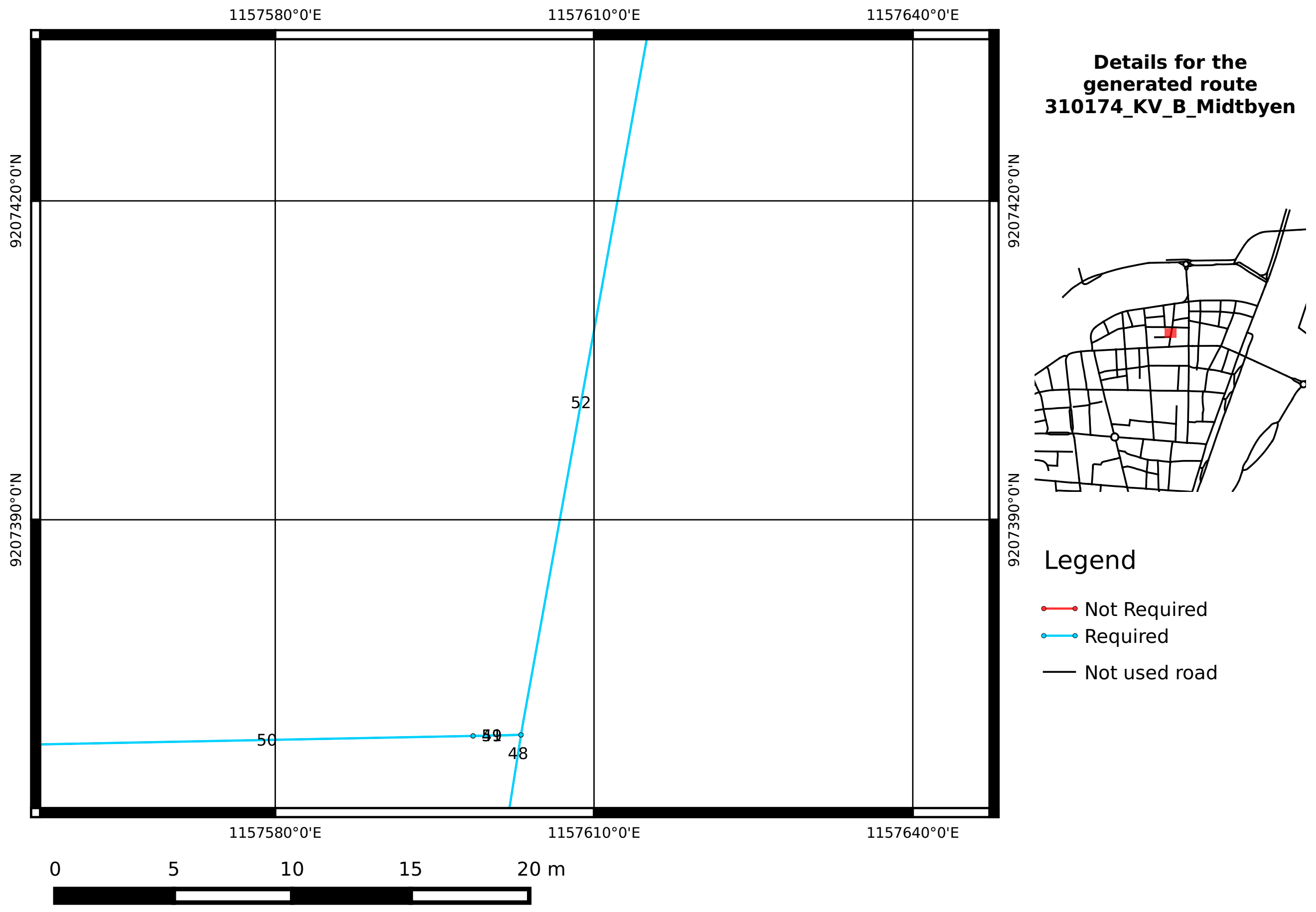
🟥	Not Required
🟦	Required
—	Not used road



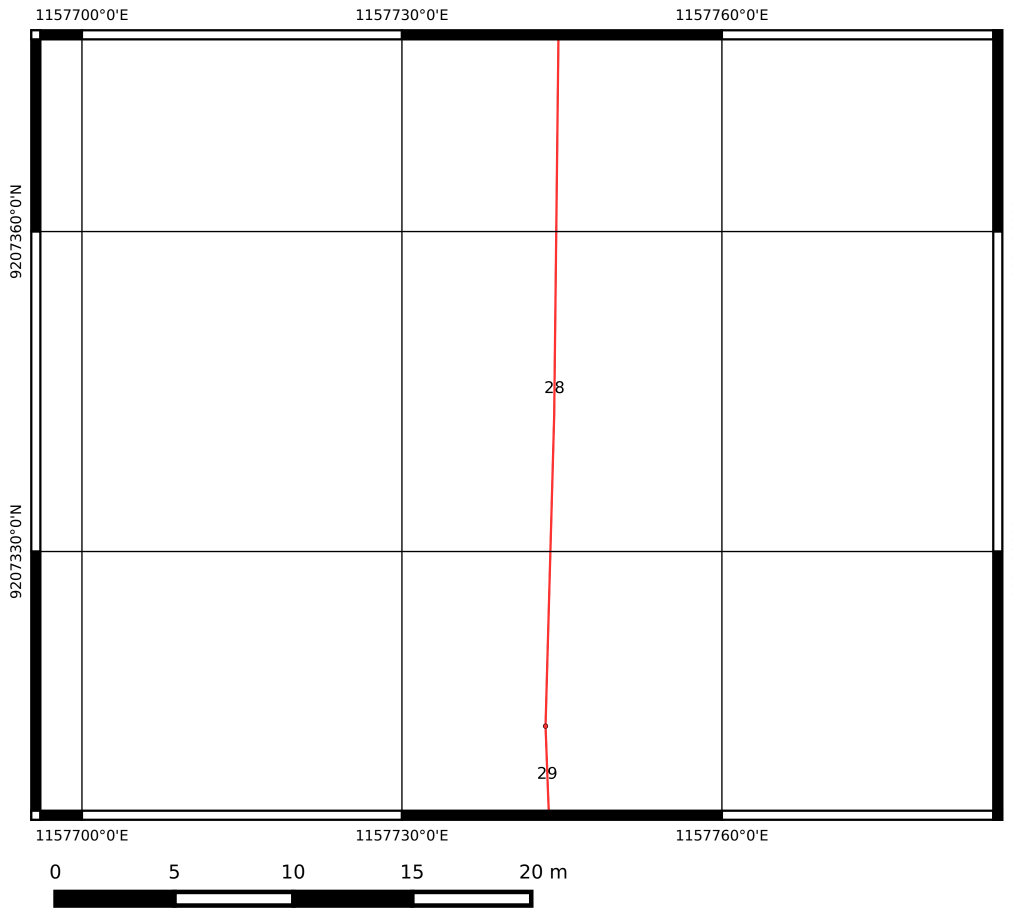
Details for the generated route 310174_KV_B_Midtbyen



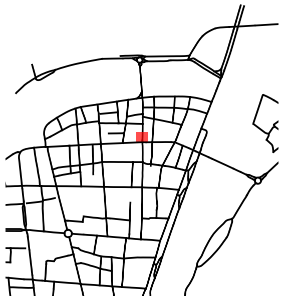




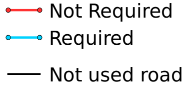
1157580°0'E
1157610°0'E
1157640°0'E
9207420°0'N
9207390°0'N
52
50
49
51
48
0
5
10
15
20 m
Details for the generated route 310174_KV_B_Midtbyen
Legend
Not Required
Required
Not used road

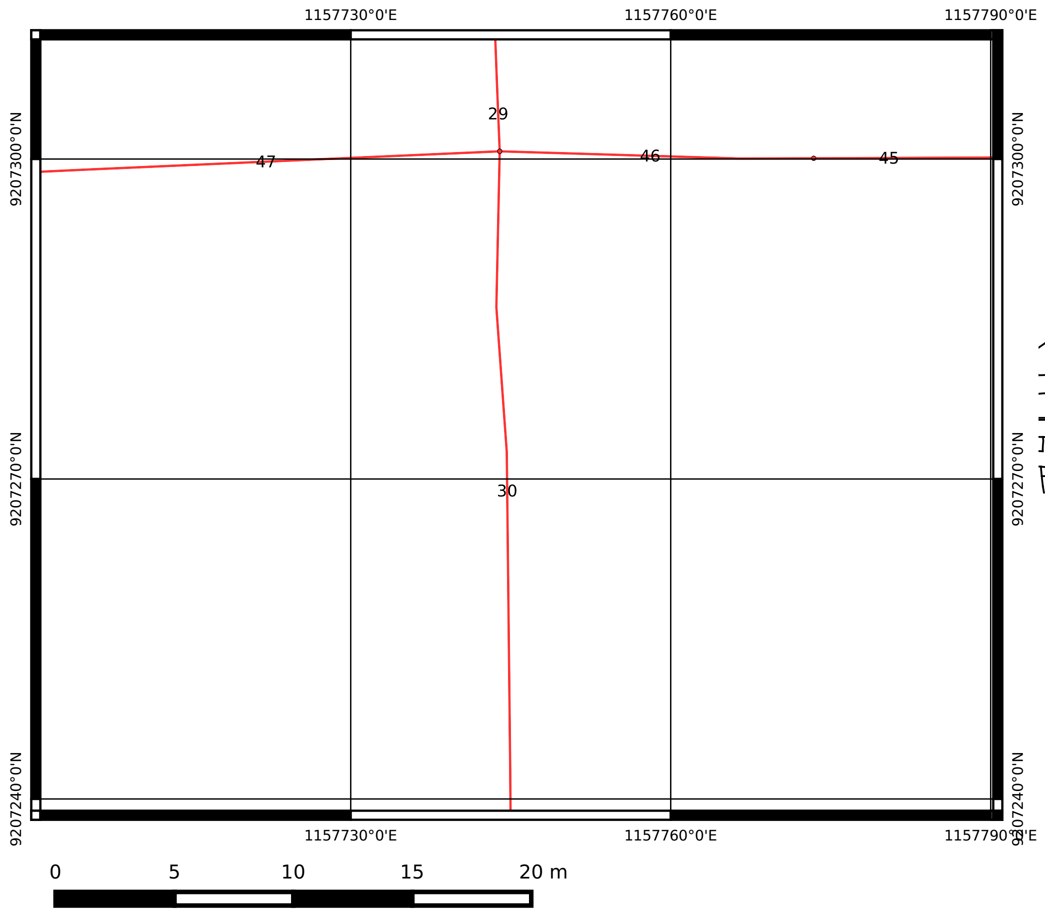


Details for the generated route 310174_KV_B_Midtbyen

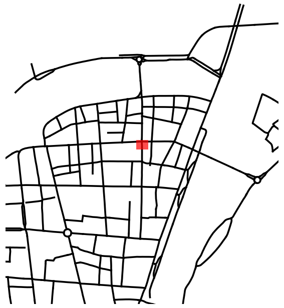


Legend



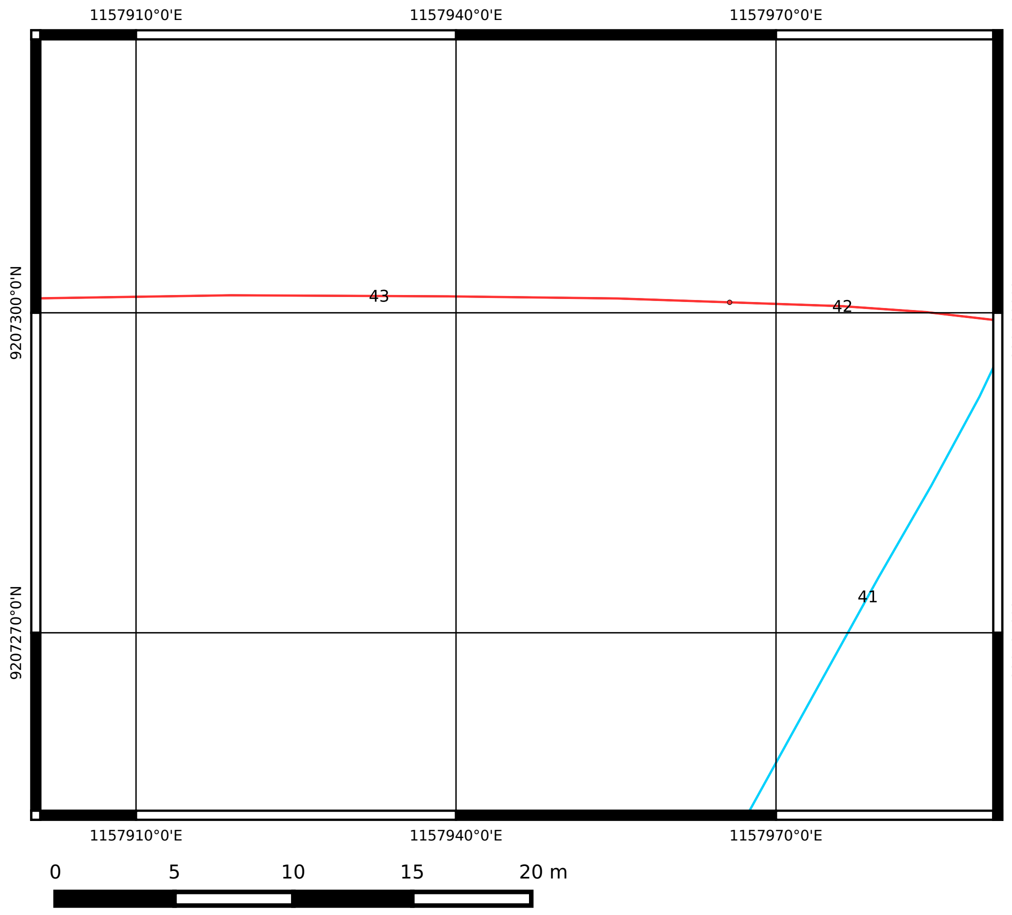


Details for the generated route 310174_KV_B_Midtbyen

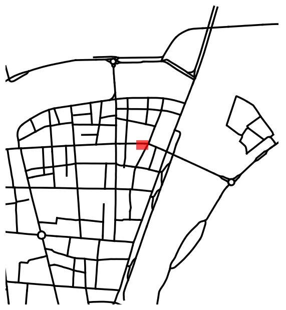


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

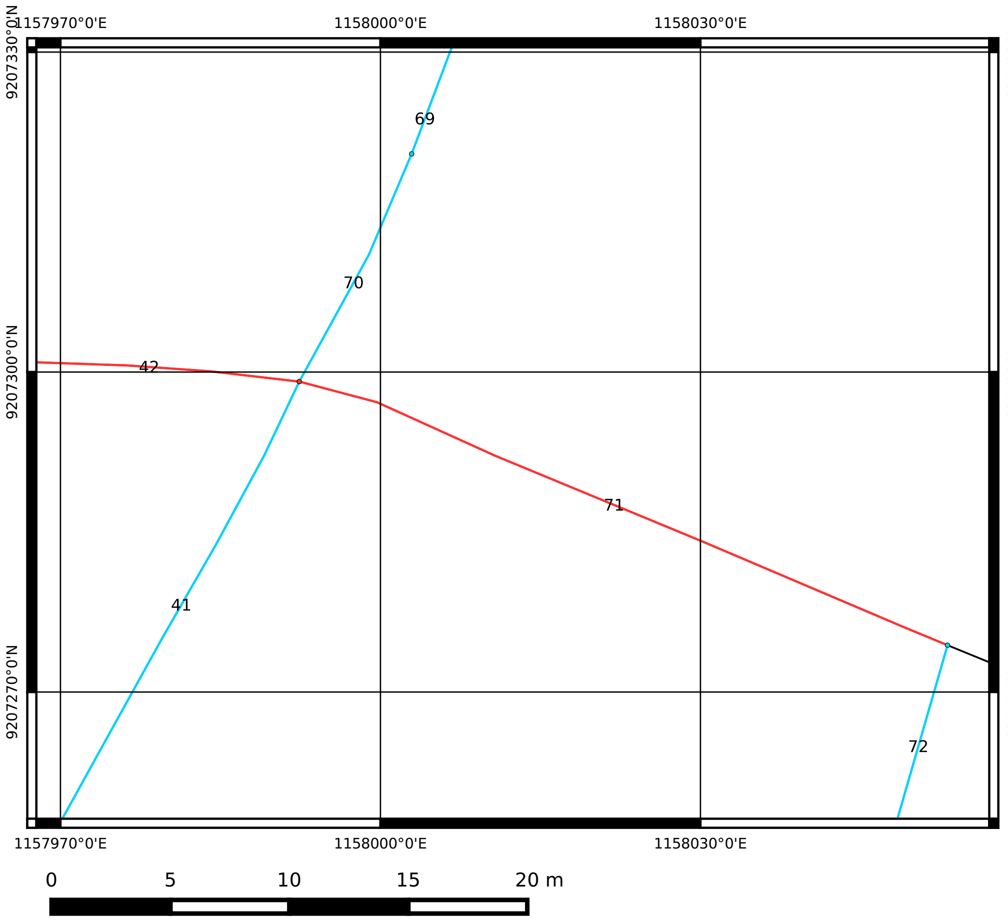


Legend

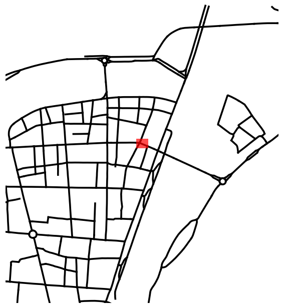
Not Required

Required

Not used road

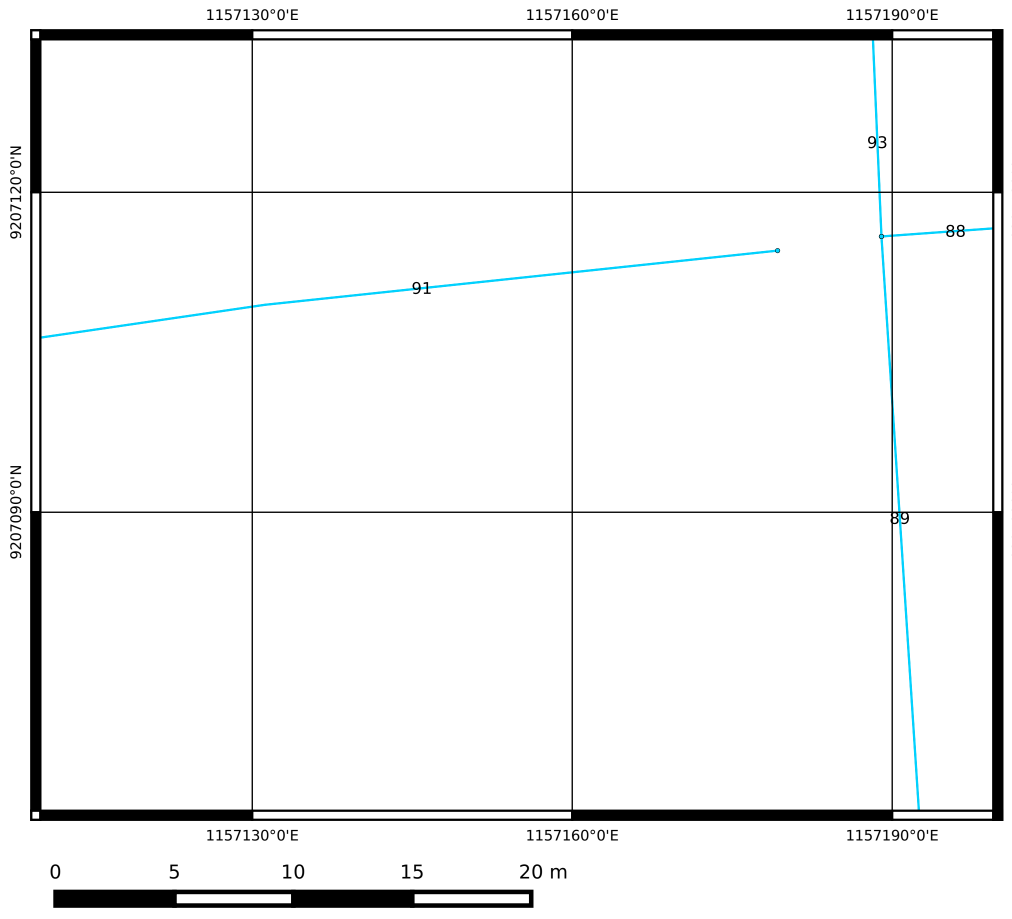


Details for the generated route 310174_KV_B_Midtbyen

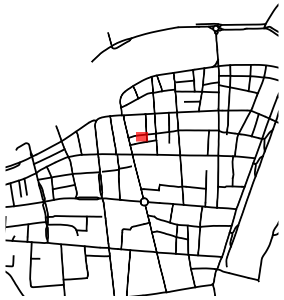


Legend

🟥	Not Required
🟦	Required
—	Not used road

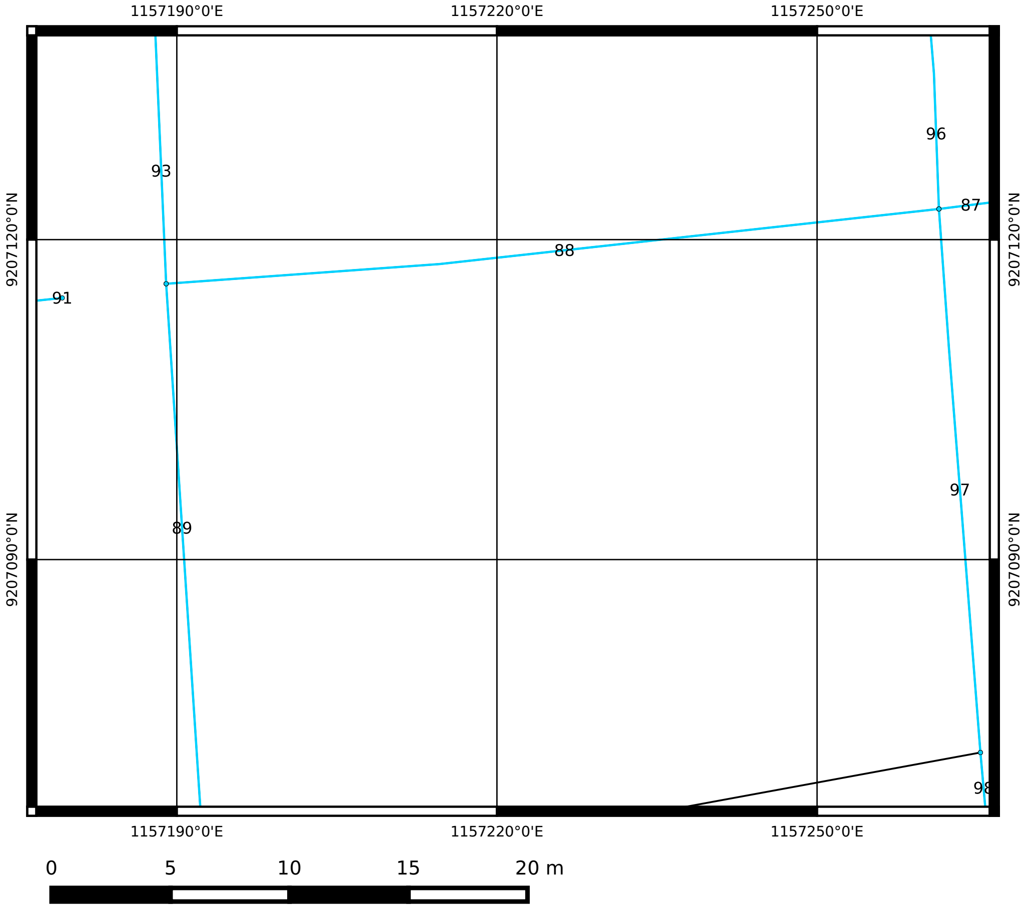


Details for the generated route 310174_KV_B_Midtbyen

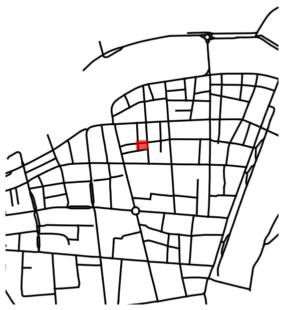


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

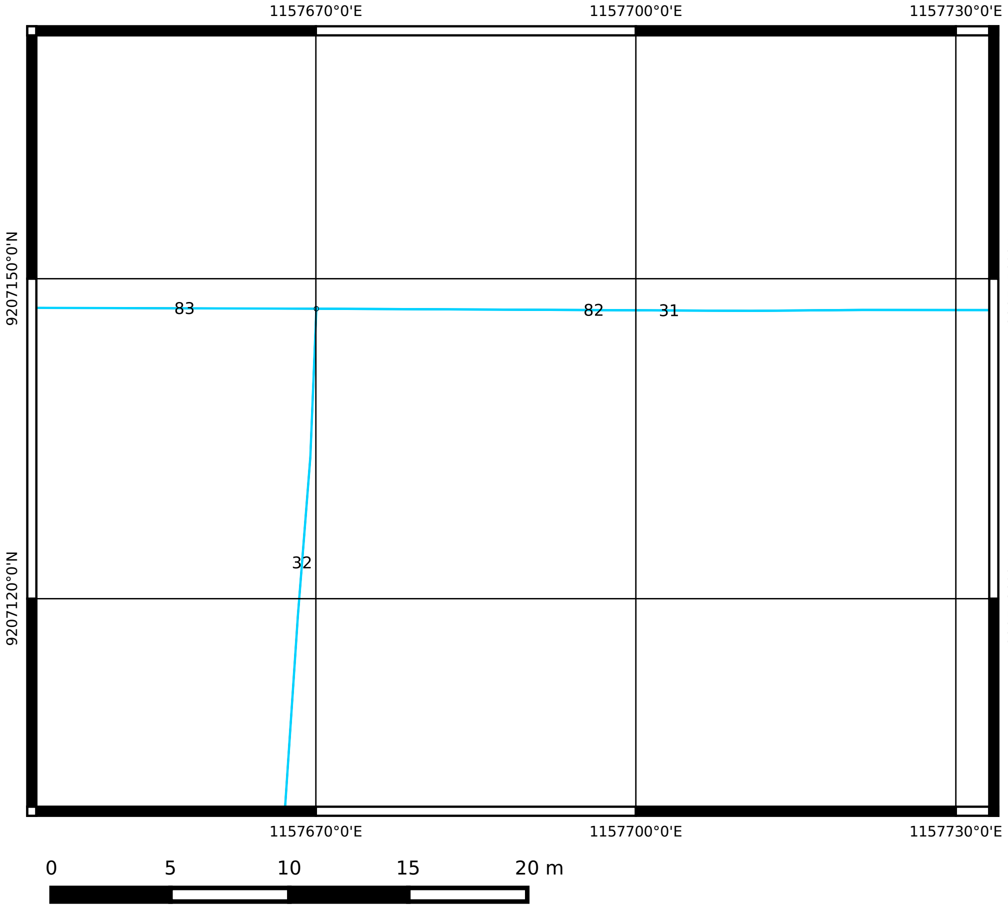


Legend

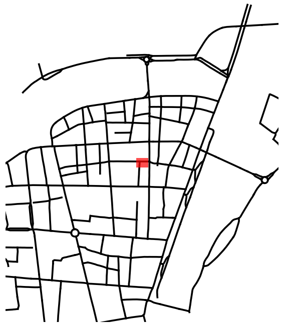
🟥 Not Required

🟦 Required

— Not used road

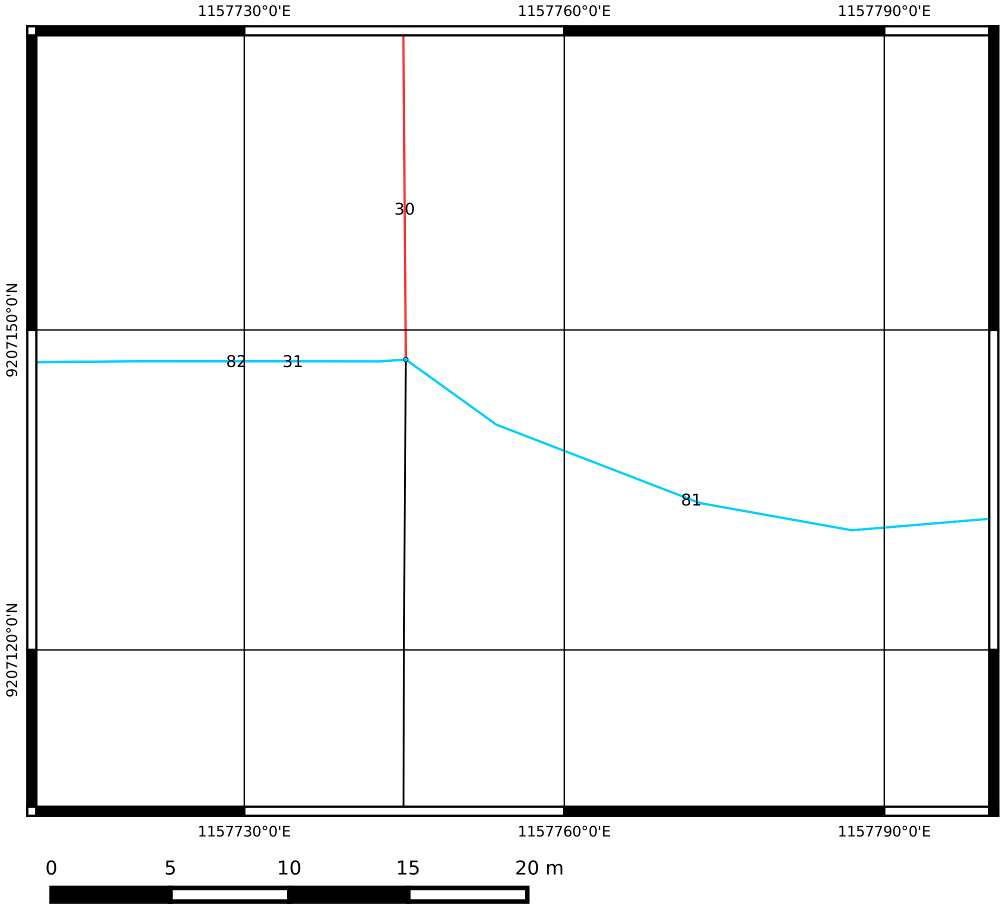


Details for the generated route 310174_KV_B_Midtbyen

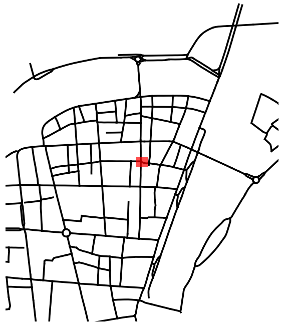


Legend

- Not Required
- Required
- Not used road



Details for the generated route 310174_KV_B_Midtbyen

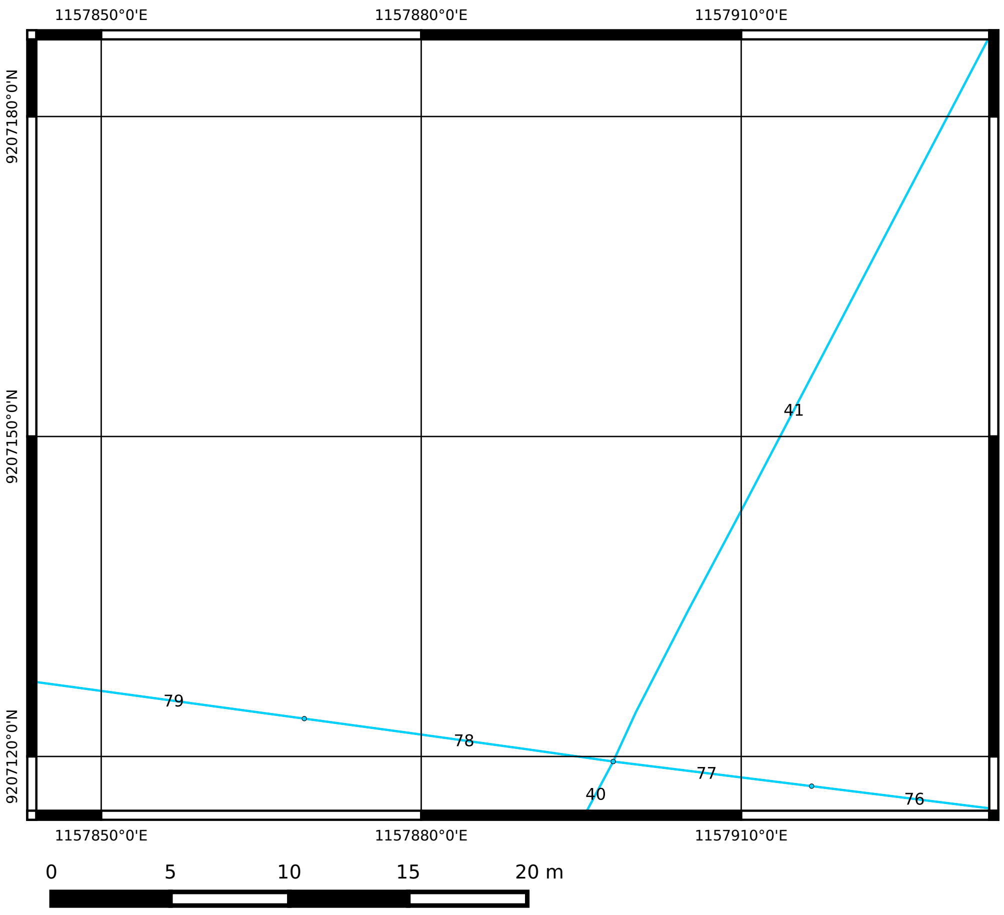


Legend

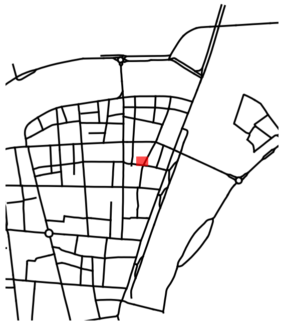
🟥 Not Required

🟦 Required

— Not used road

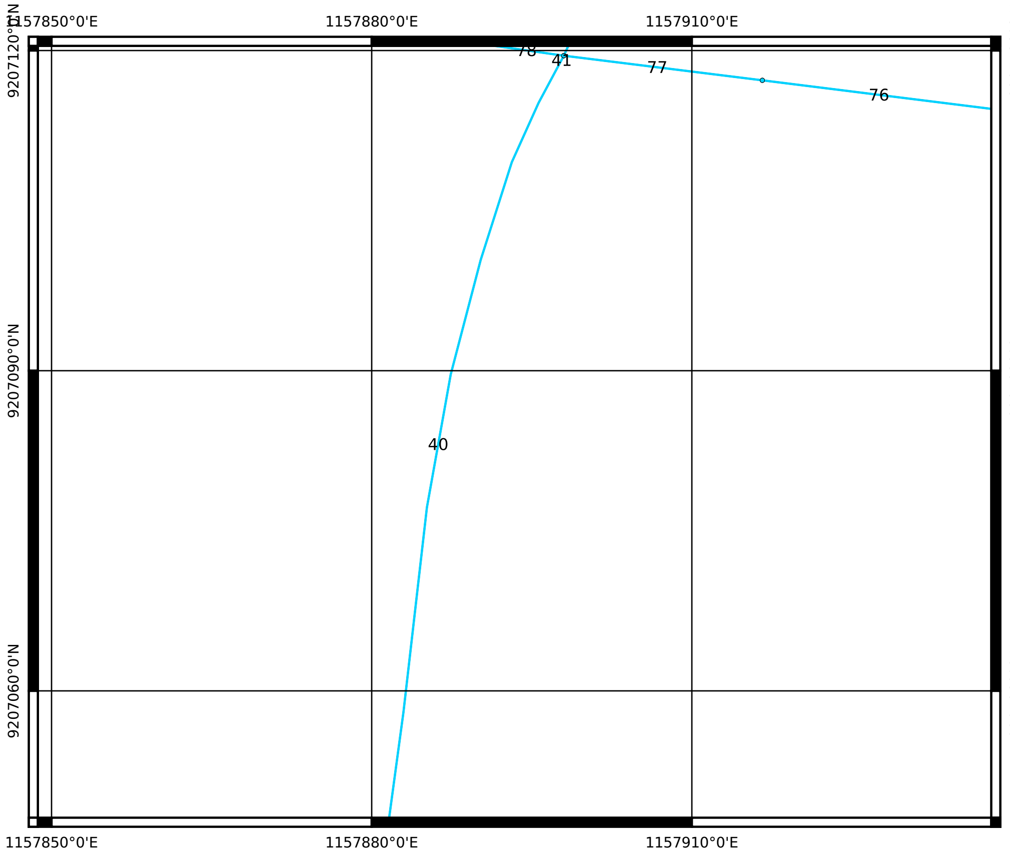


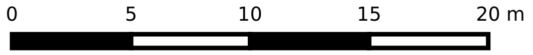
Details for the generated route 310174_KV_B_Midtbyen



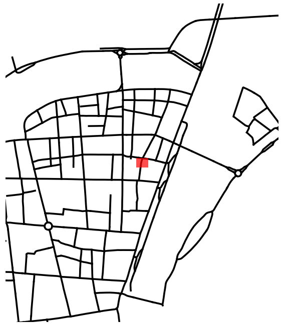
Legend

🟥	Not Required
🟦	Required
—	Not used road



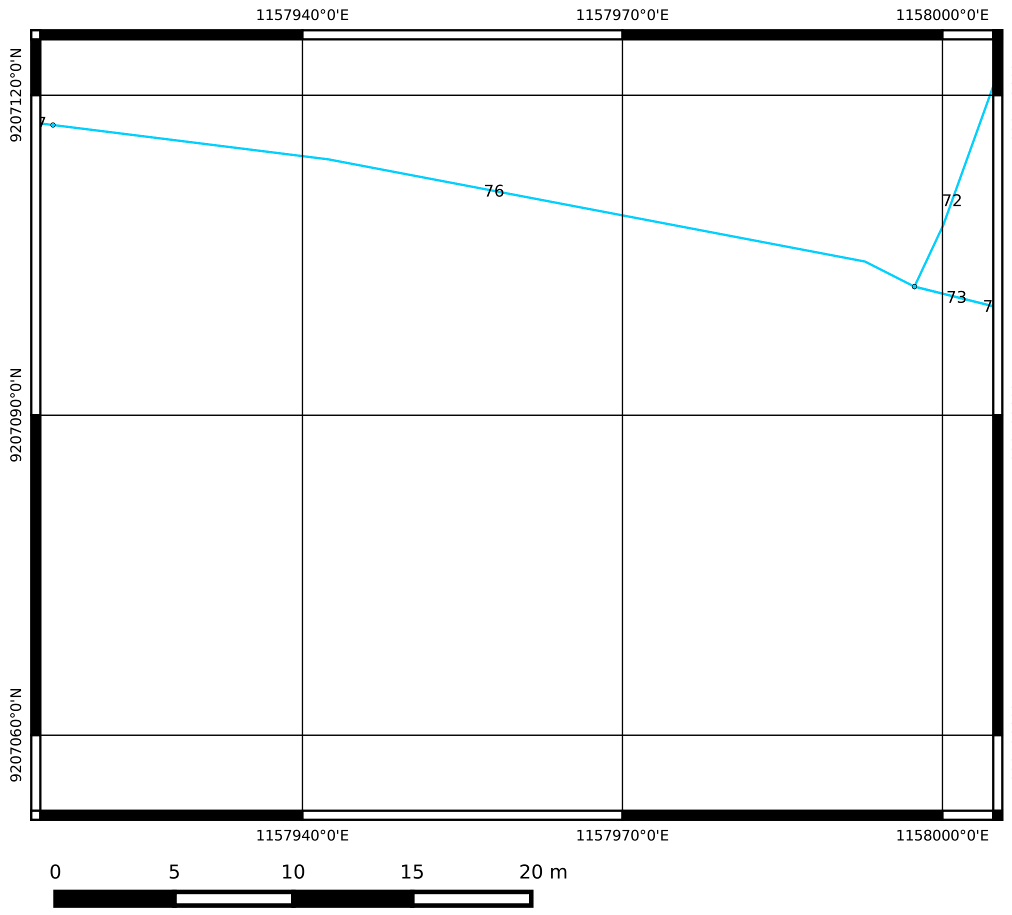


Details for the generated route 310174_KV_B_Midtbyen

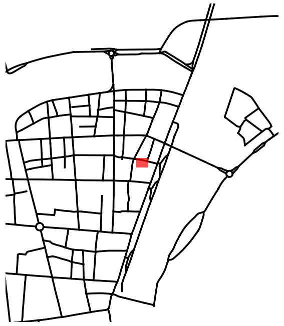


Legend

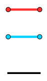
- Not Required
- Required
- Not used road

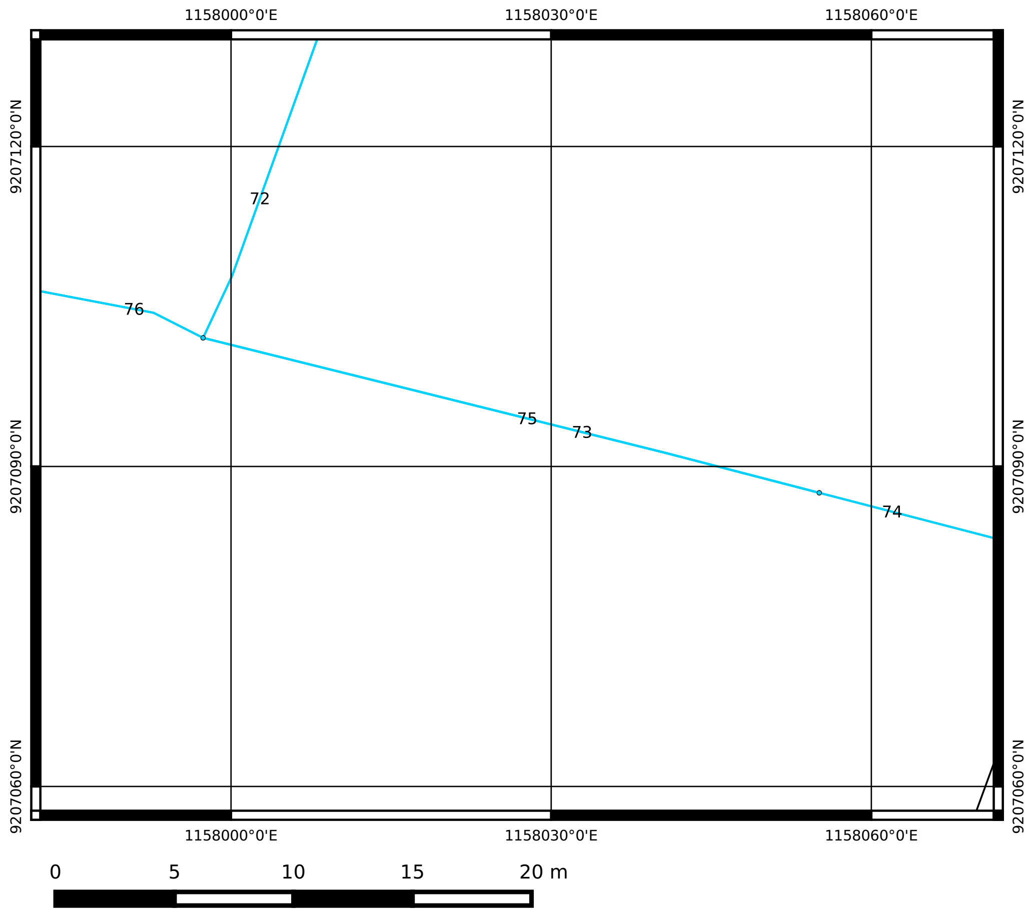


Details for the generated route 310174_KV_B_Midtbyen

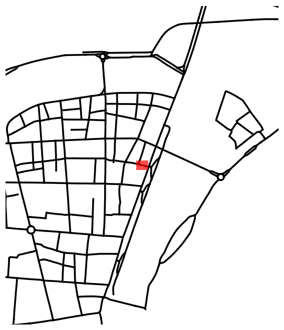


Legend



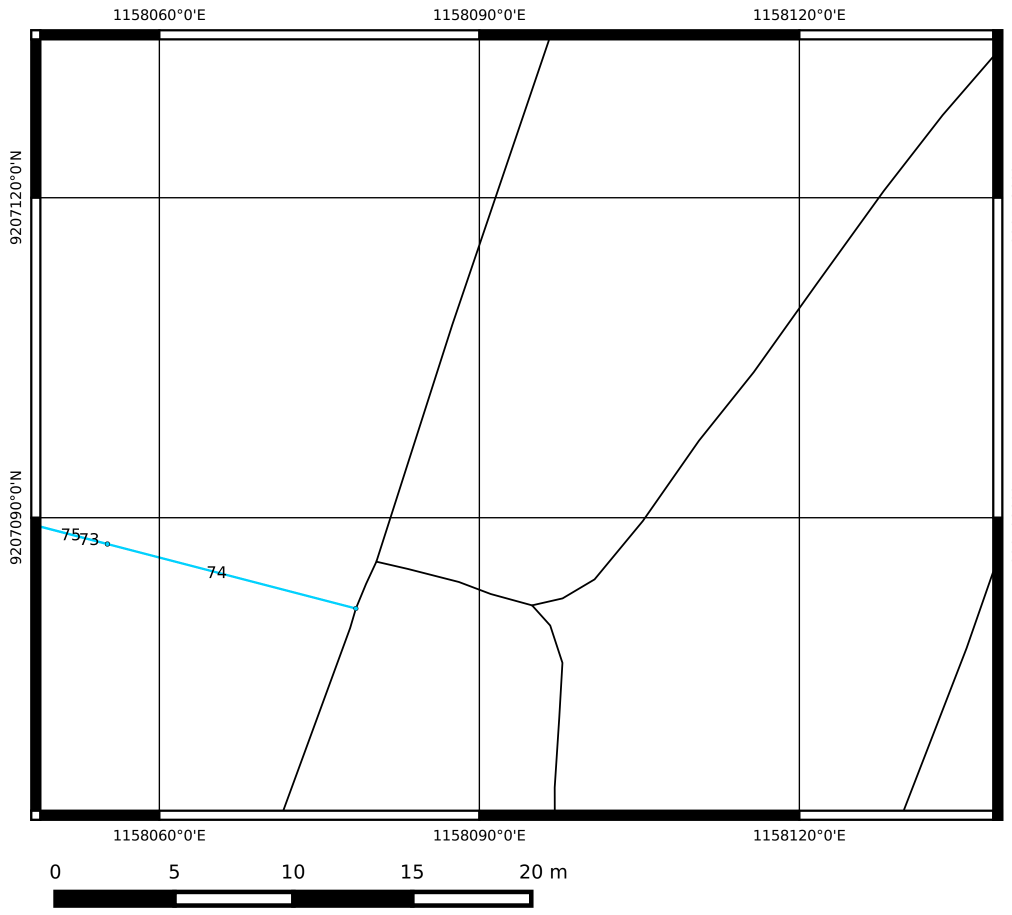


Details for the generated route 310174_KV_B_Midtbyen

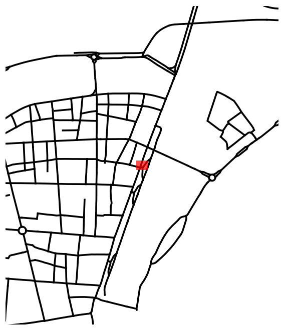


Legend

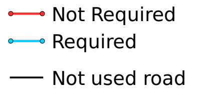
- 🟥 Not Required
- 🟦 Required
- — Not used road

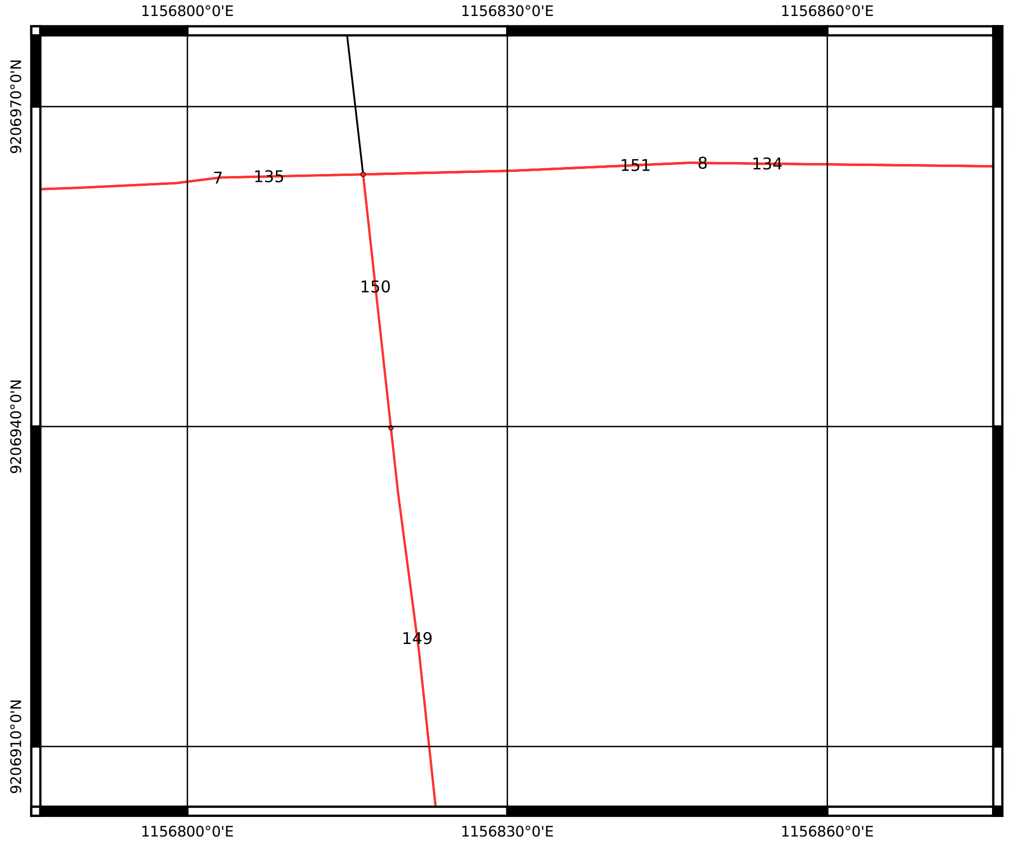


Details for the generated route 310174_KV_B_Midtbyen

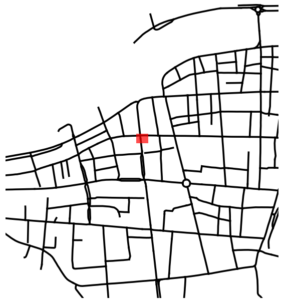


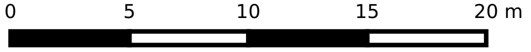
Legend





Details for the generated route 310174_KV_B_Midtbyen



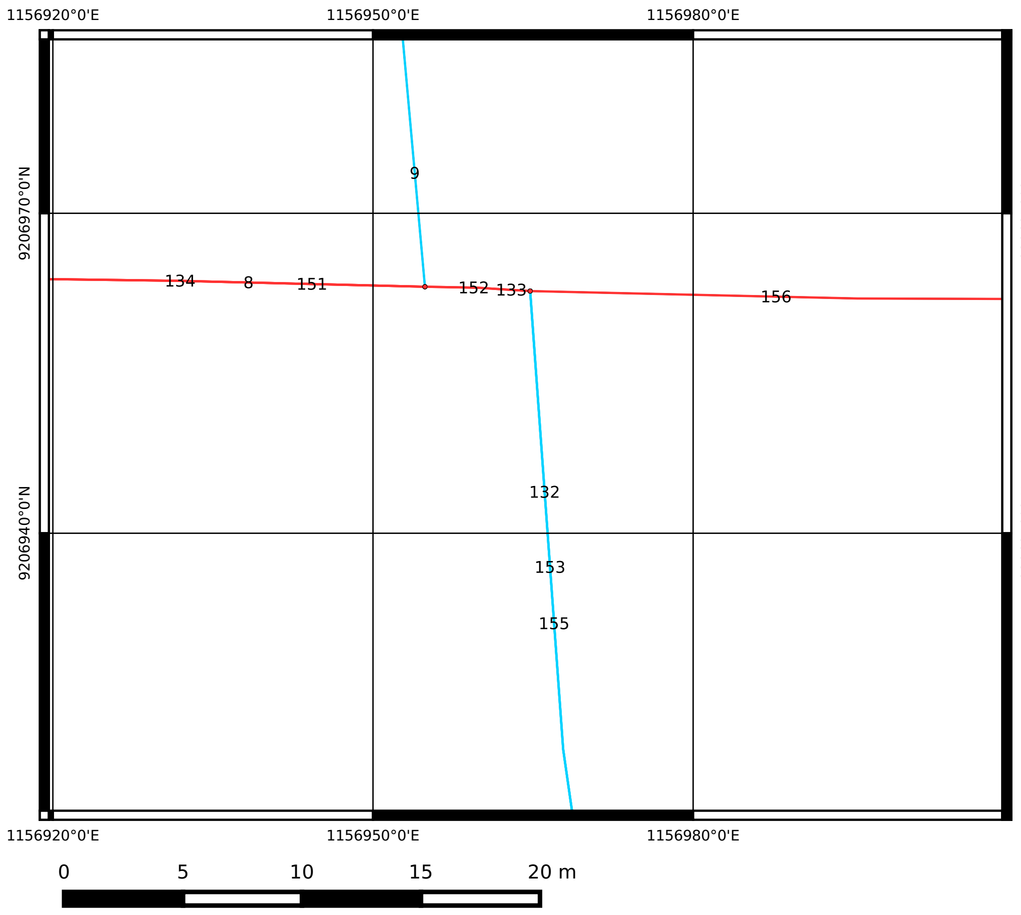


Legend

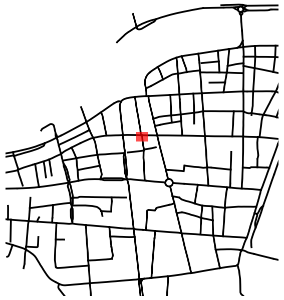
🟥 Not Required

🟦 Required

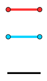
— Not used road



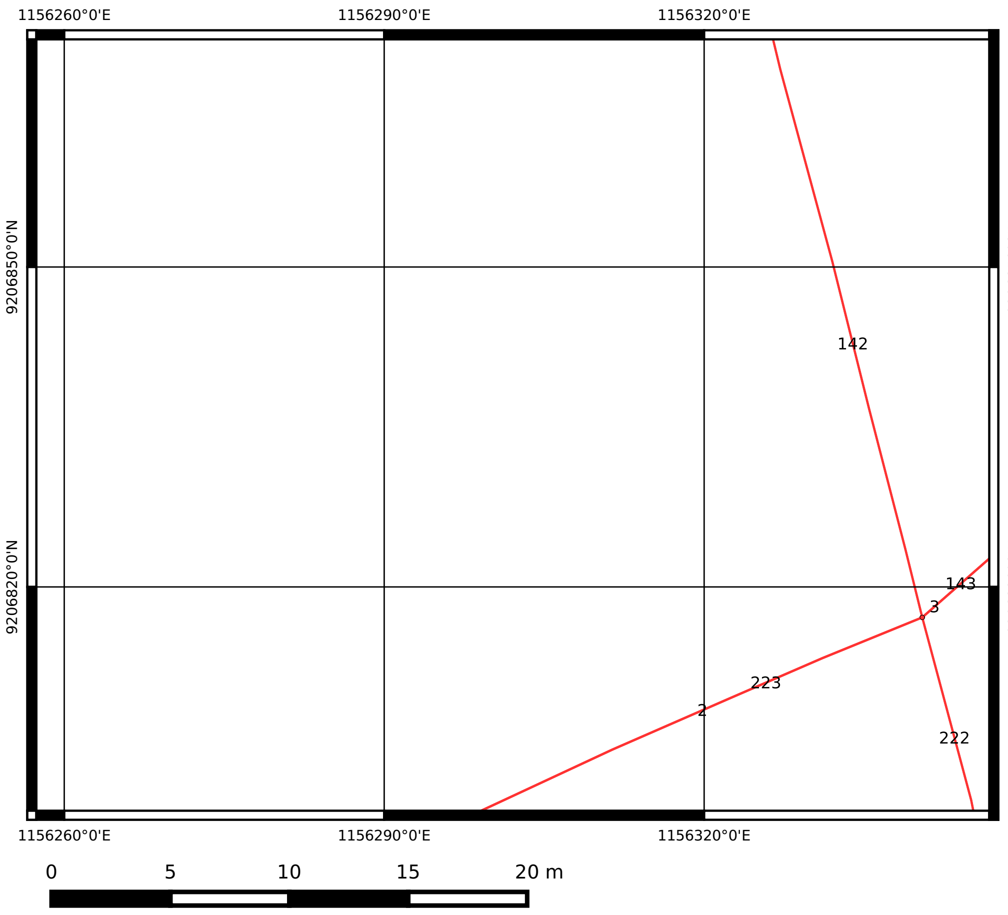
Details for the generated route 310174_KV_B_Midtbyen



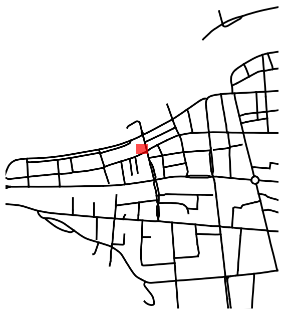
Legend



🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

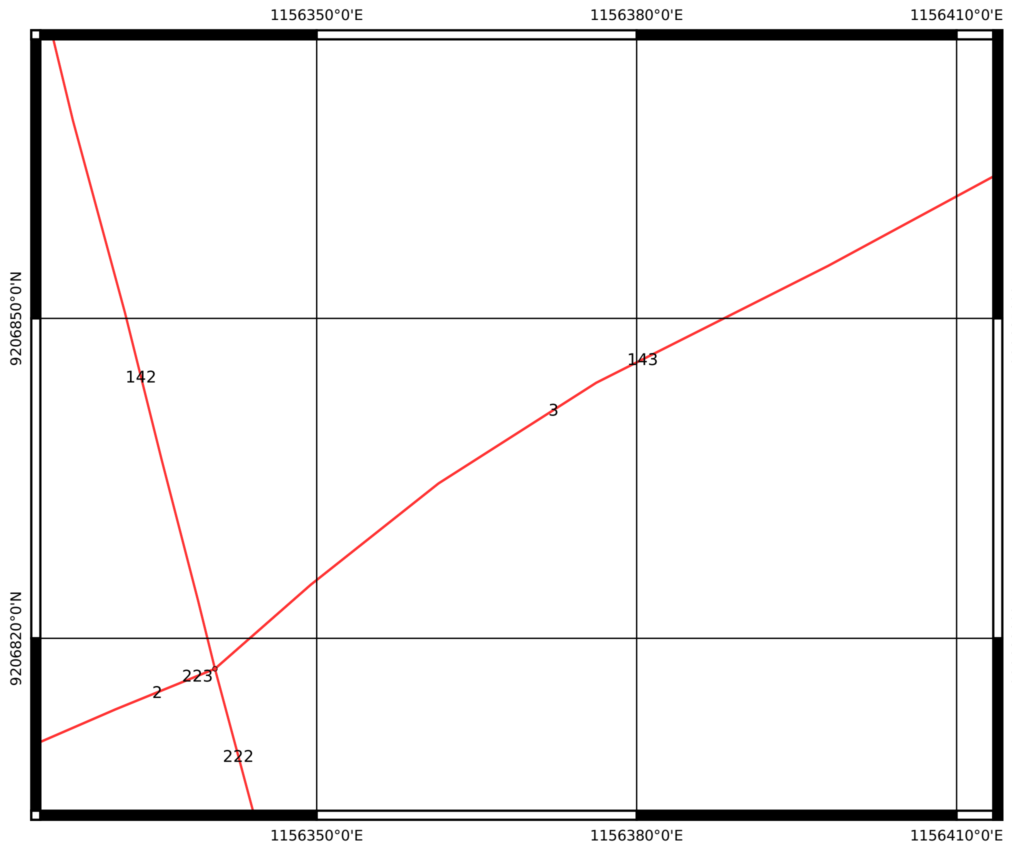


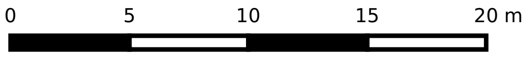
Legend

🟥 Not Required

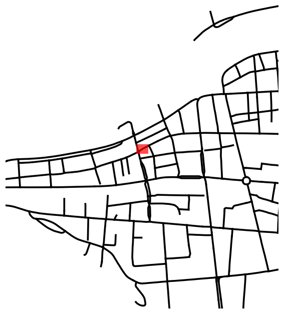
🟦 Required

— Not used road



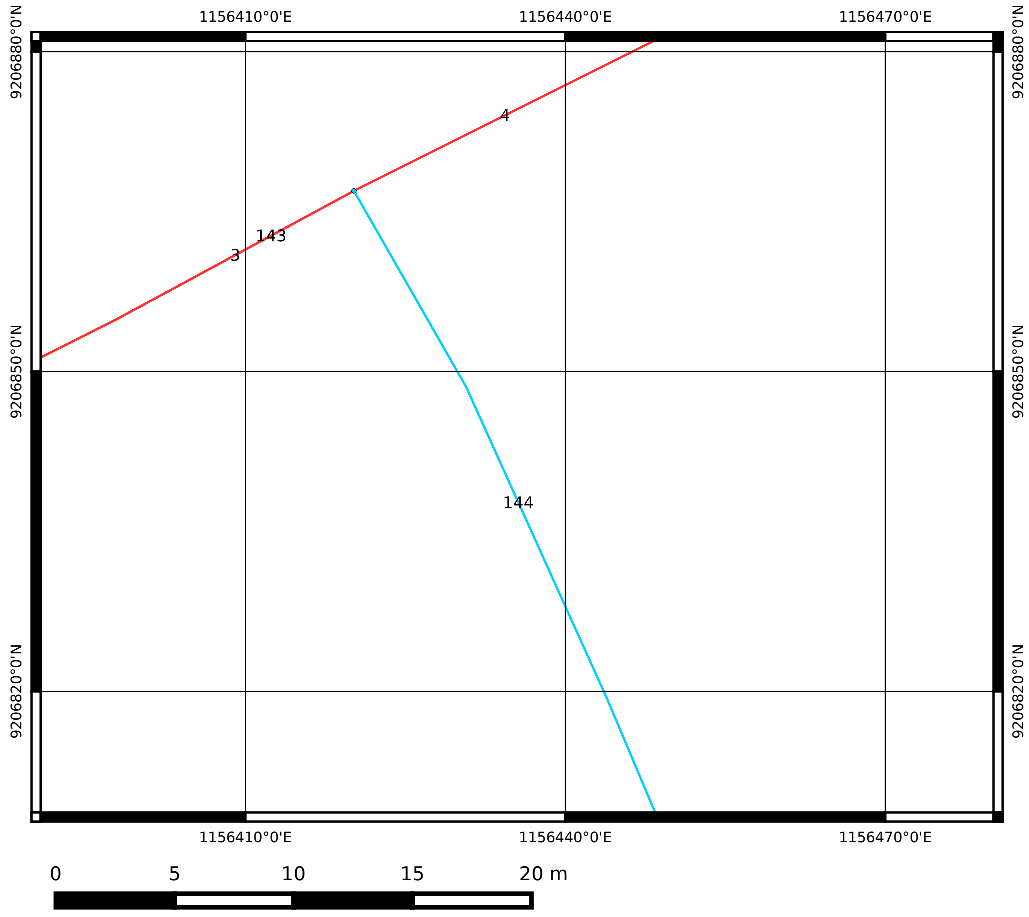


Details for the generated route 310174_KV_B_Midtbyen

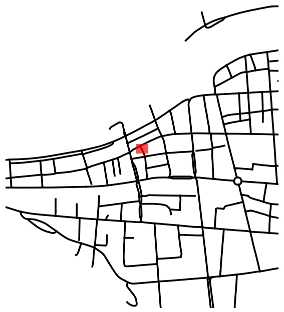


Legend

🟥	Not Required
🟦	Required
―	Not used road

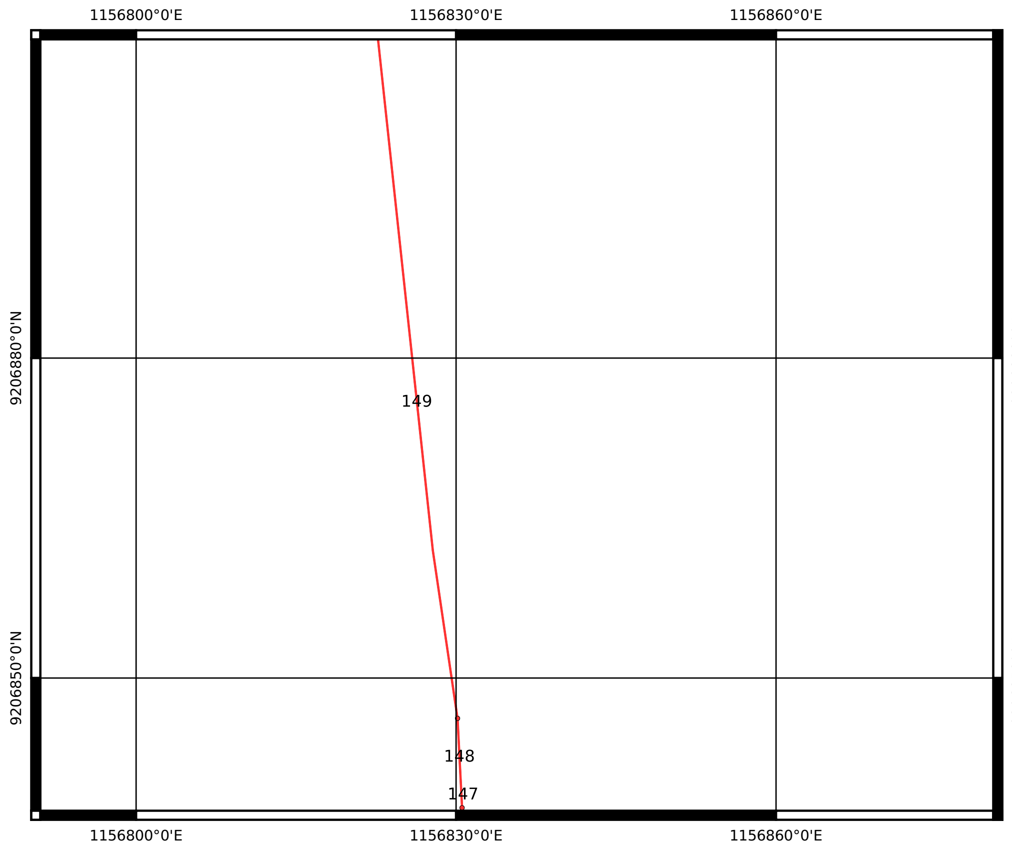


Details for the generated route 310174_KV_B_Midtbyen



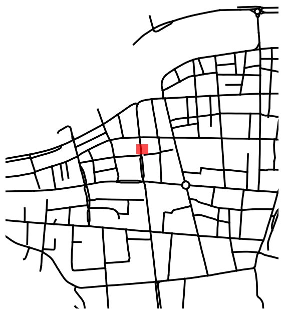
Legend

🟥	Not Required
🟦	Required
—	Not used road



0 5 10 15 20 m

Details for the generated route 310174_KV_B_Midtbyen

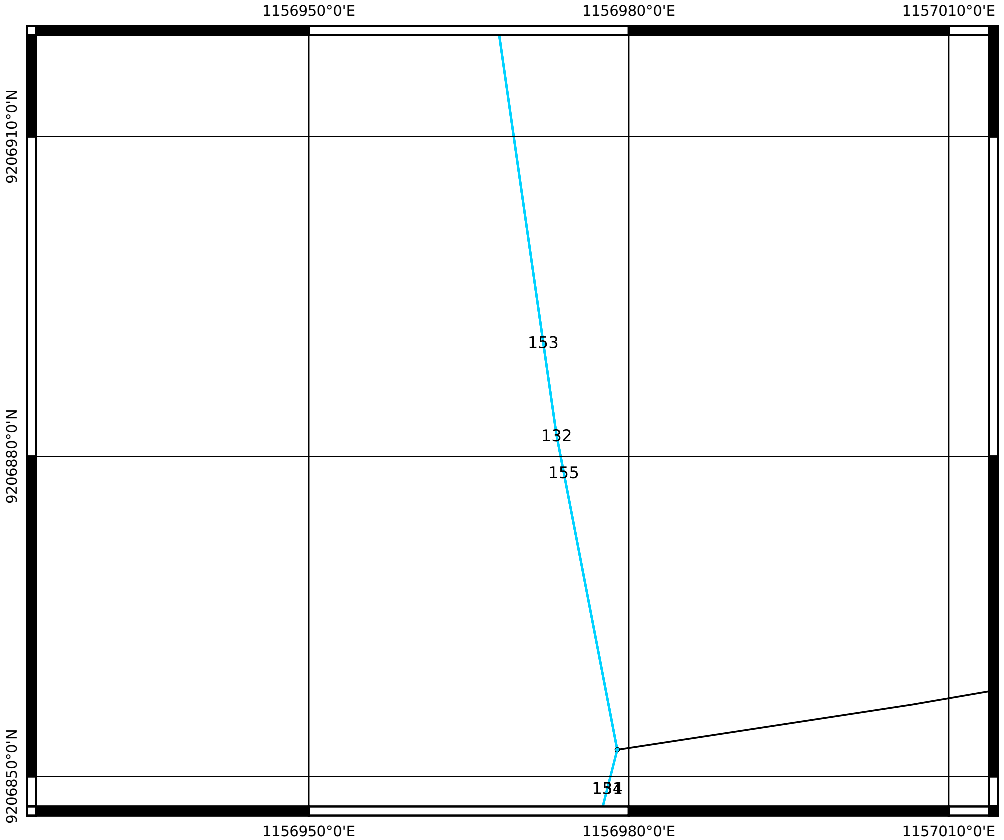


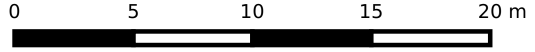
Legend

🟥 Not Required

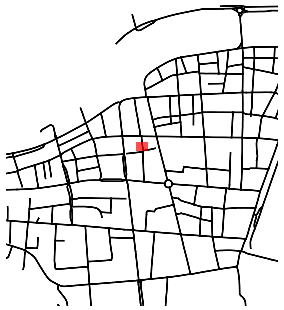
🟦 Required

— Not used road



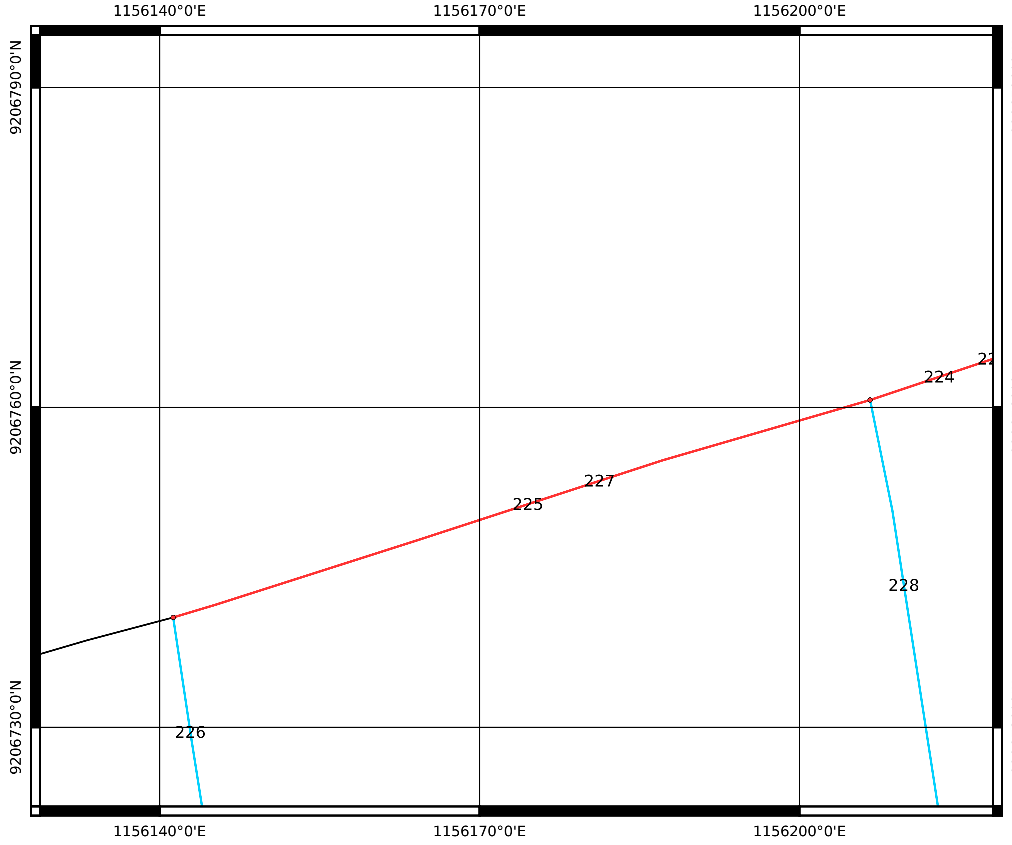


Details for the generated route 310174_KV_B_Midtbyen

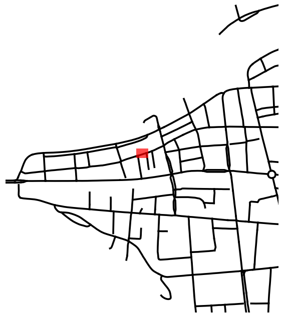


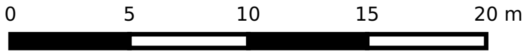
Legend

🟥	Not Required
🟦	Required
—	Not used road



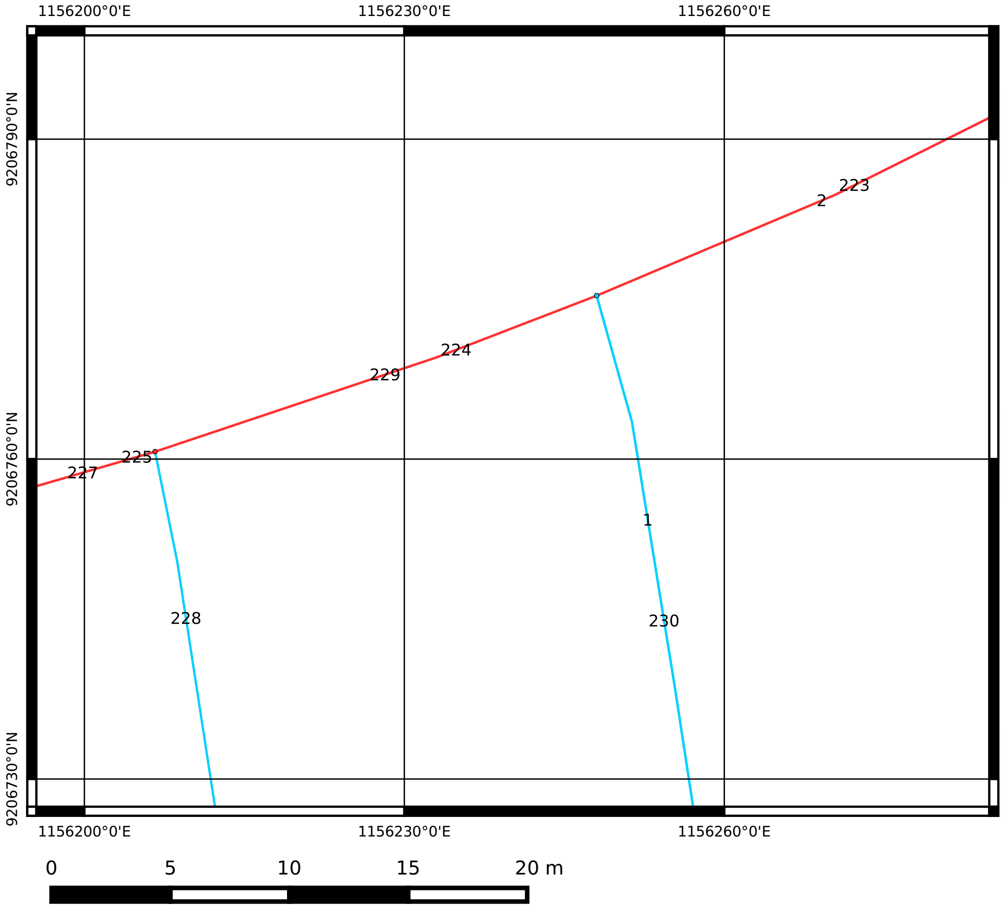
Details for the generated route 310174_KV_B_Midtbyen



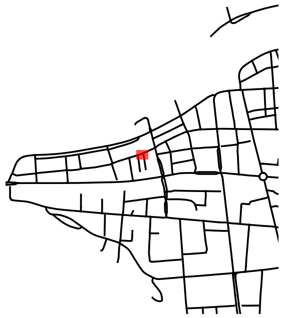


Legend

🟥	Not Required
🟦	Required
—	Not used road

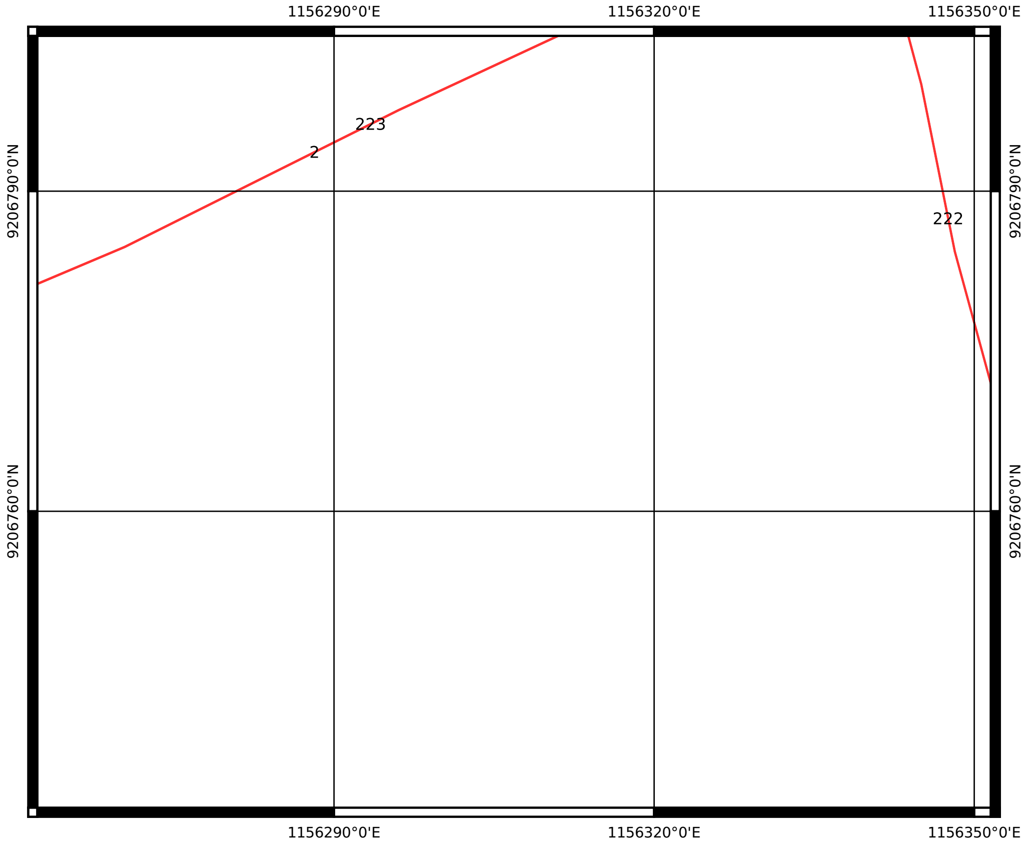


Details for the generated route 310174_KV_B_Midtbyen

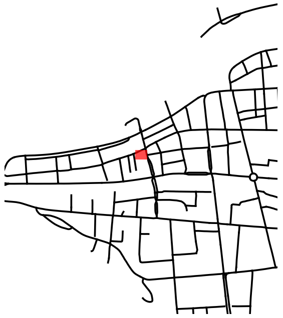


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

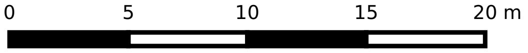


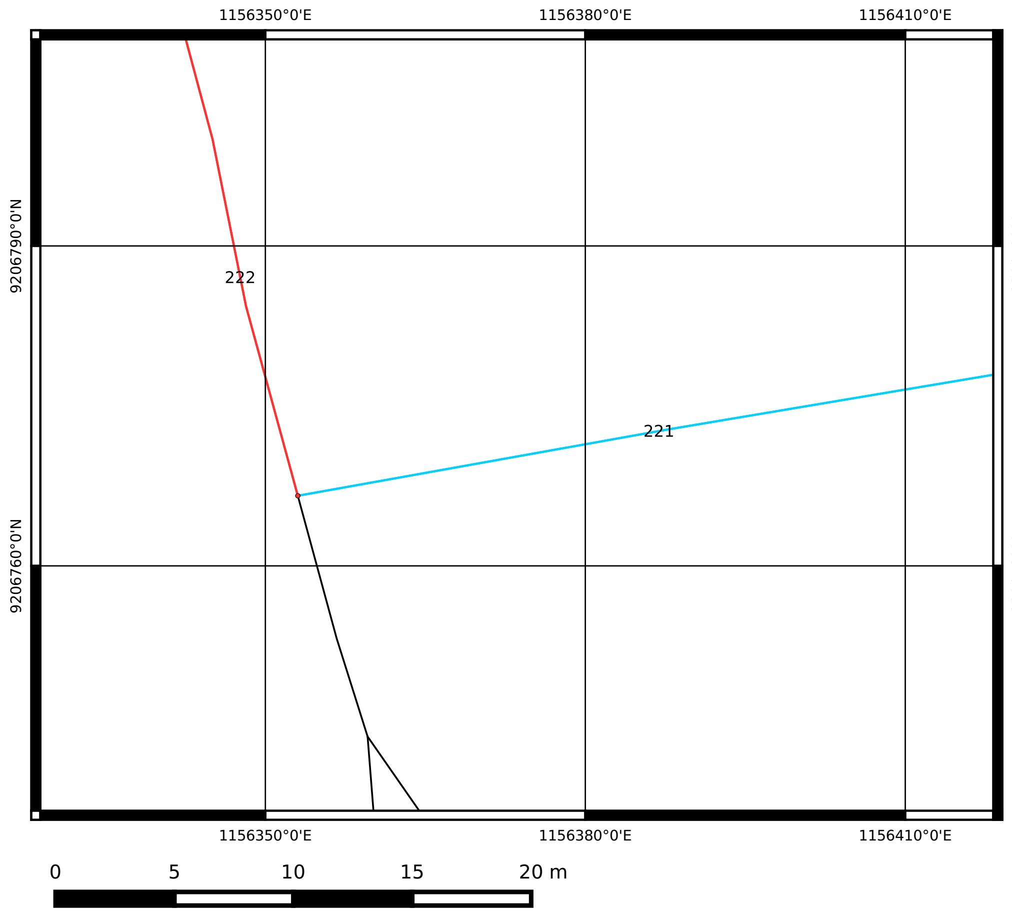
Legend

🟥 Not Required

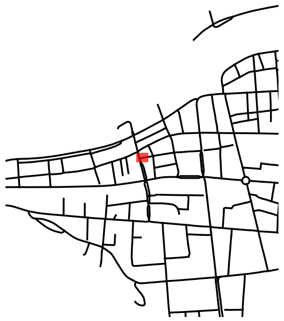
🟦 Required

— Not used road



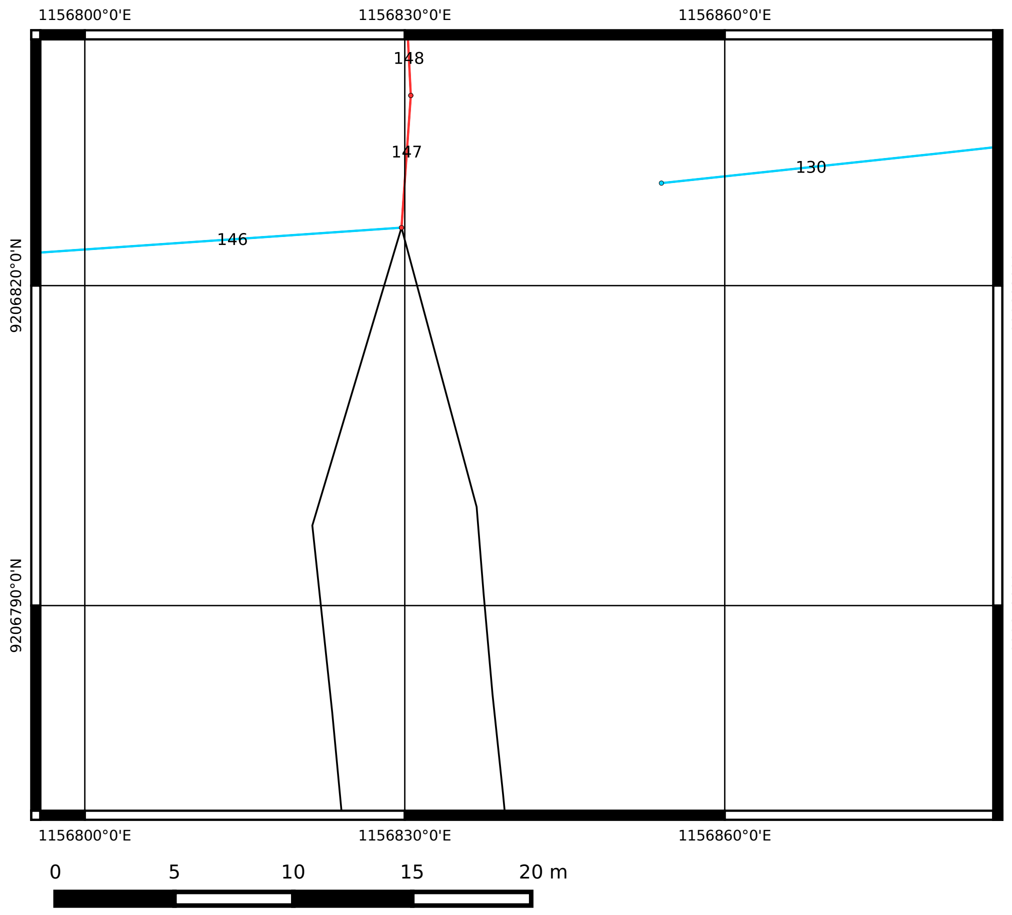


Details for the generated route 310174_KV_B_Midtbyen

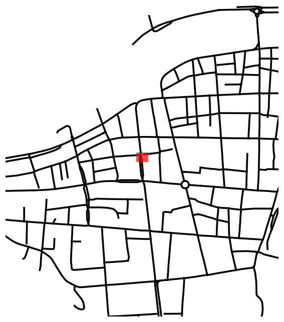


Legend

- 🟥 Not Required
- 🟦 Required
- — Not used road



Details for the generated route 310174_KV_B_Midtbyen

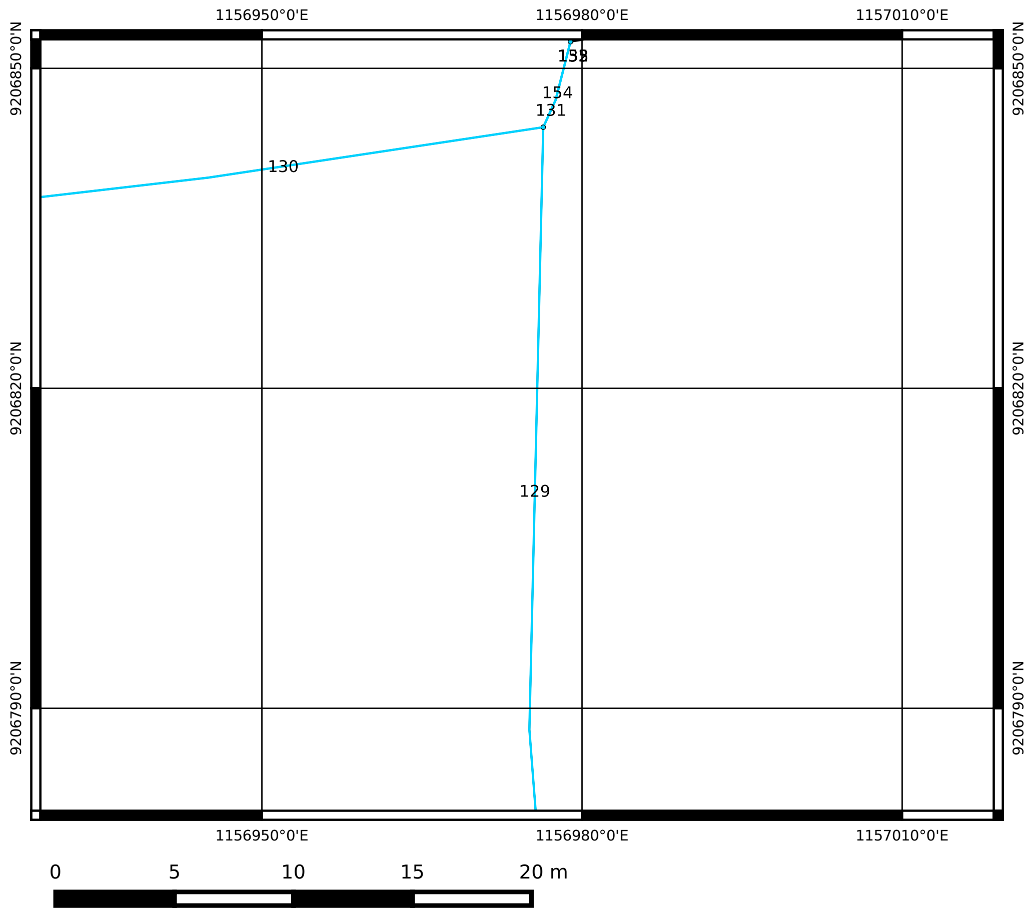


Legend

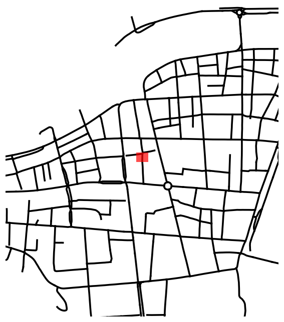
🟥 Not Required

🟦 Required

— Not used road



Details for the generated route 310174_KV_B_Midtbyen

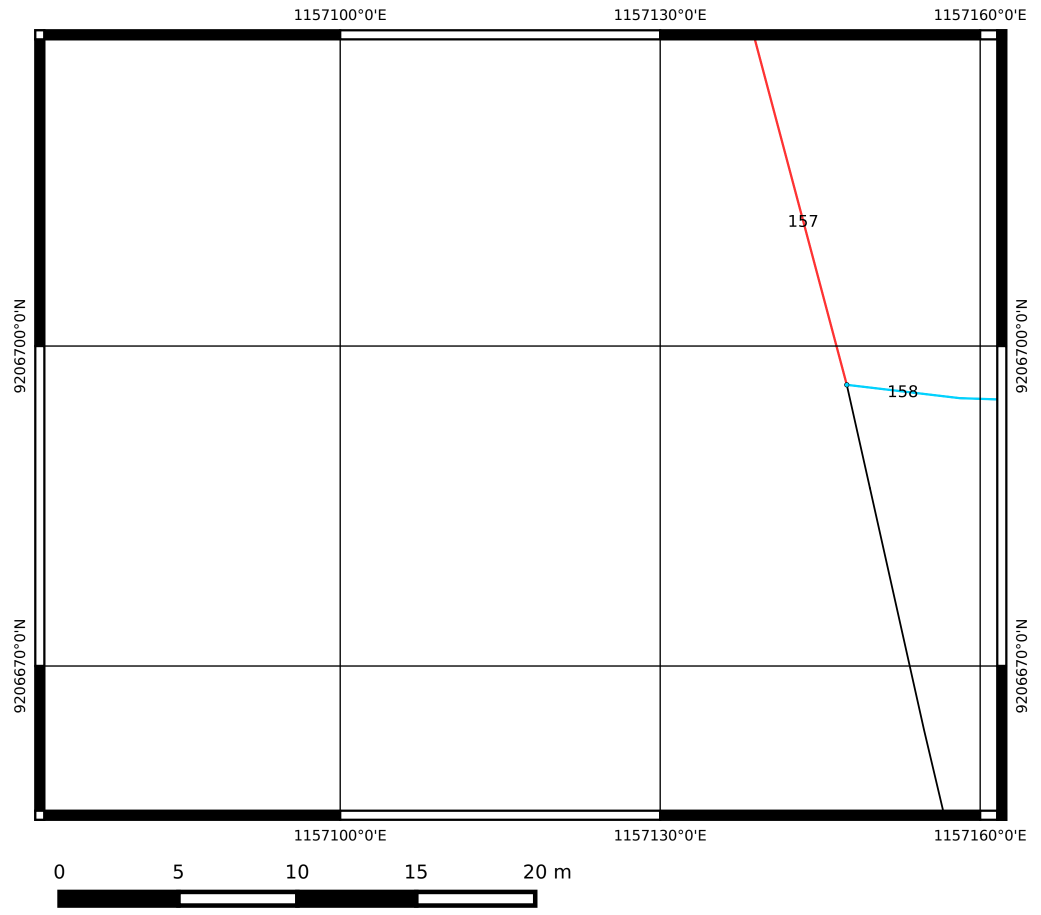


Legend

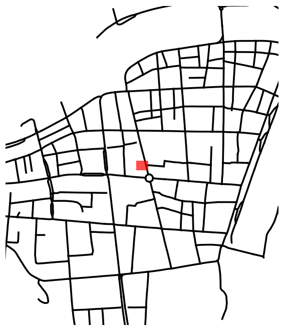
Not Required

Required

Not used road

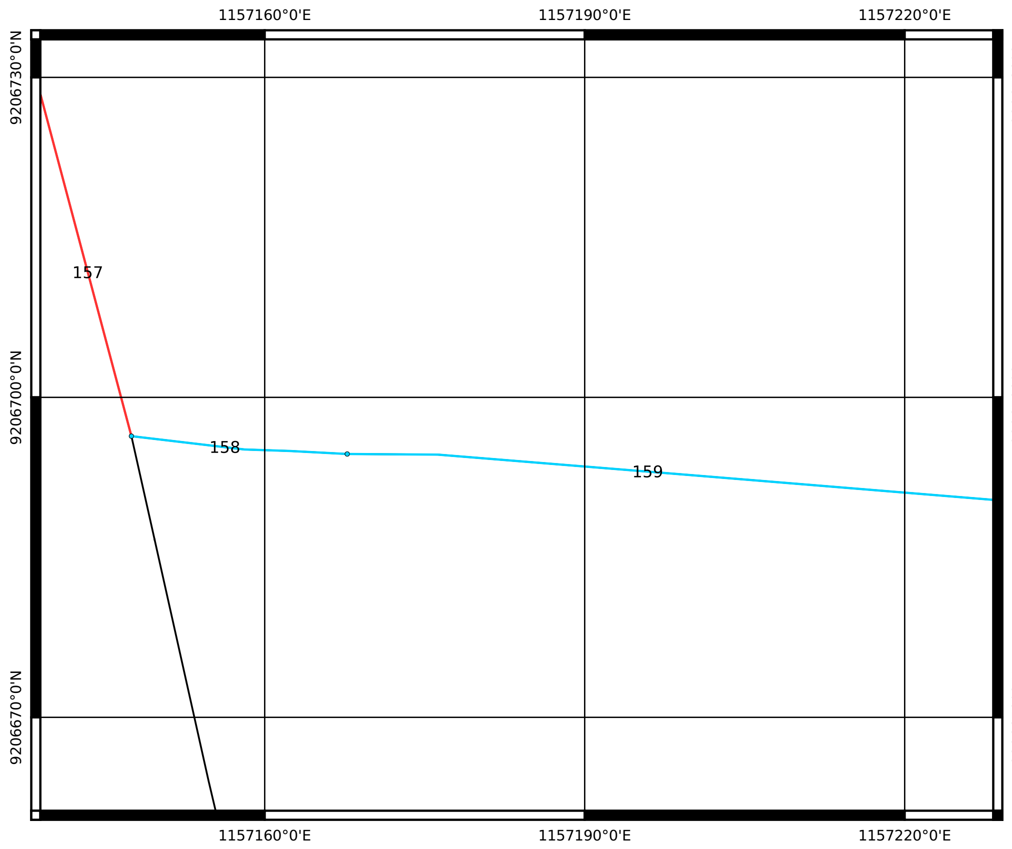


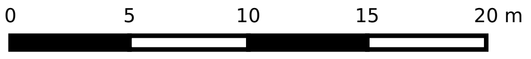
Details for the generated route 310174_KV_B_Midtbyen



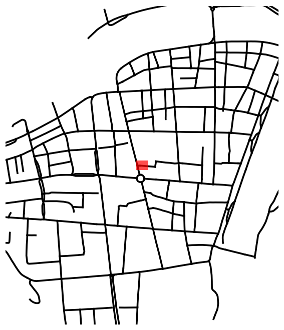
Legend

🟥	Not Required
🟦	Required
—	Not used road



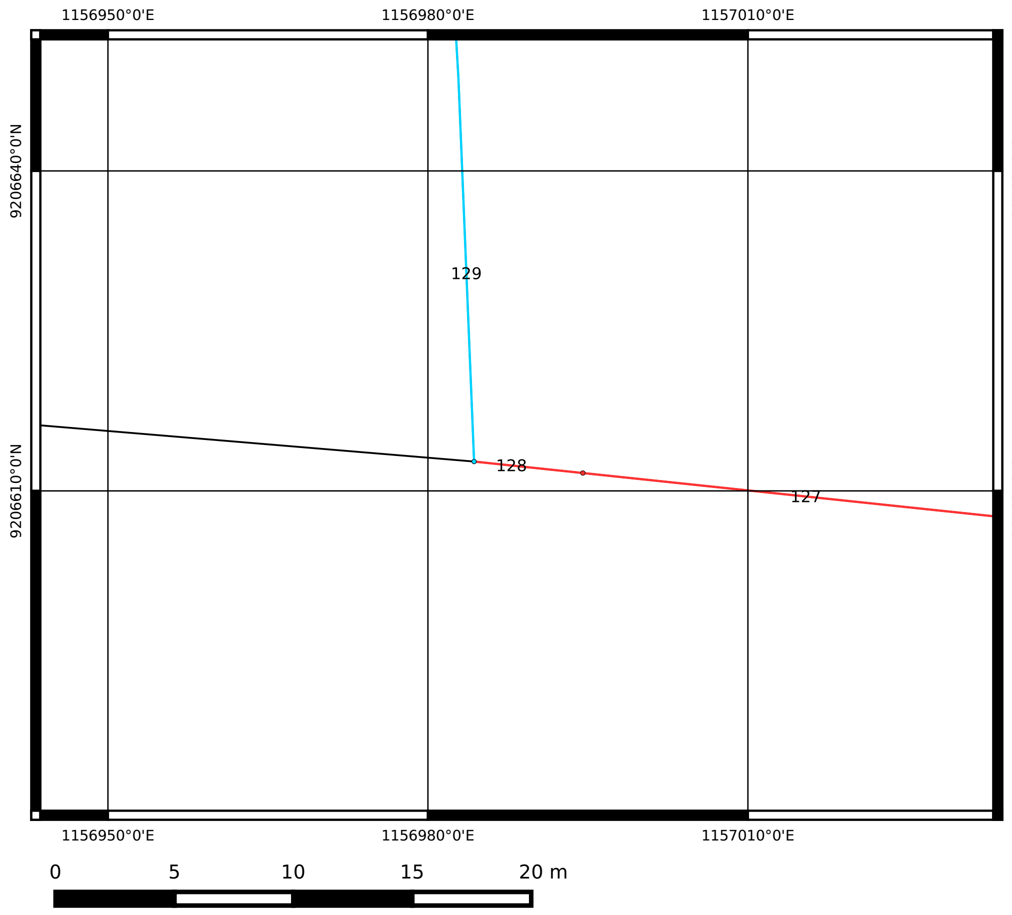


Details for the generated route 310174_KV_B_Midtbyen

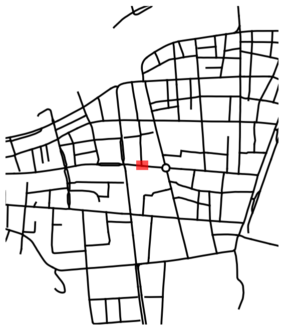


Legend

🟥	Not Required
🟦	Required
—	Not used road

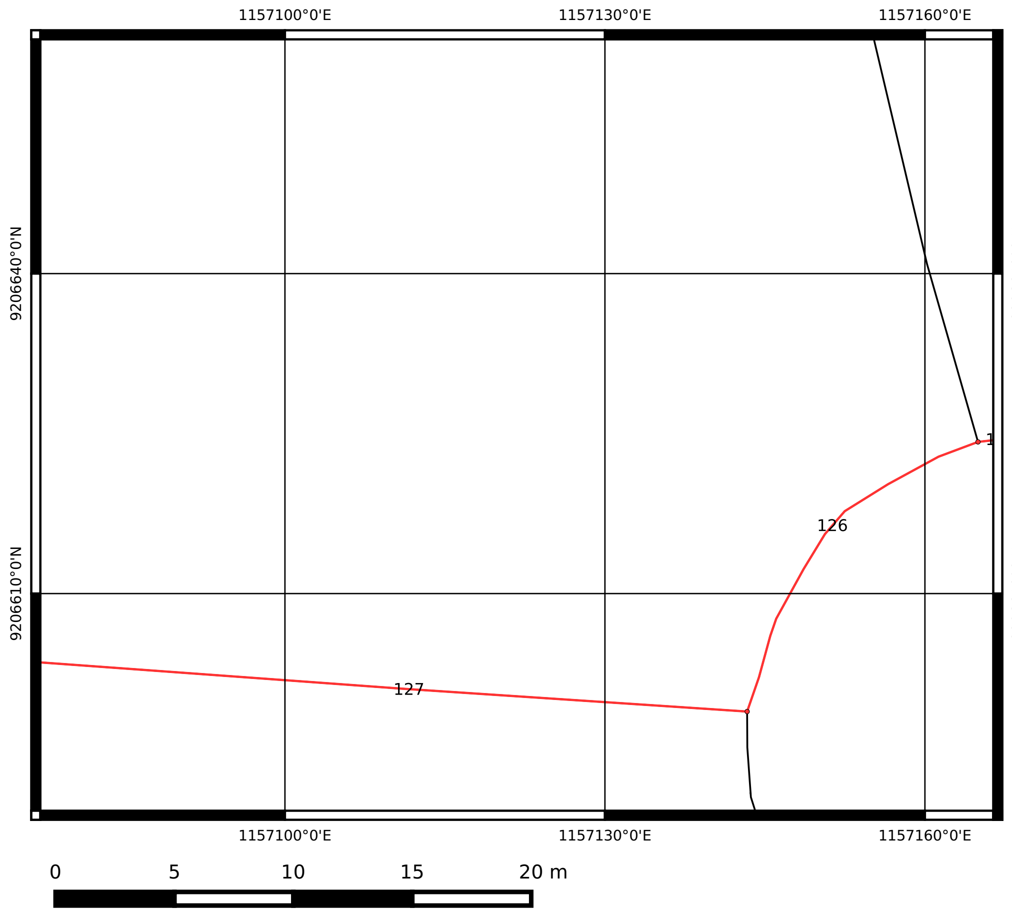


Details for the generated route 310174_KV_B_Midtbyen

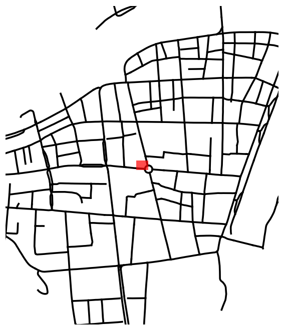


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

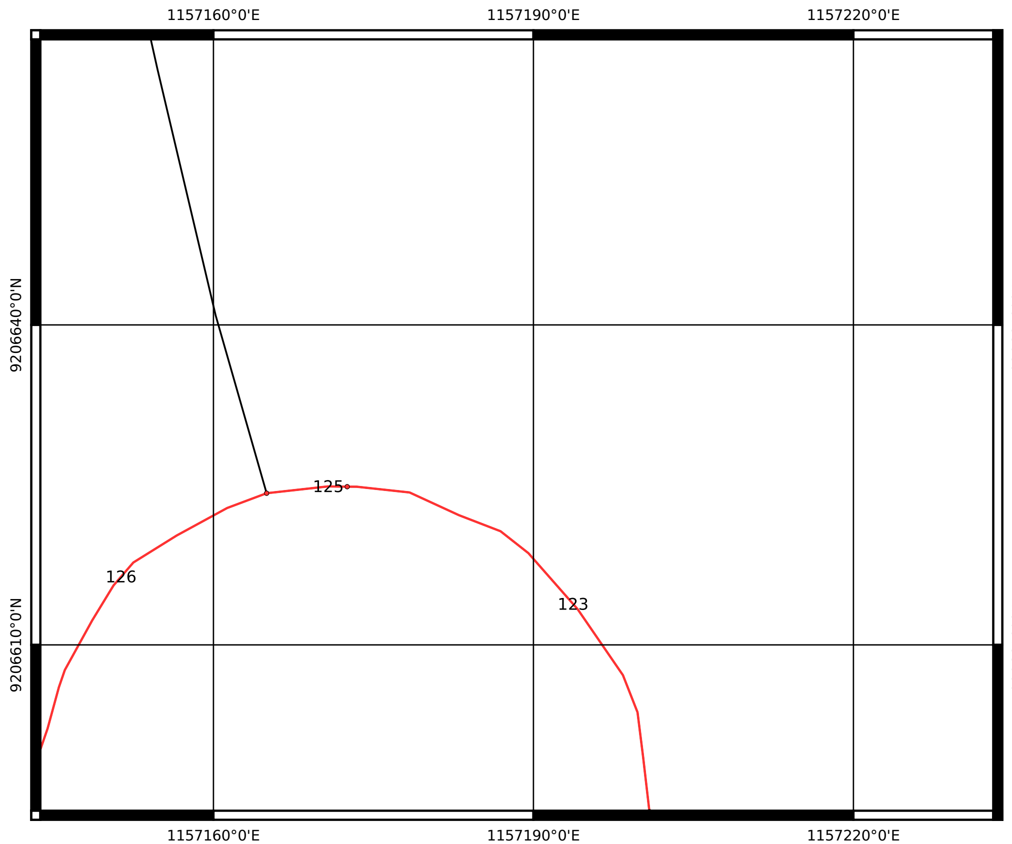


Legend

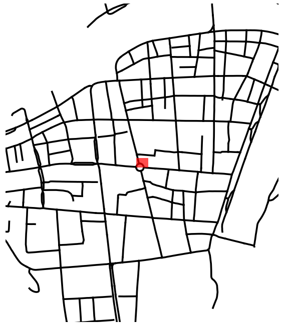
🟥 Not Required

🟦 Required

— Not used road

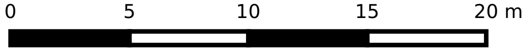


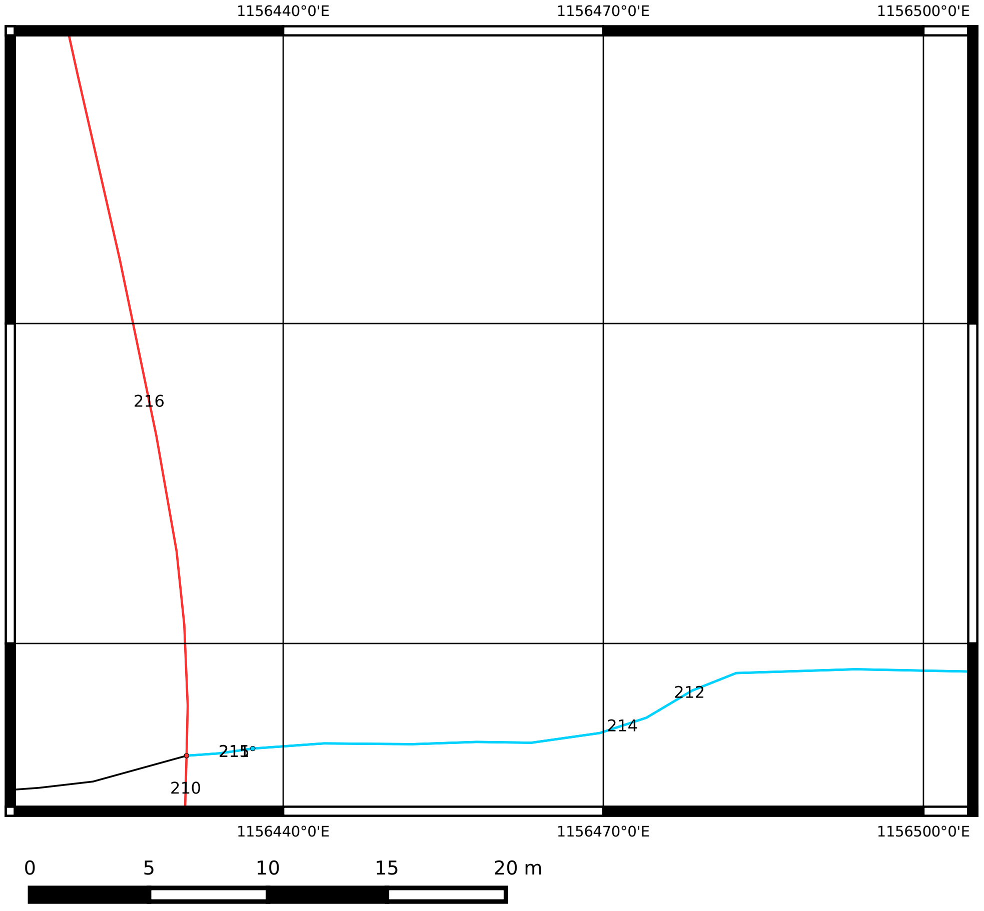
Details for the generated route 310174_KV_B_Midtbyen



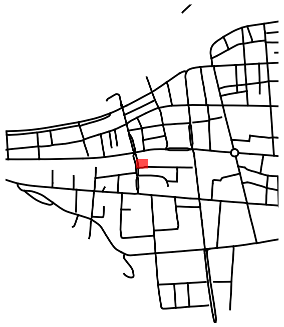
Legend

🟥	Not Required
🟦	Required
—	Not used road



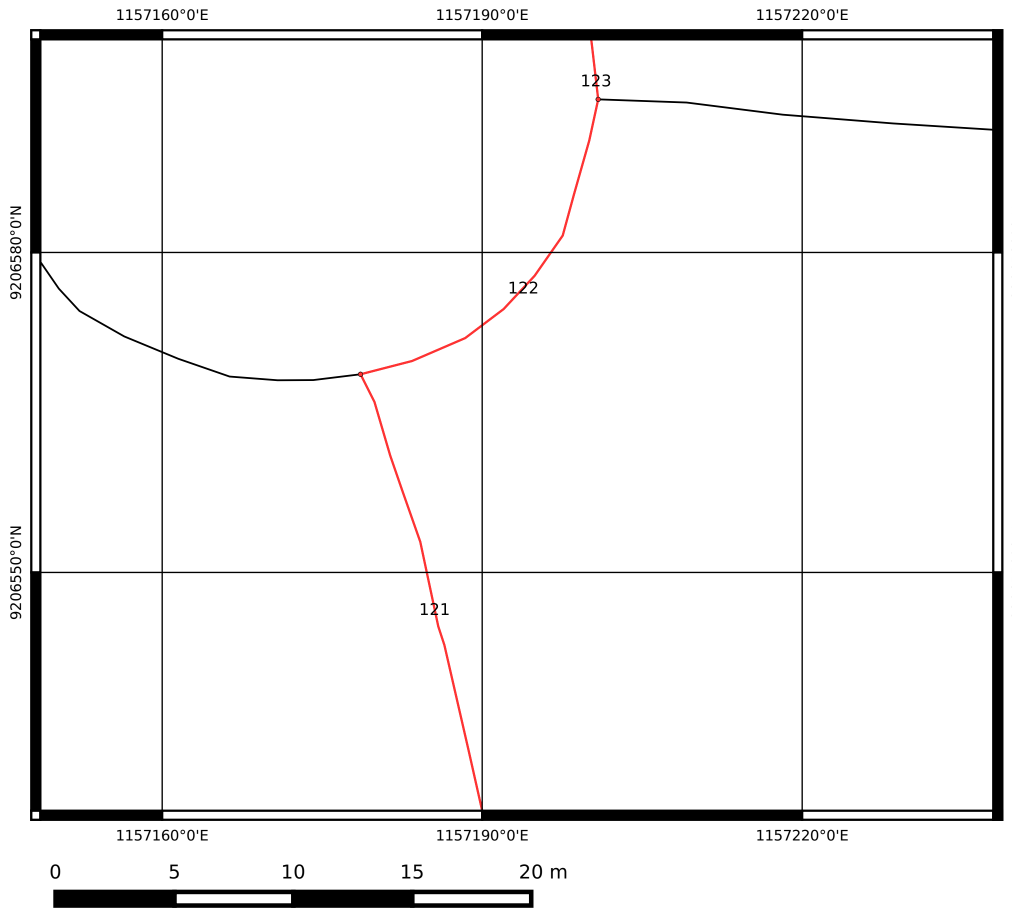


Details for the generated route 310174_KV_B_Midtbyen

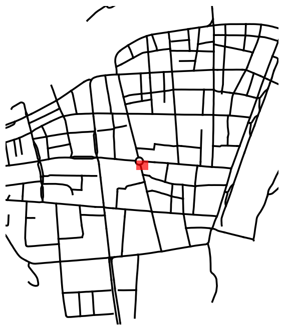


Legend

🟥	Not Required
🟦	Required
—	Not used road

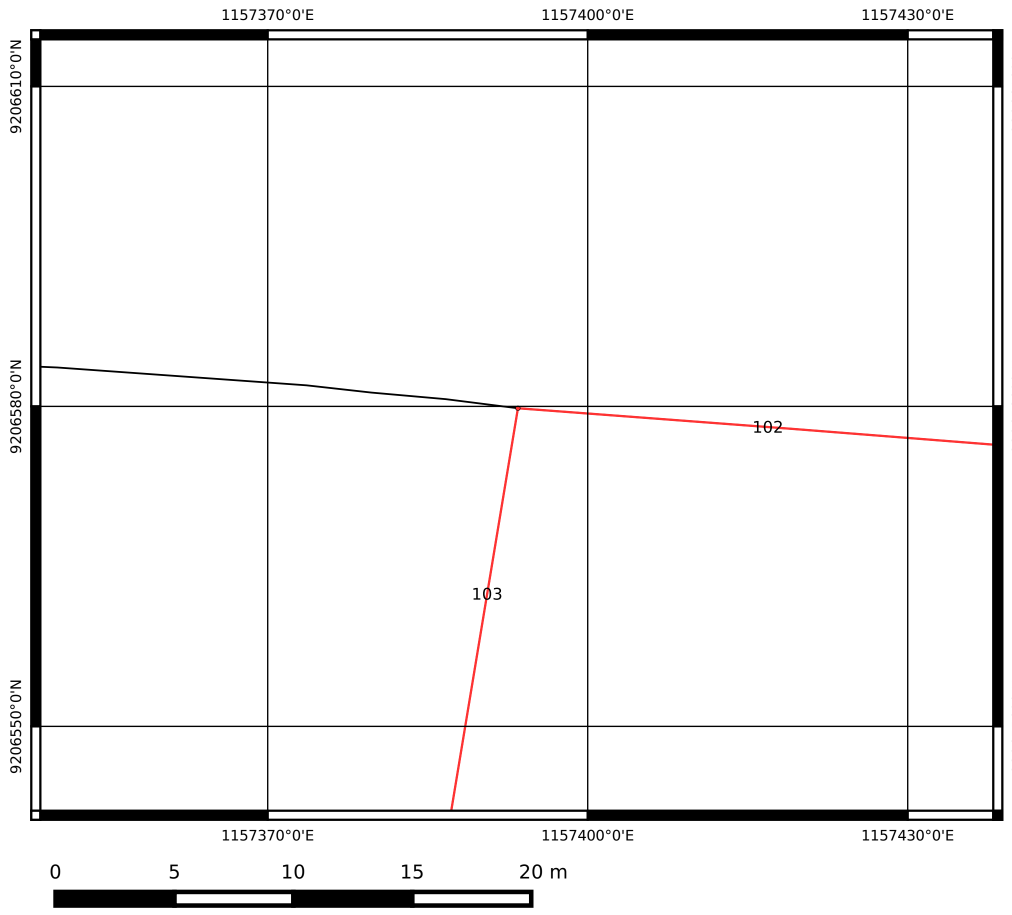


Details for the generated route 310174_KV_B_Midtbyen

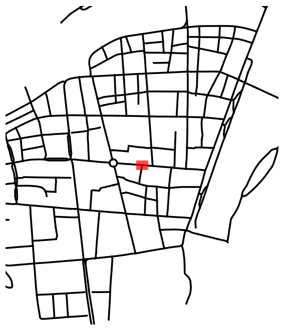


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

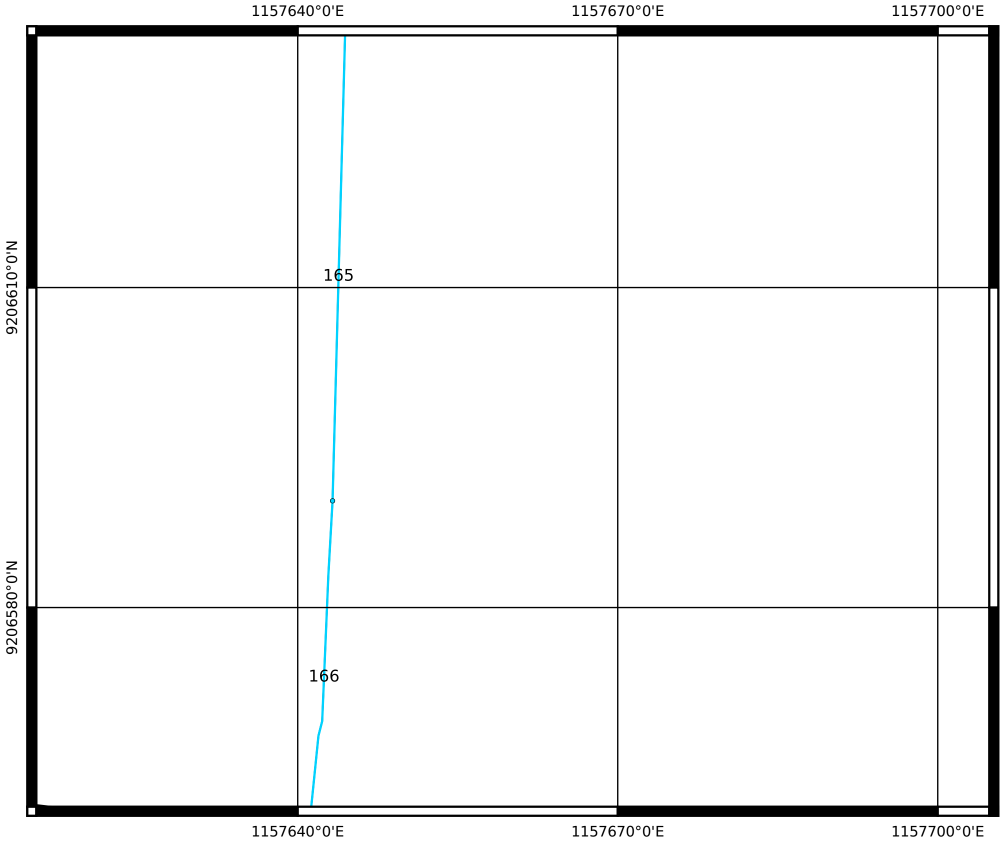


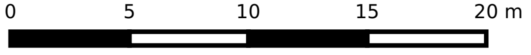
Legend

🟥 Not Required

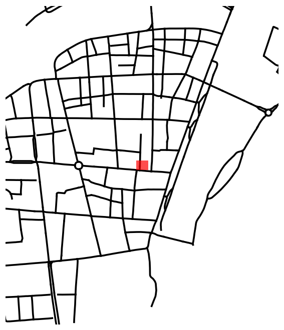
🟦 Required

— Not used road

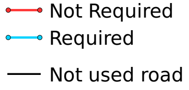


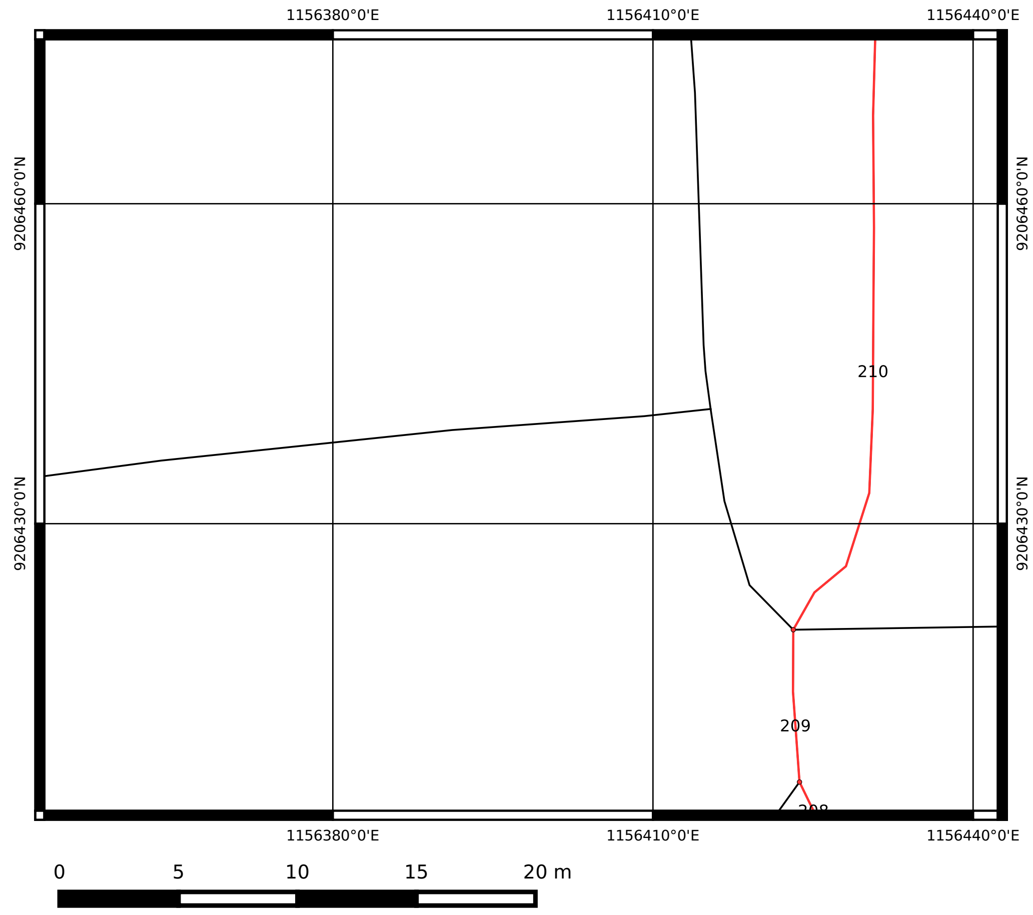


Details for the generated route 310174_KV_B_Midtbyen

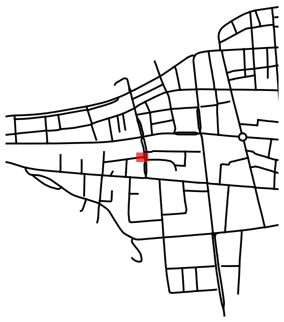


Legend



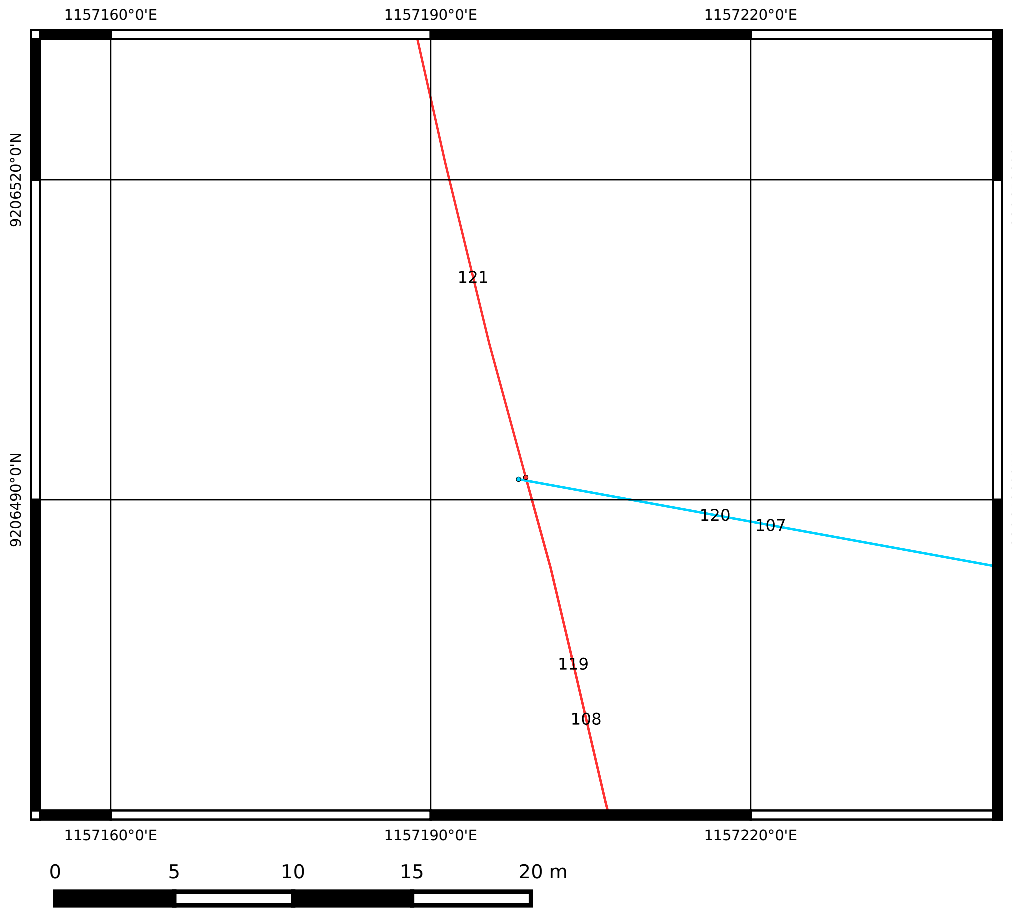


Details for the generated route 310174_KV_B_Midtbyen

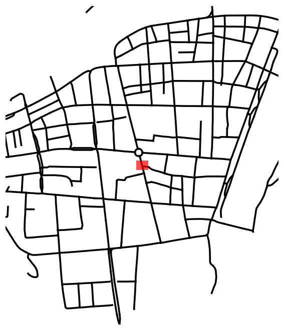


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

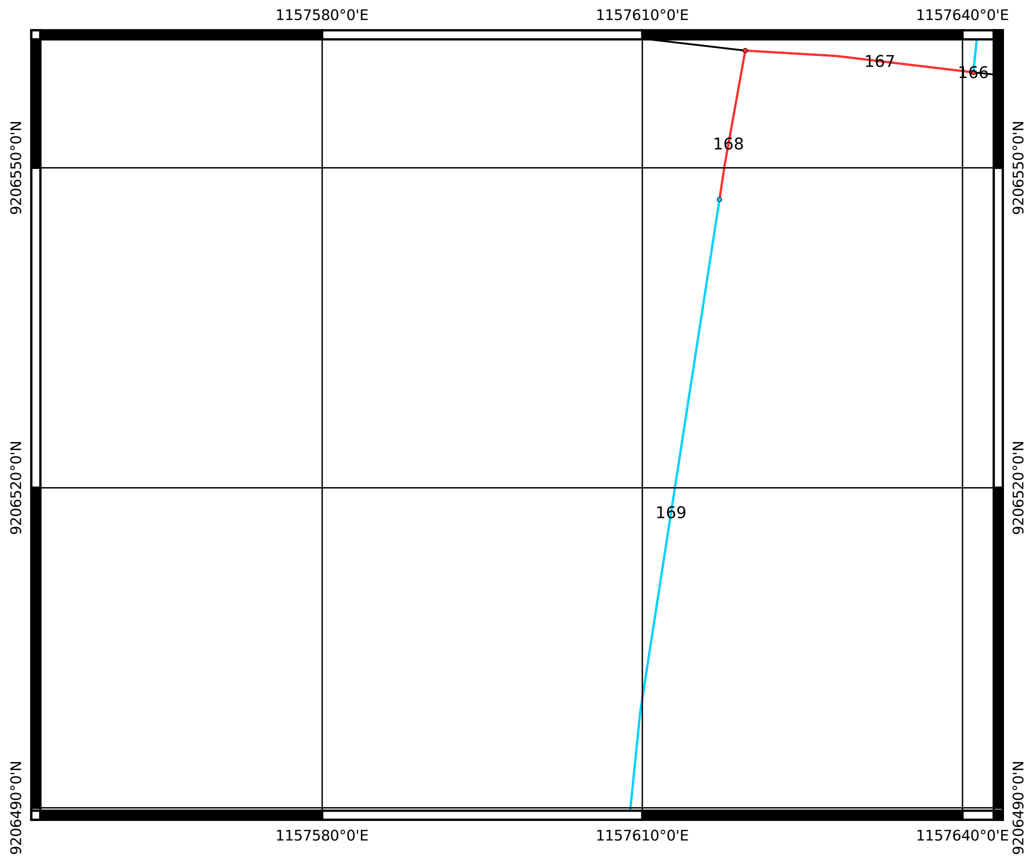


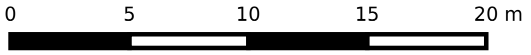
Legend

🟥 Not Required

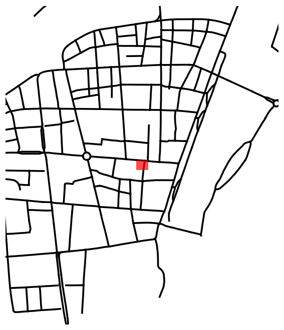
🟦 Required

— Not used road



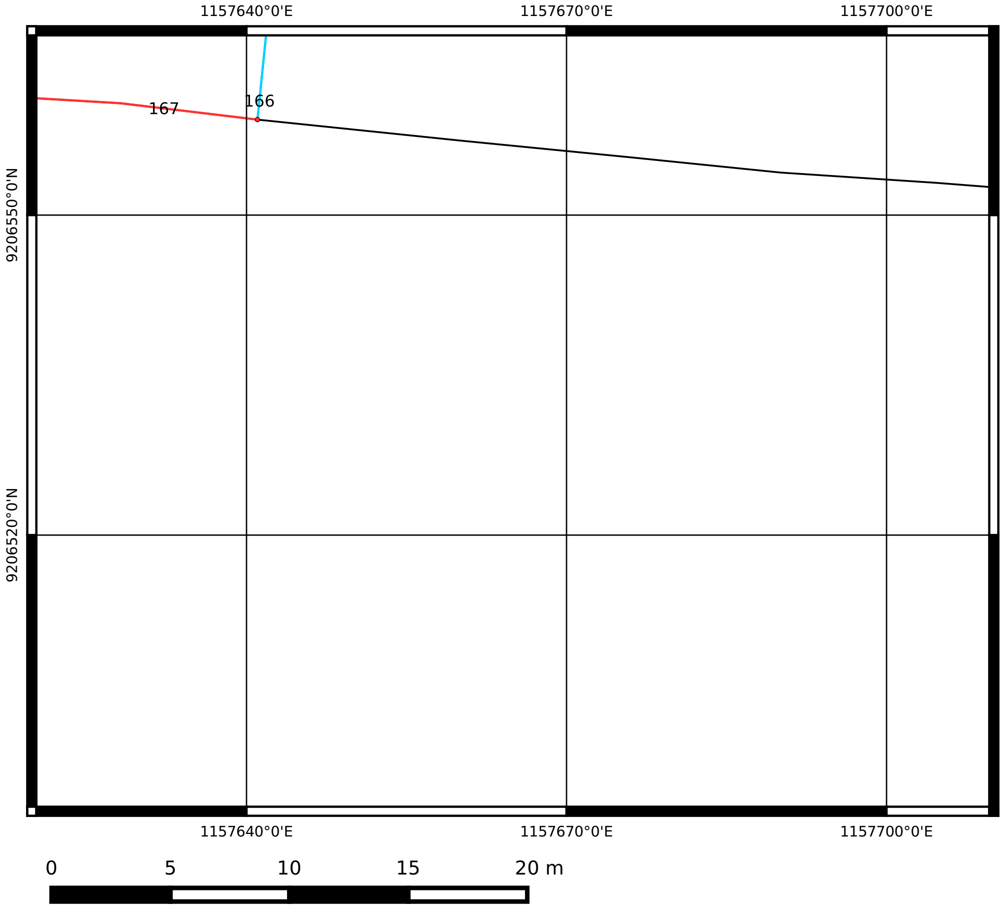


Details for the generated route 310174_KV_B_Midtbyen

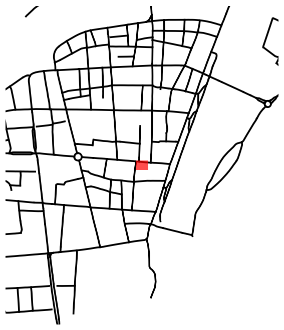


Legend

- Not Required
- Required
- Not used road



Details for the generated route 310174_KV_B_Midtbyen

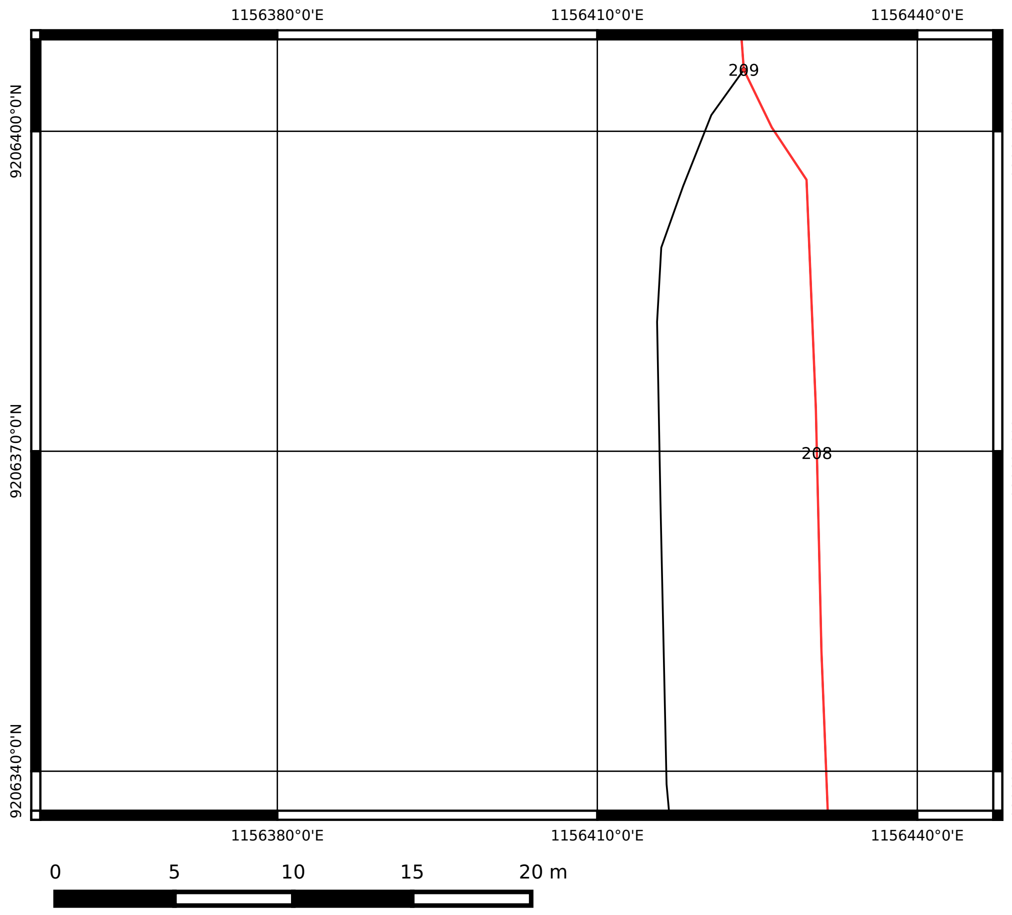


Legend

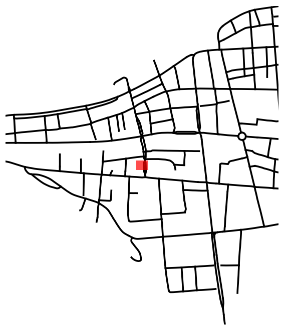
🟥 Not Required

🟦 Required

— Not used road



Details for the generated route 310174_KV_B_Midtbyen

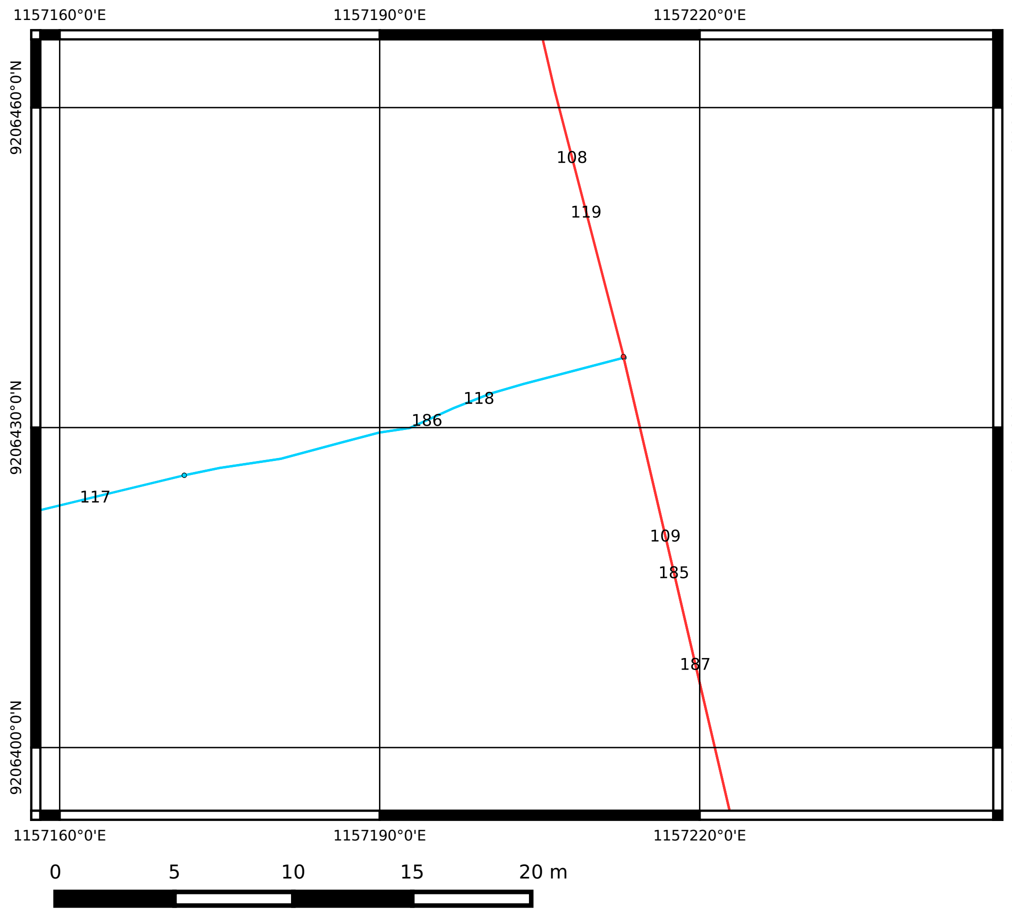


Legend

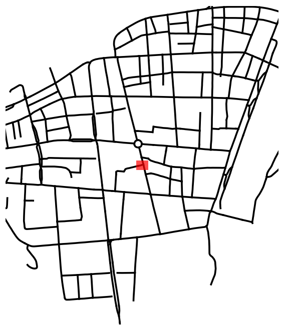
🟥 Not Required

🟦 Required

— Not used road

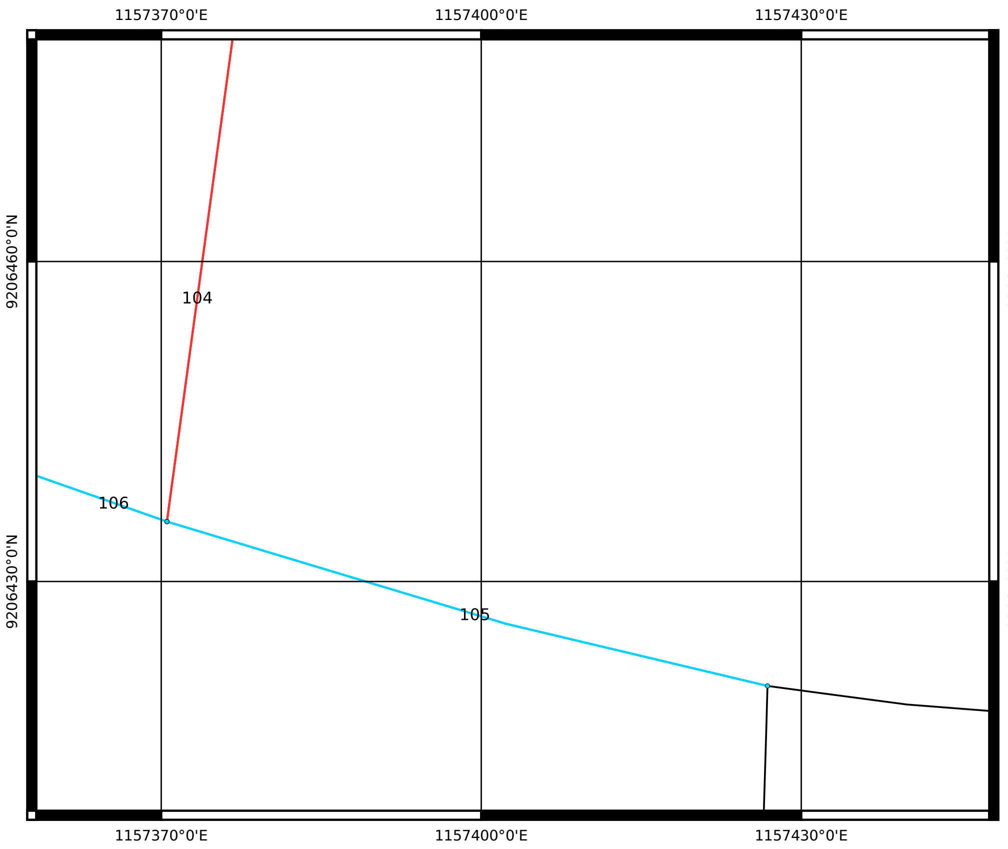


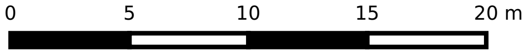
Details for the generated route 310174_KV_B_Midtbyen



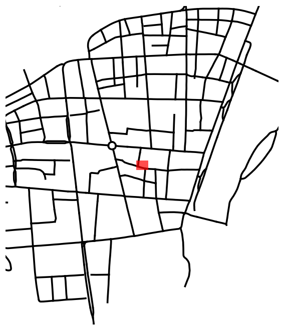
Legend

🟥	Not Required
🟦	Required
—	Not used road



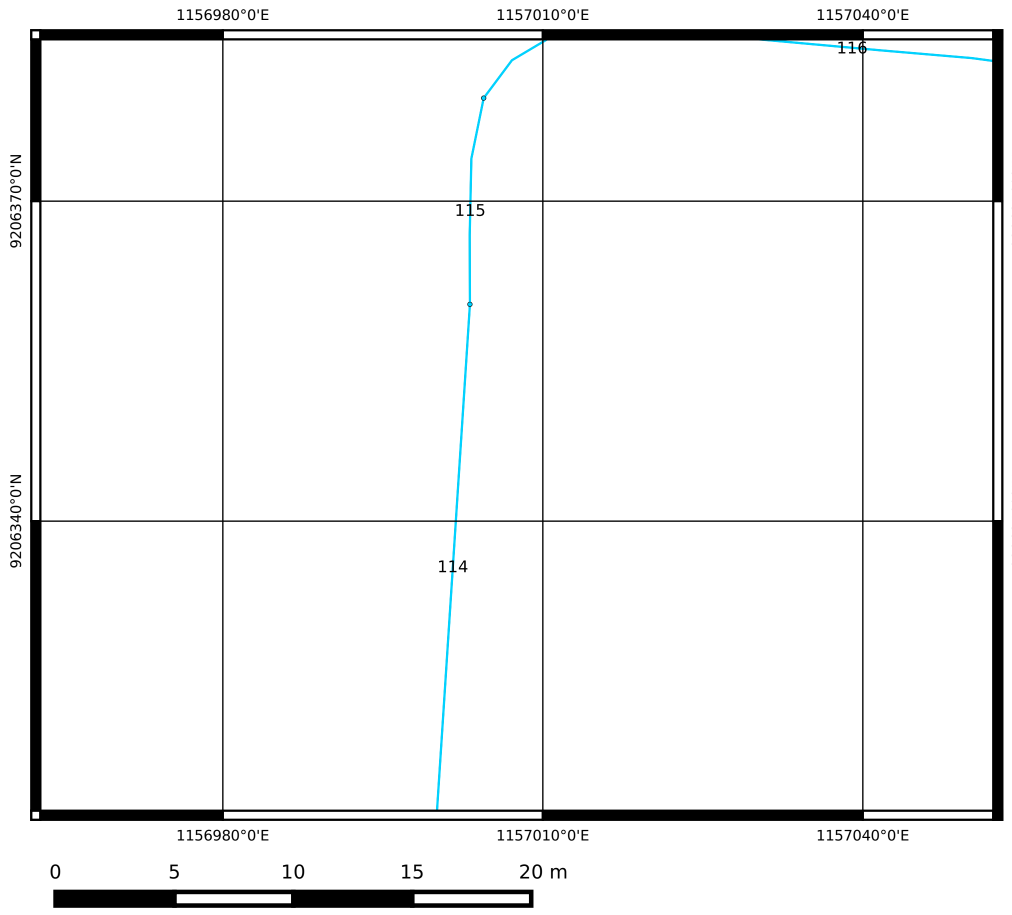


Details for the generated route 310174_KV_B_Midtbyen

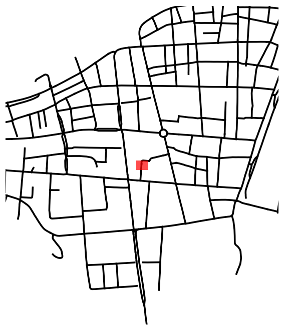


Legend

🟥	Not Required
🟦	Required
—	Not used road

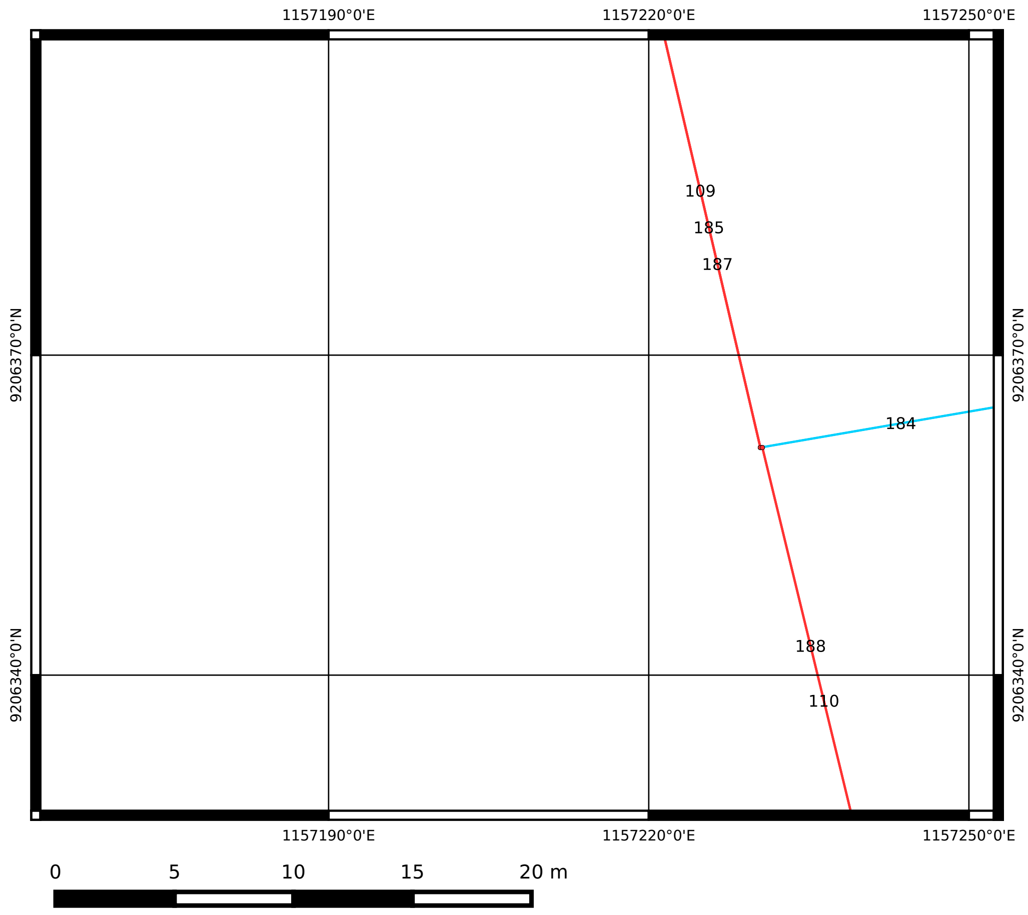


Details for the generated route 310174_KV_B_Midtbyen

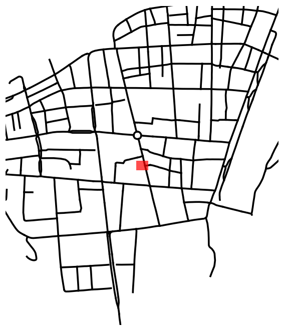


Legend

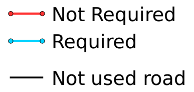
- 🟥 Not Required
- 🟦 Required
- — Not used road

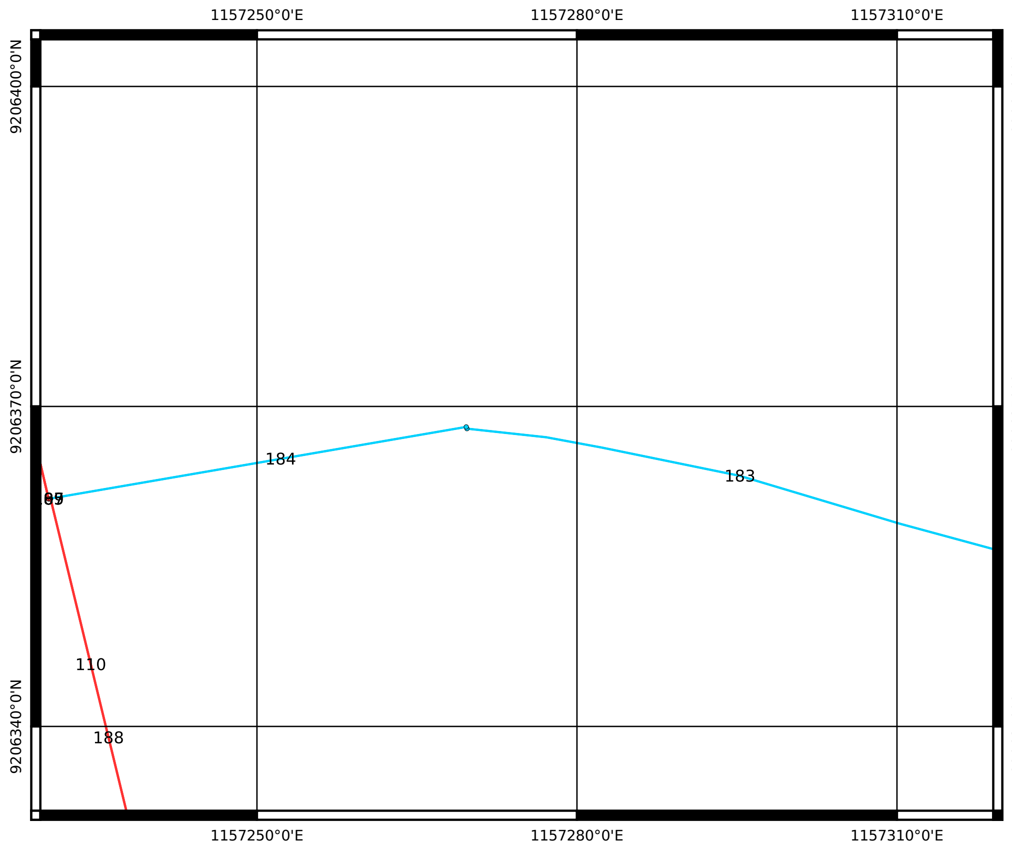


Details for the generated route 310174_KV_B_Midtbyen

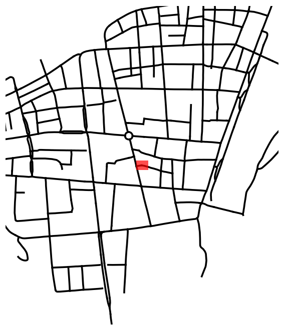


Legend



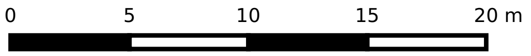


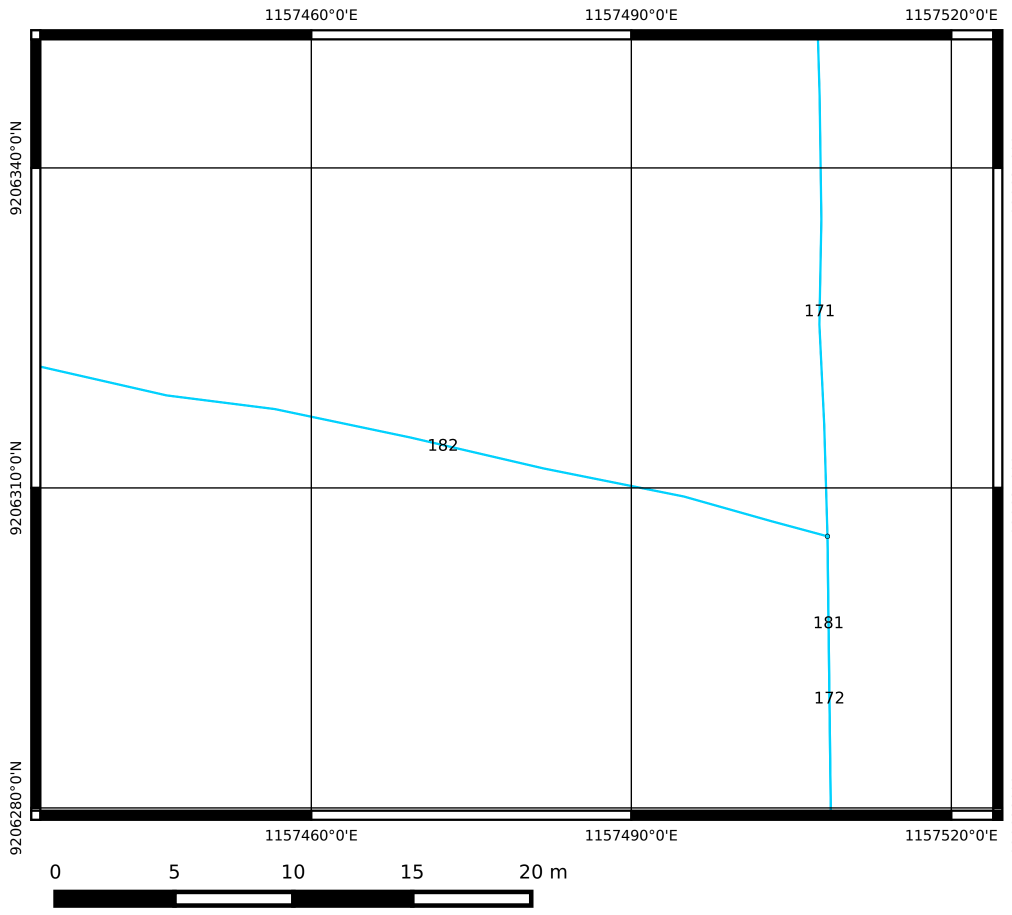
Details for the generated route 310174_KV_B_Midtbyen



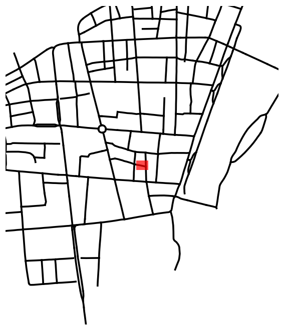
Legend

- Not Required
- Required
- Not used road



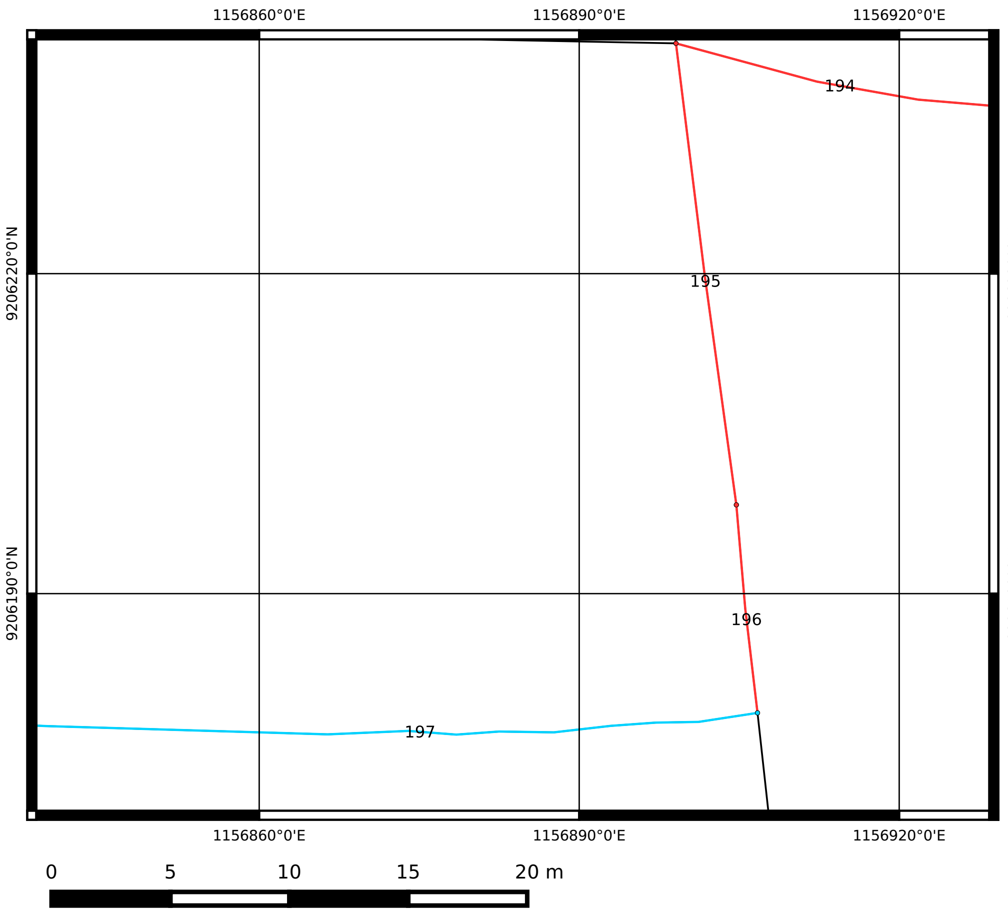


Details for the generated route 310174_KV_B_Midtbyen

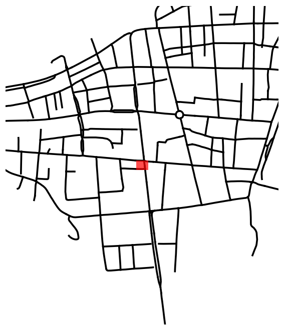


Legend

🟥 Not Required
🟦 Required
— Not used road

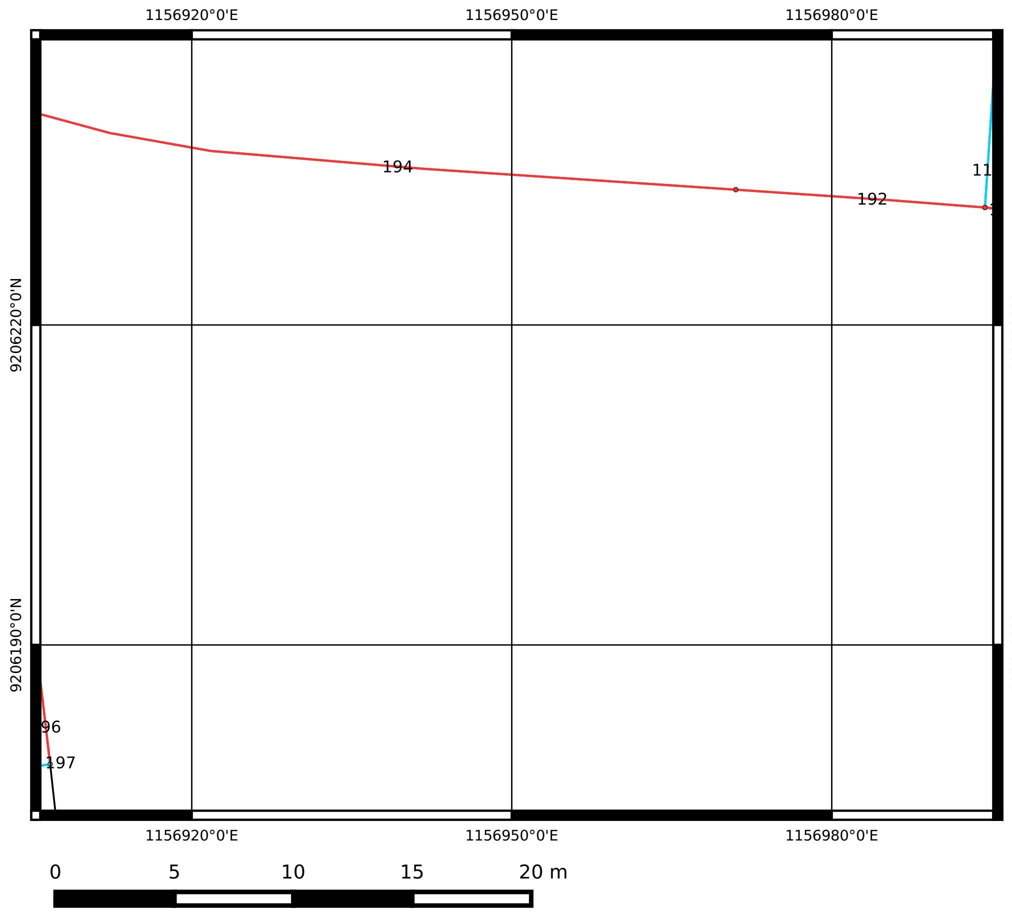


Details for the generated route 310174_KV_B_Midtbyen

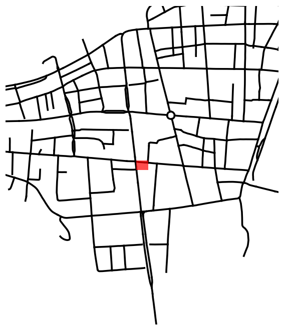


Legend

🟥	Not Required
🟦	Required
—	Not used road

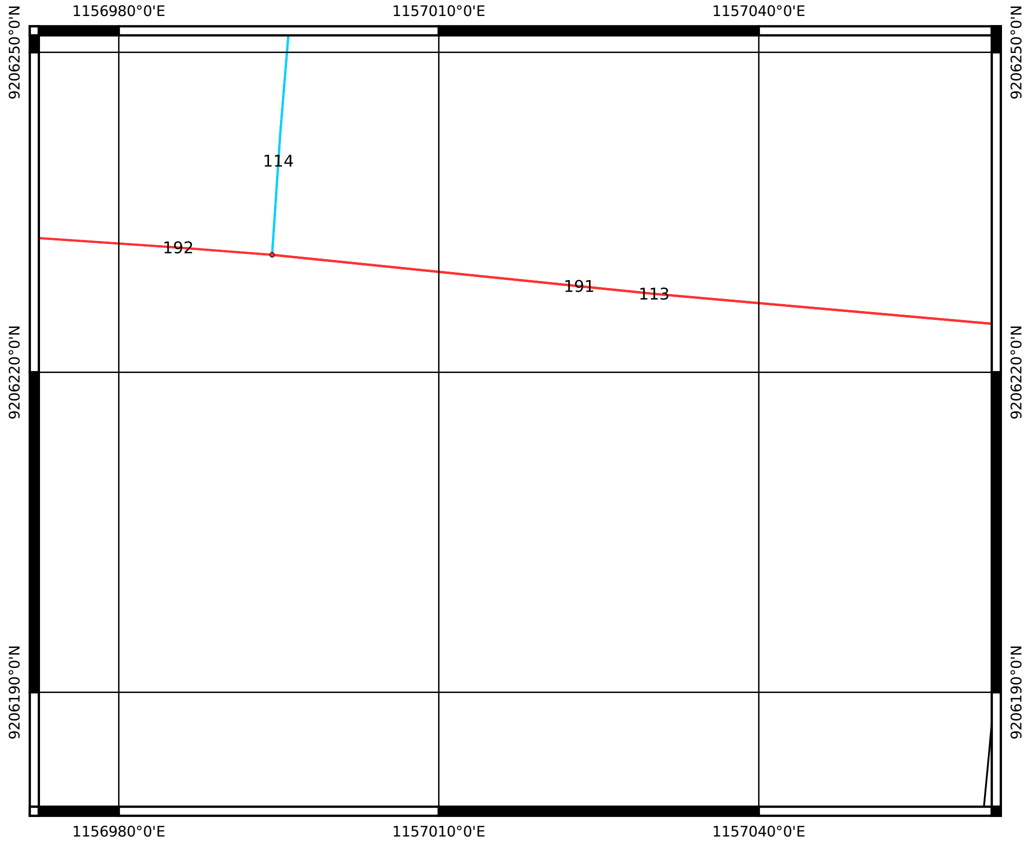


Details for the generated route 310174_KV_B_Midtbyen

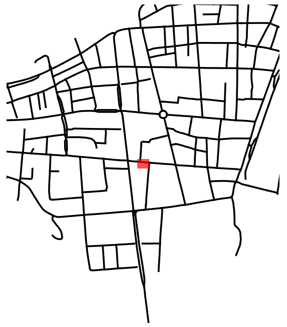


Legend

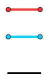
- Not Required
- Required
- Not used road



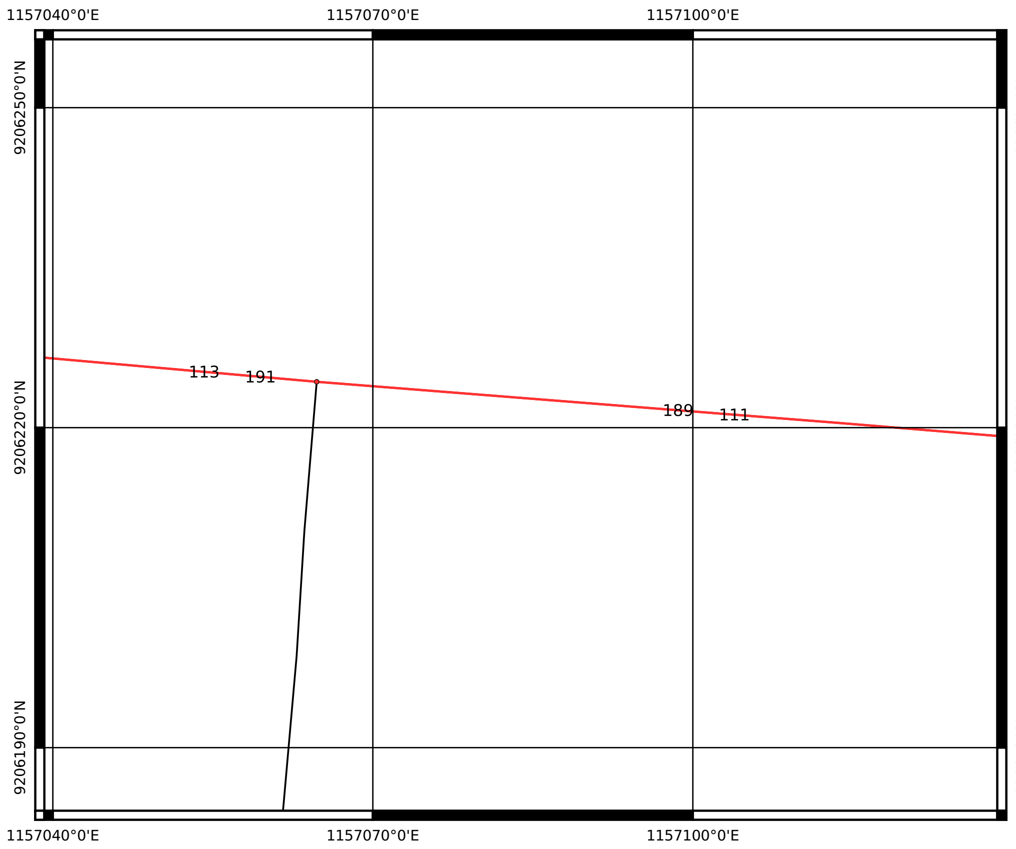
Details for the generated route 310174_KV_B_Midtbyen



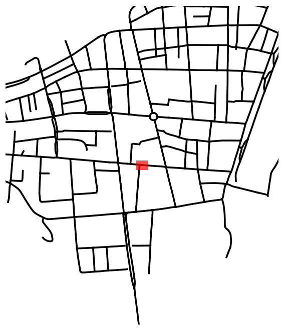
Legend



🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310174_KV_B_Midtbyen

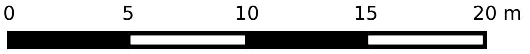


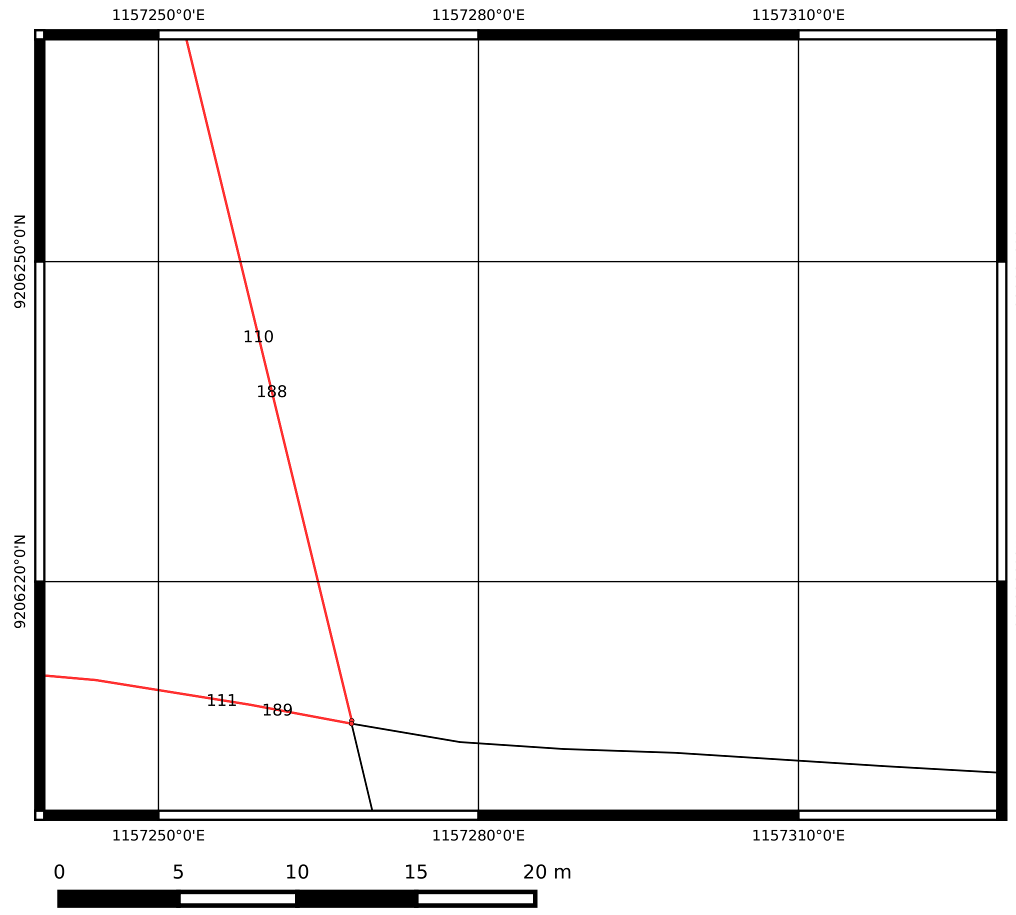
Legend

🟥 Not Required

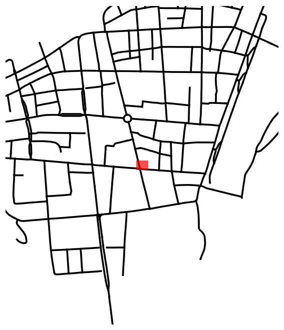
🟦 Required

— Not used road



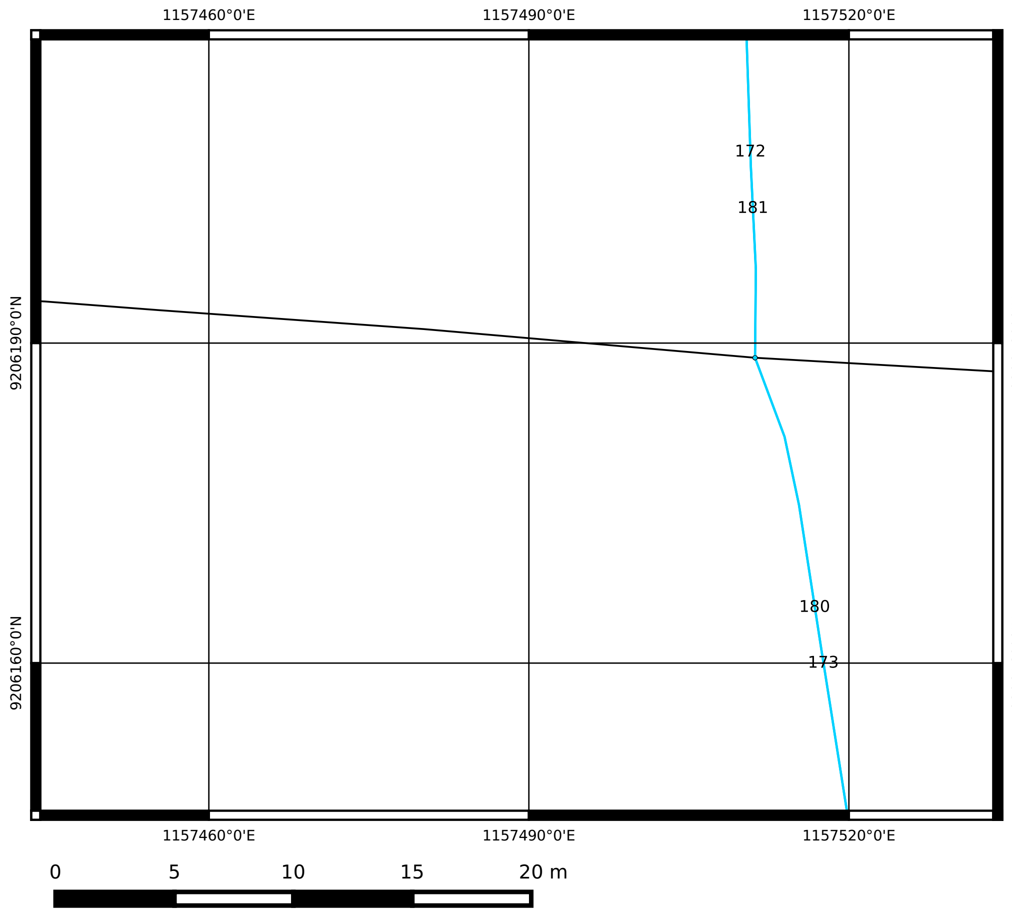


Details for the generated route 310174_KV_B_Midtbyen

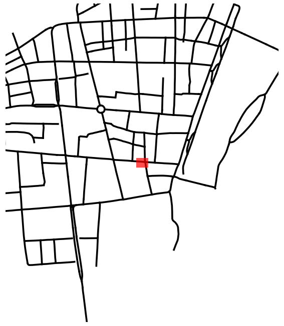


Legend

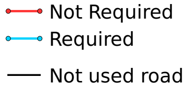
🟥	Not Required
🟦	Required
—	Not used road

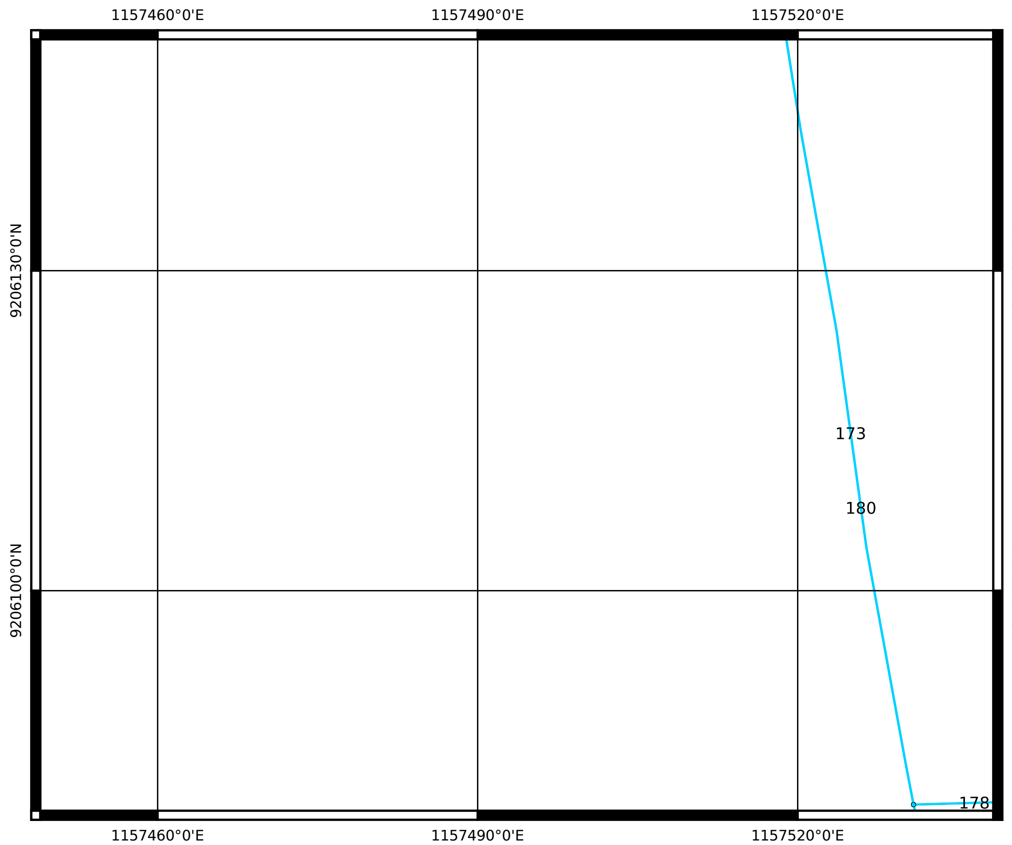


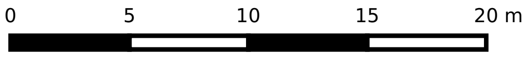
Details for the generated route 310174_KV_B_Midtbyen



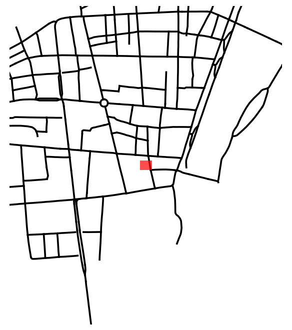
Legend



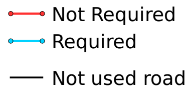


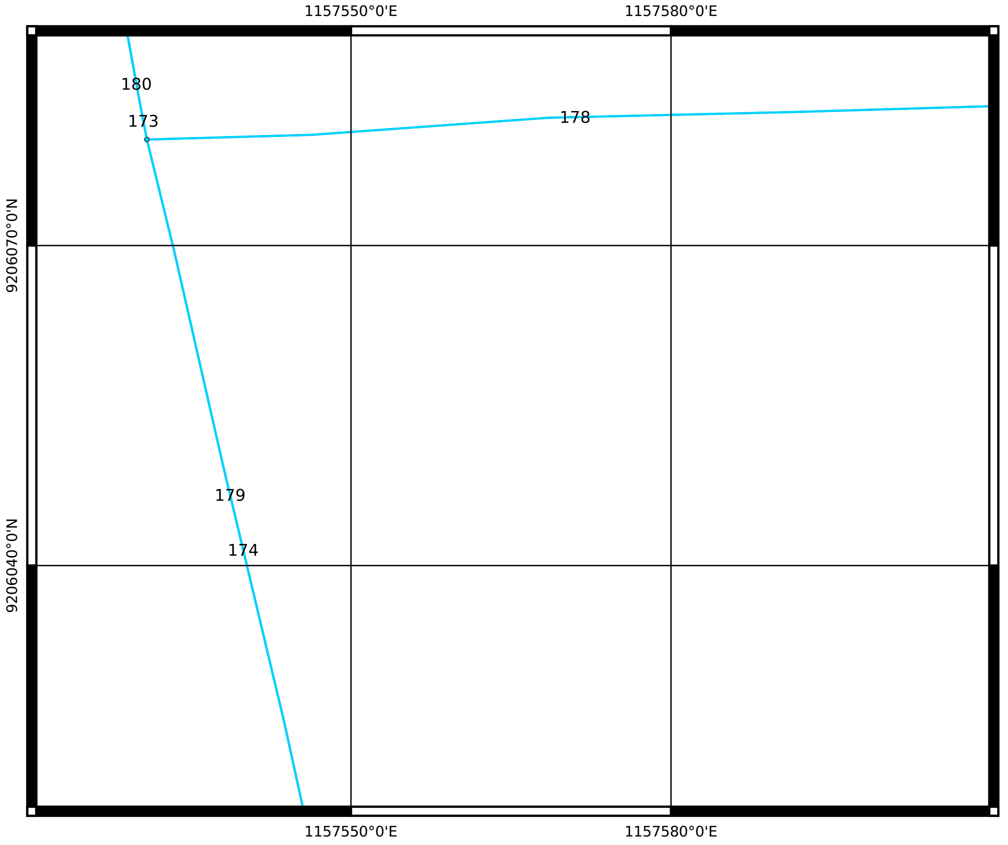


Details for the generated route 310174_KV_B_Midtbyen

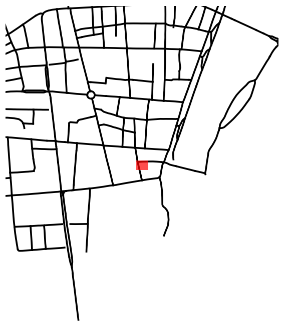


Legend



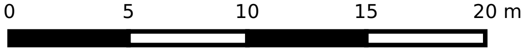


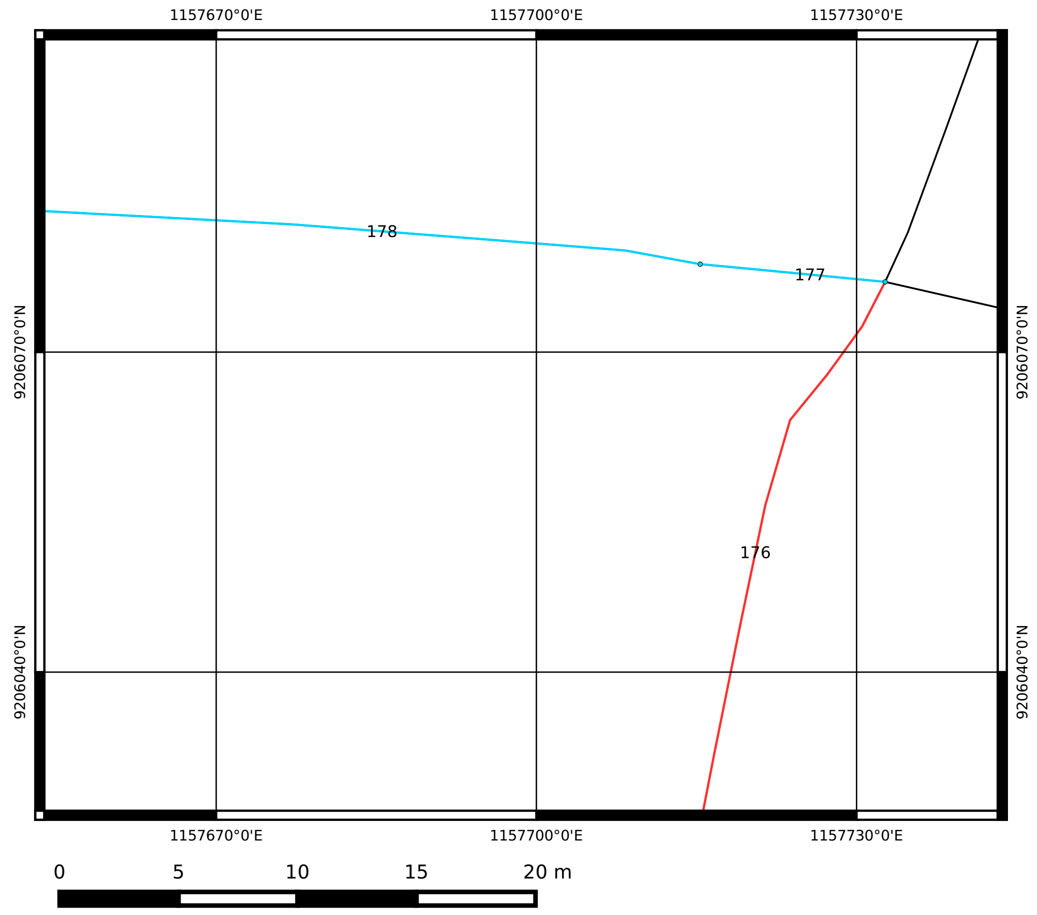
Details for the generated route 310174_KV_B_Midtbyen



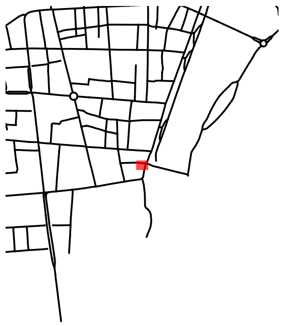
Legend

🟥	Not Required
🟦	Required
—	Not used road

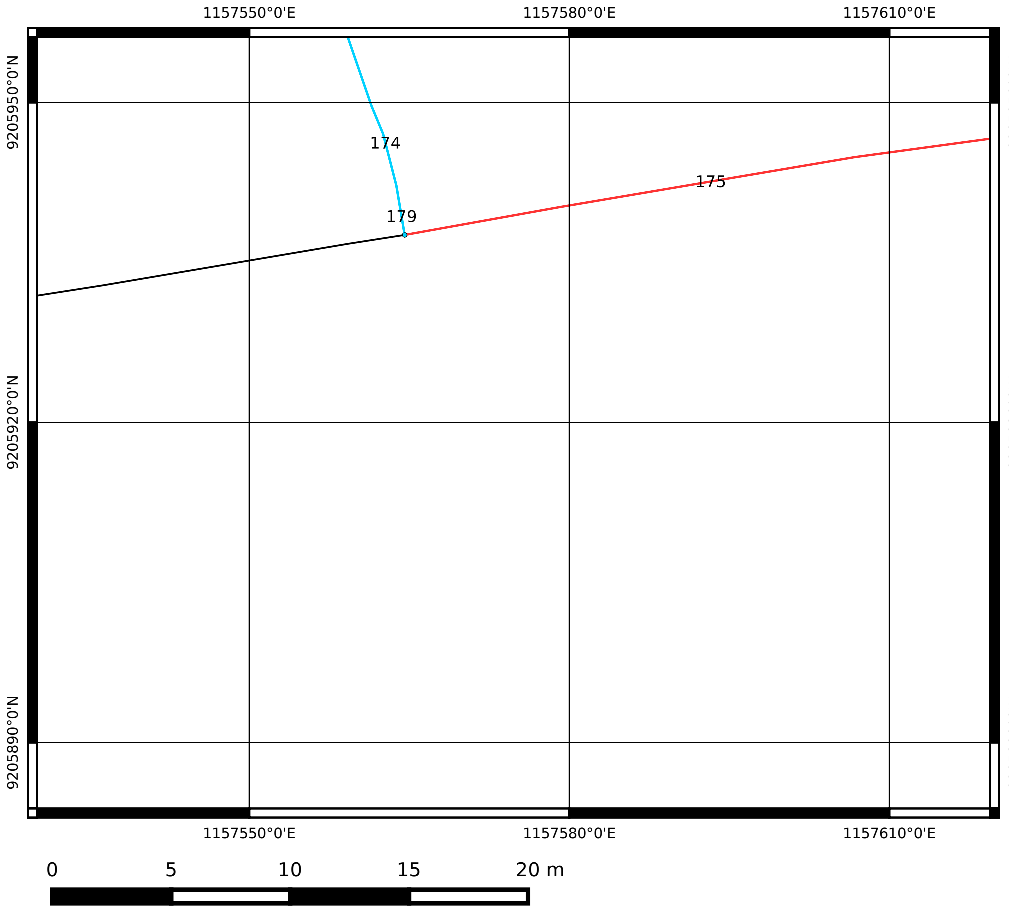




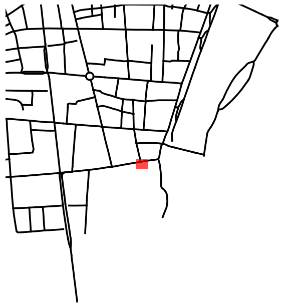
Details for the generated route 310174_KV_B_Midtbyen



Legend



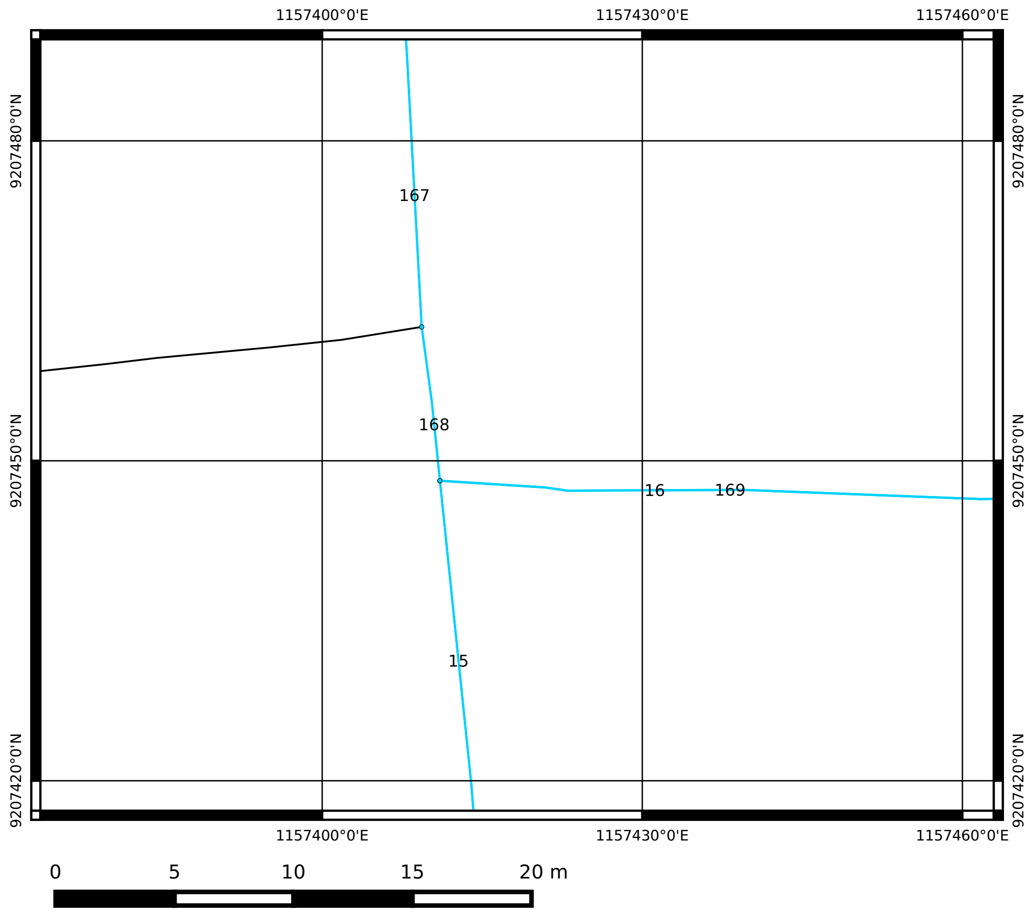
Details for the generated route 310174_KV_B_Midtbyen



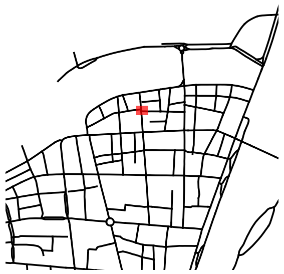
Legend

- Not Required
- Required
- Not used road

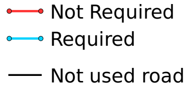
G.2 Detailed Generated Map for Route 310179_KV_H

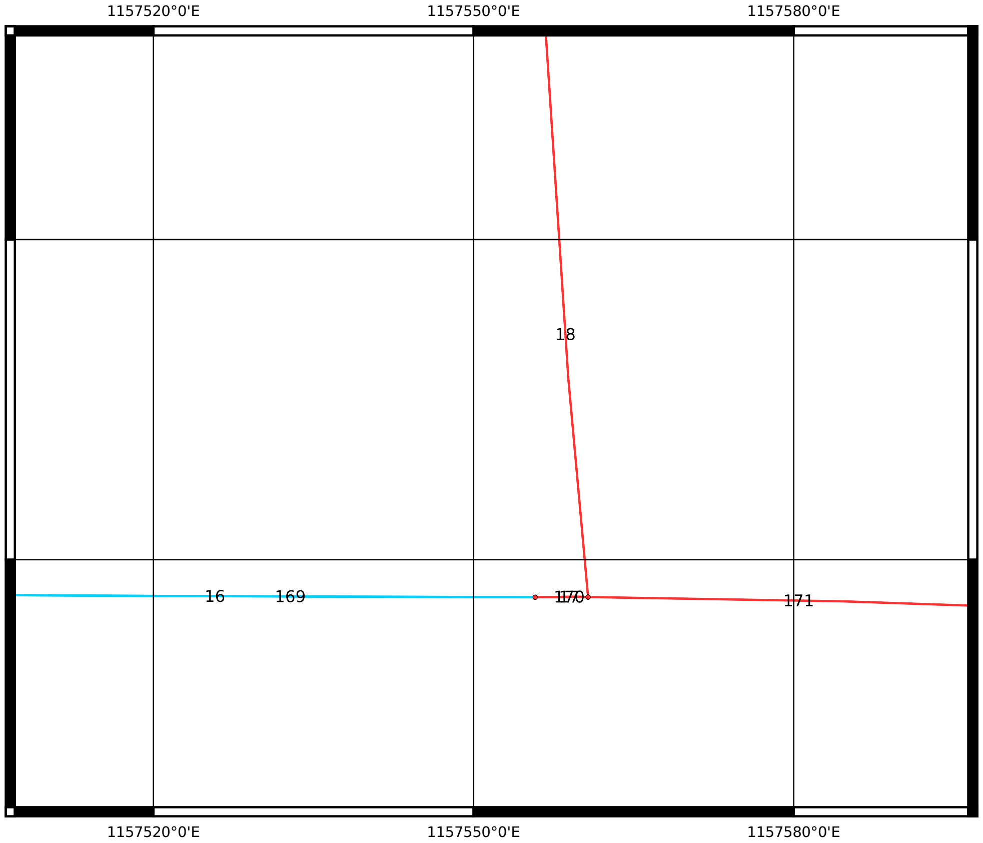


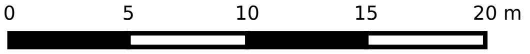
Details for the generated route 310179_KV_H_Midtbyen



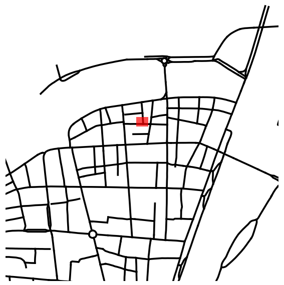
Legend





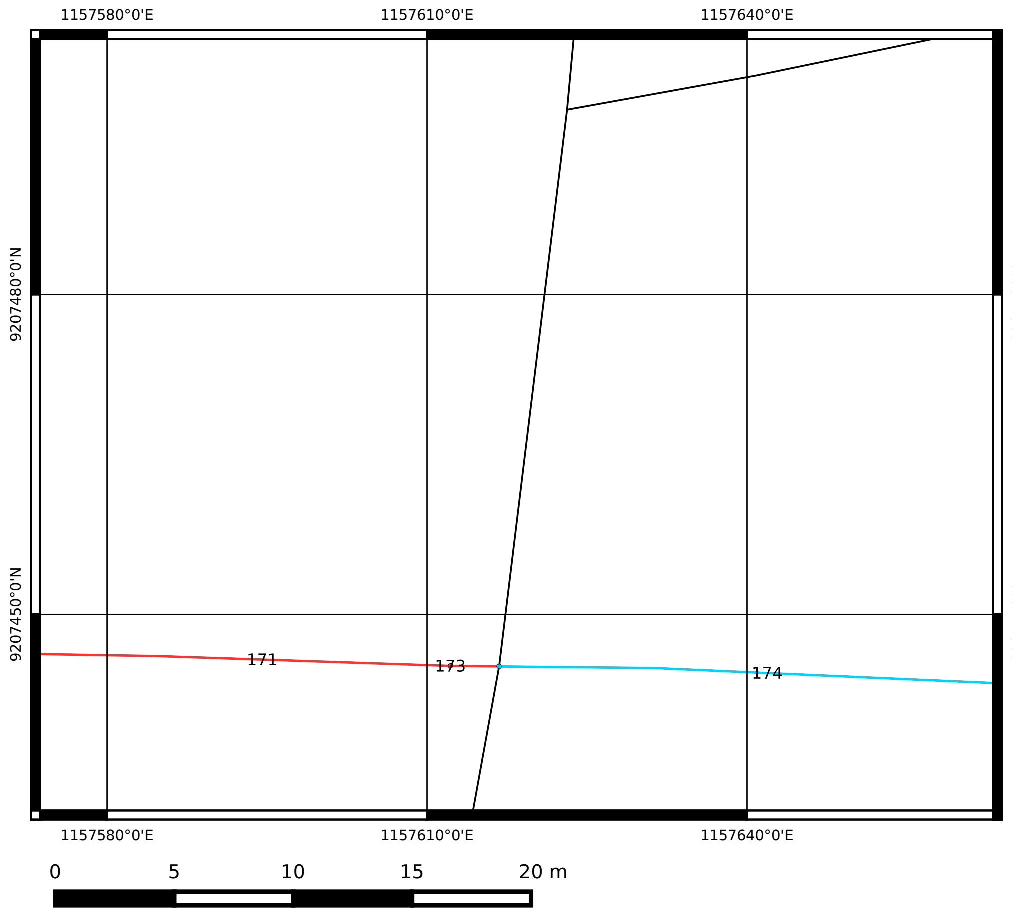


Details for the generated route 310179_KV_H_Midtbyen

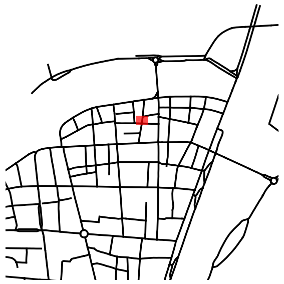


Legend

- Not Required
- Required
- Not used road

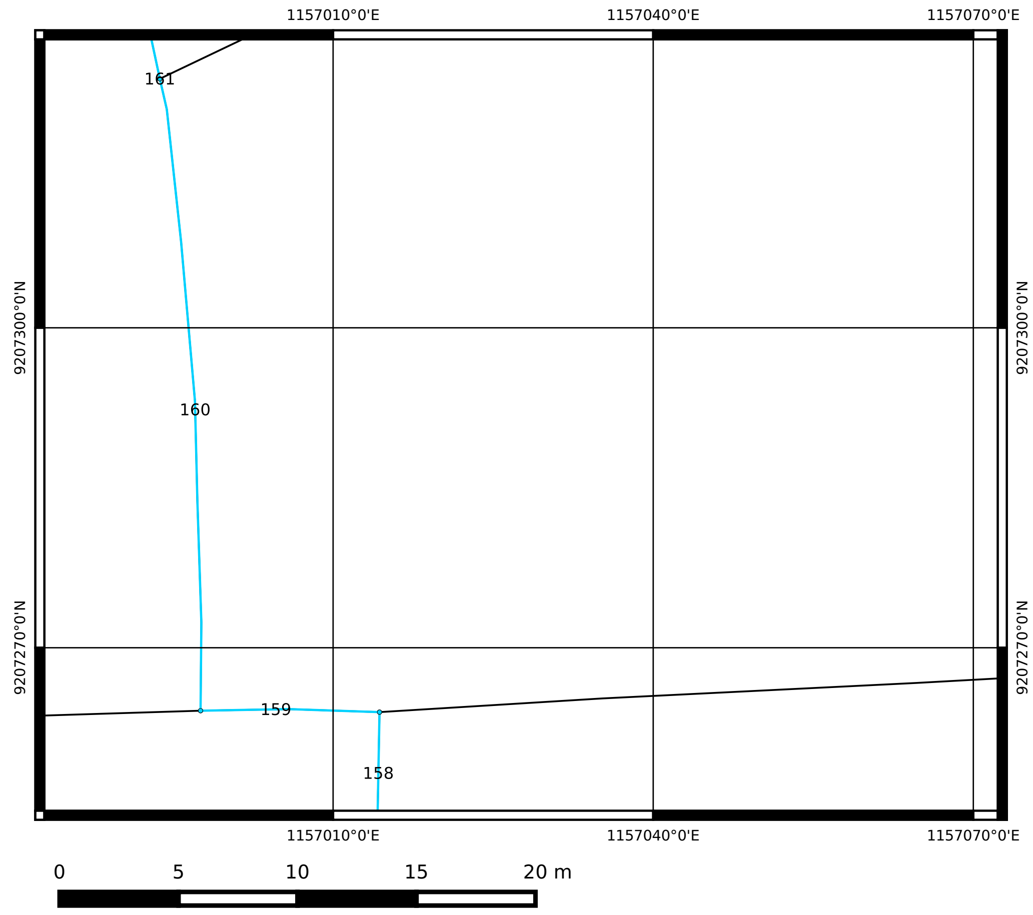


Details for the generated route 310179_KV_H_Midtbyen

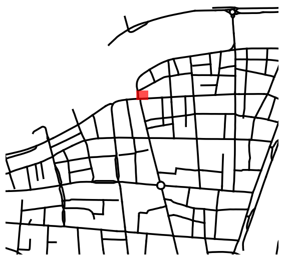


Legend

- 🟥 Not Required
- 🟦 Required
- — Not used road



Details for the generated route 310179_KV_H_Midtbyen

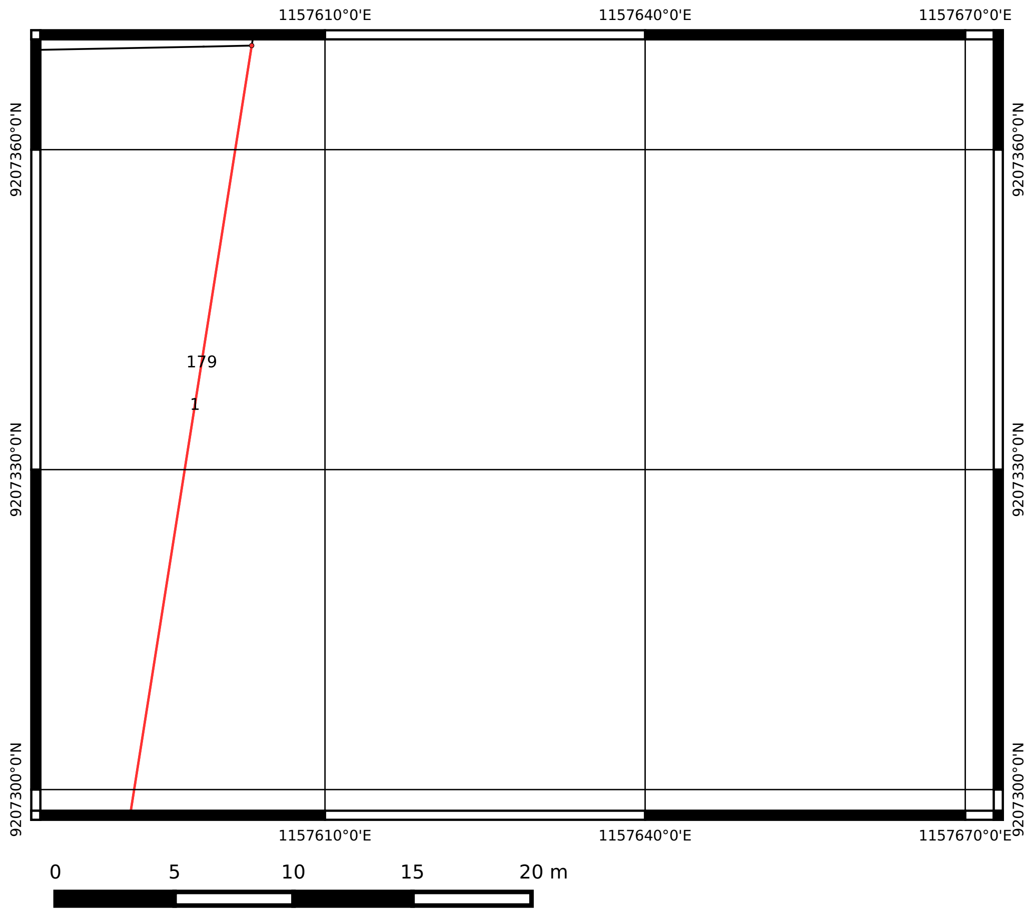


Legend

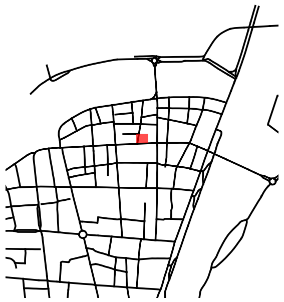
🟥 Not Required

🟦 Required

— Not used road

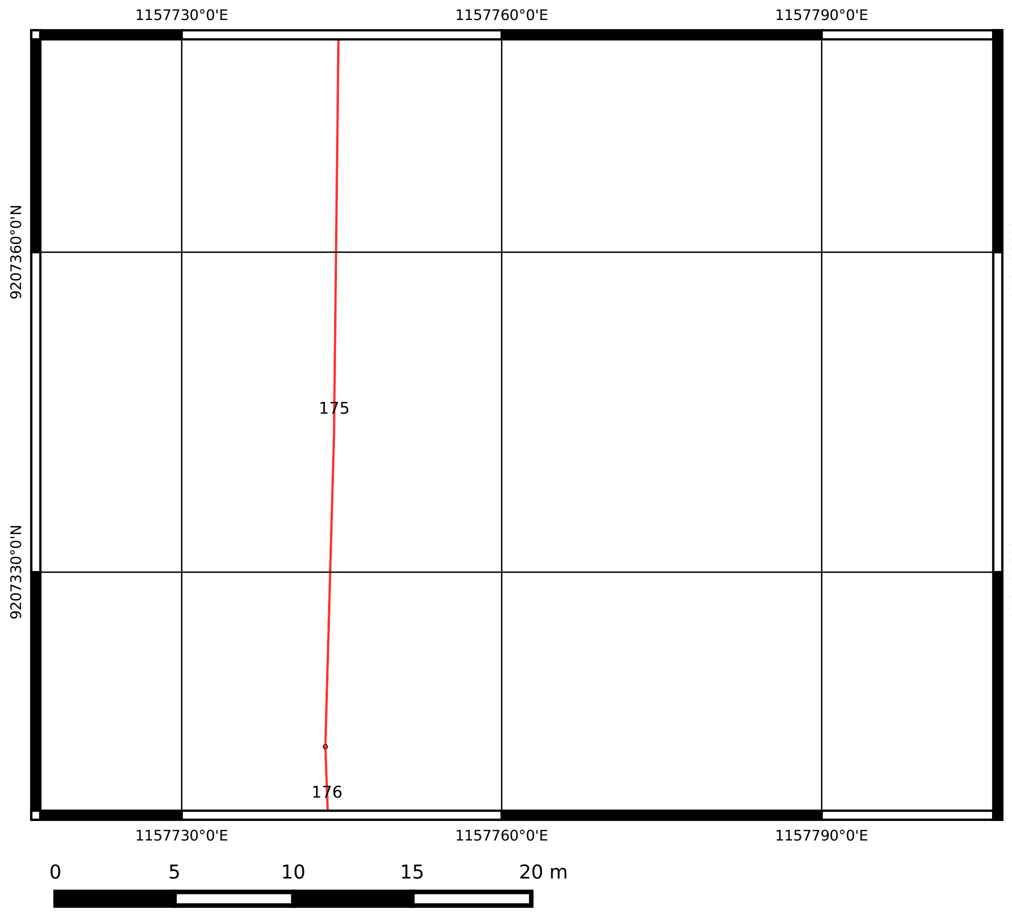


Details for the generated route 310179_KV_H_Midtbyen

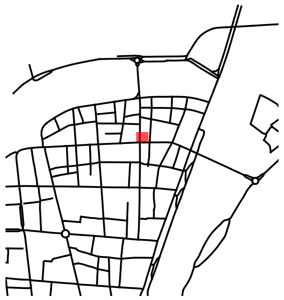


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310179_KV_H_Midtbyen

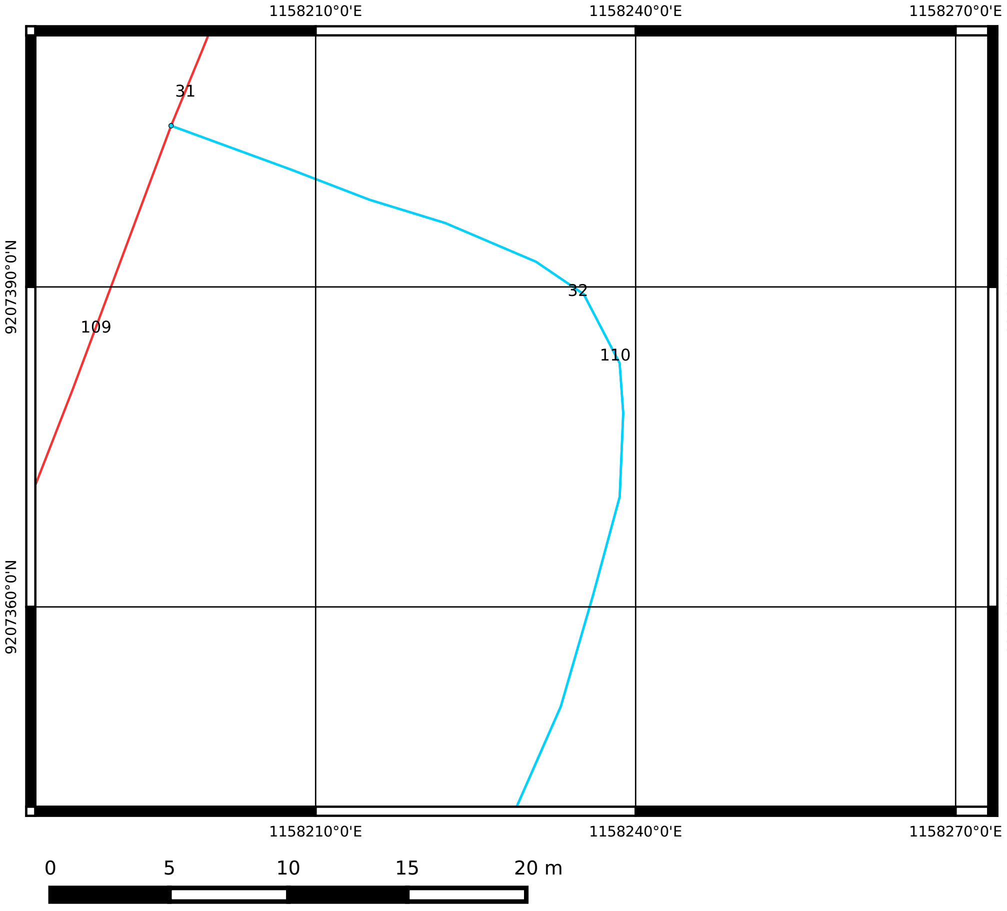


Legend

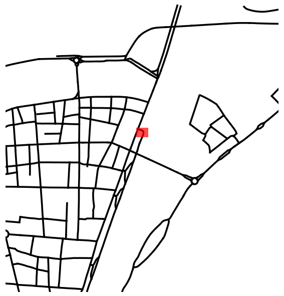
🟥 Not Required

🟦 Required

— Not used road

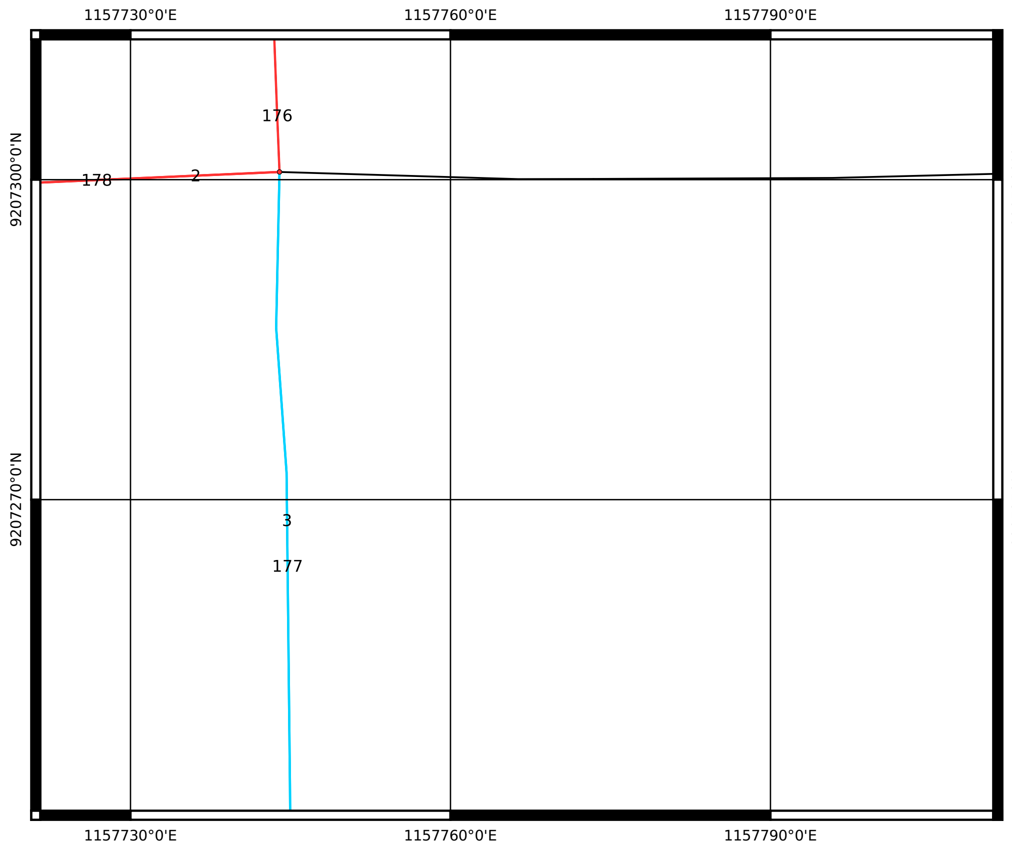


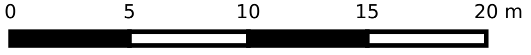
Details for the generated route 310179_KV_H_Midtbyen



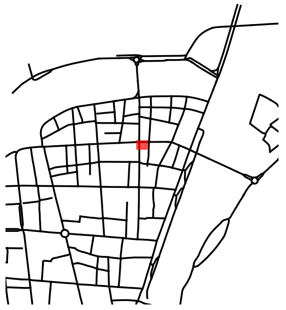
Legend

🟥	Not Required
🟦	Required
—	Not used road



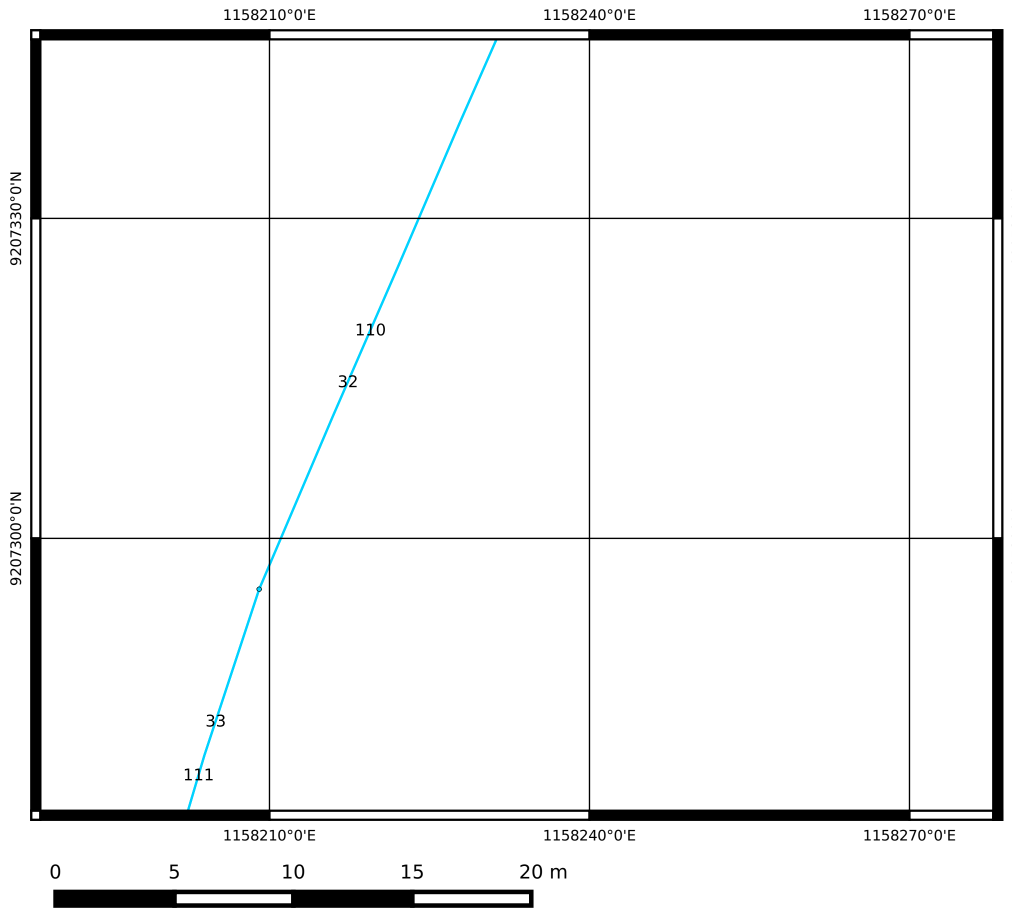


Details for the generated route 310179_KV_H_Midtbyen

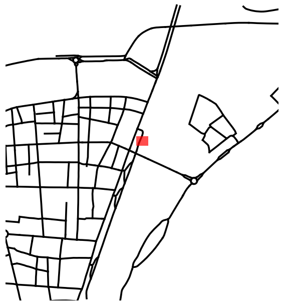


Legend

- Not Required
- Required
- Not used road

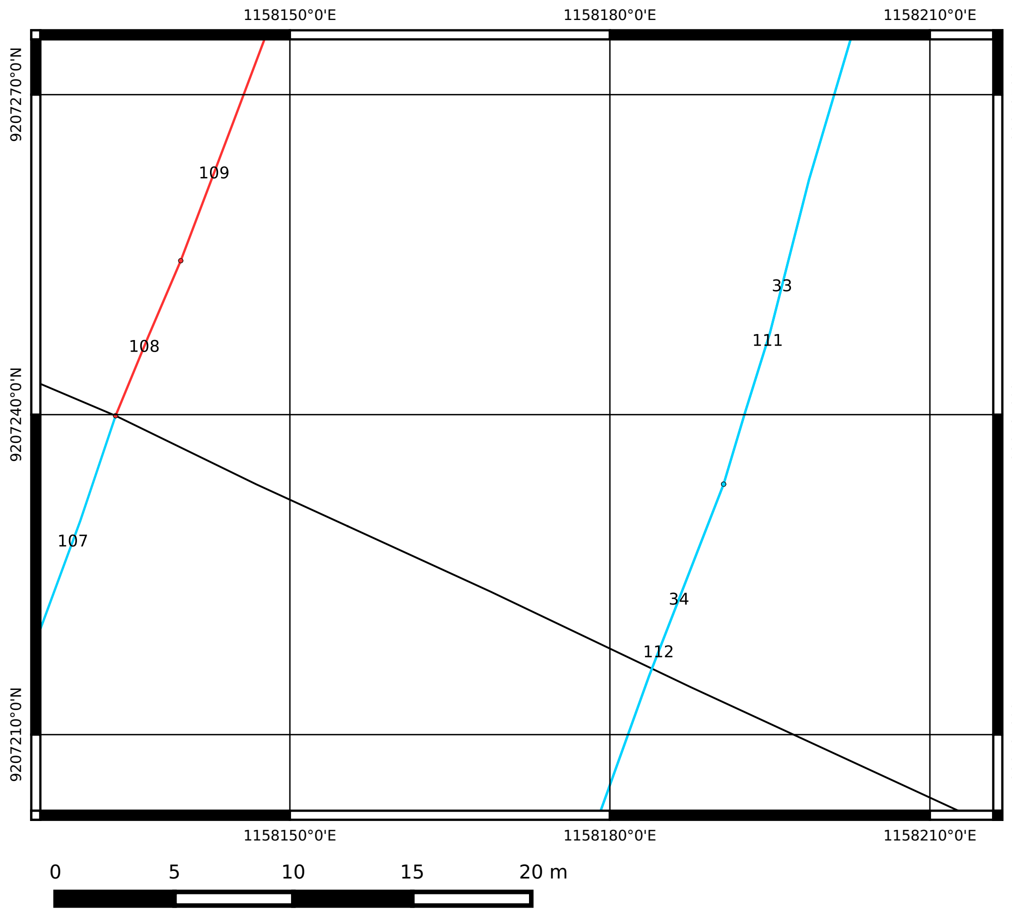


Details for the generated route 310179_KV_H_Midtbyen

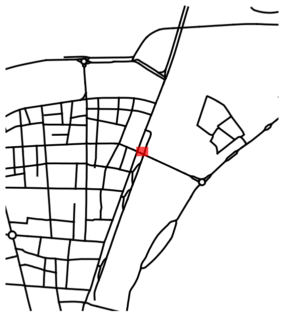


Legend

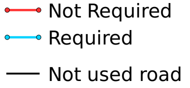
🟥	Not Required
🟦	Required
—	Not used road

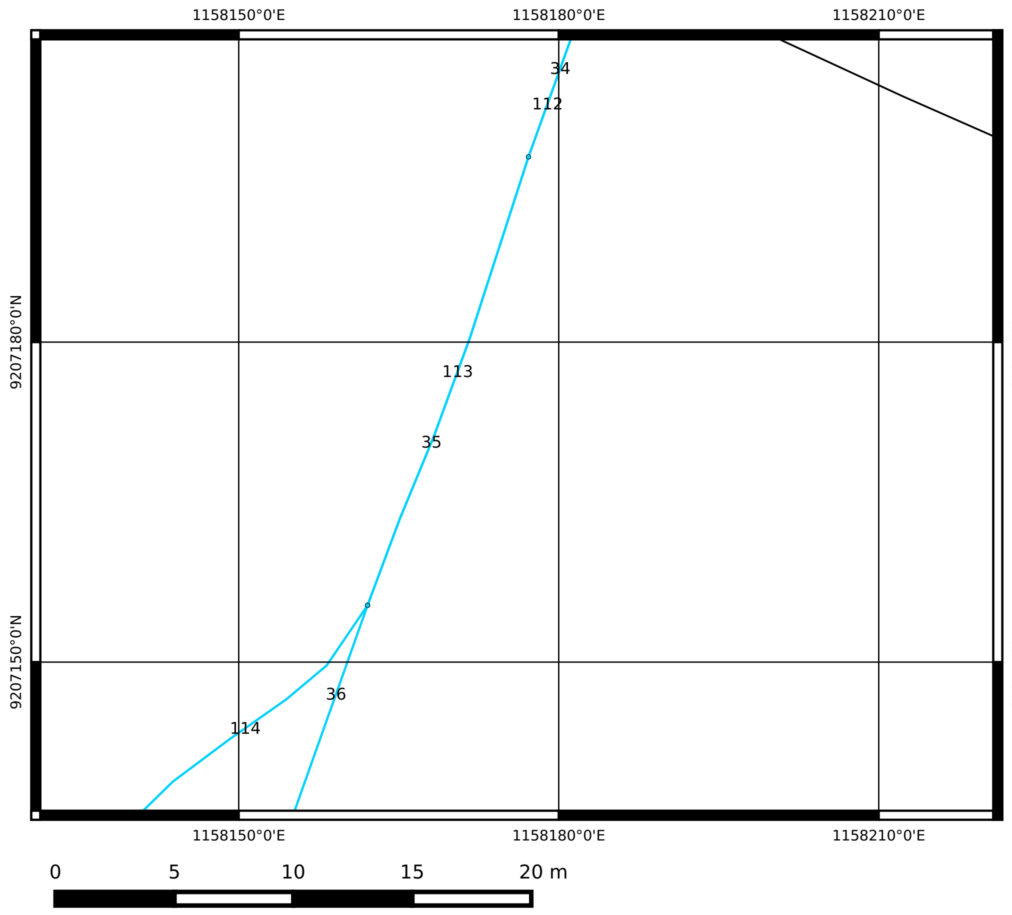


Details for the generated route 310179_KV_H_Midtbyen

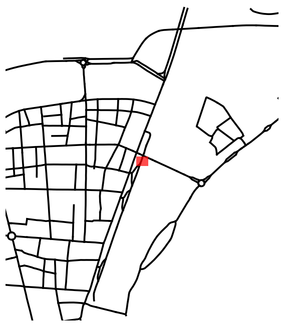


Legend





Details for the generated route 310179_KV_H_Midtbyen

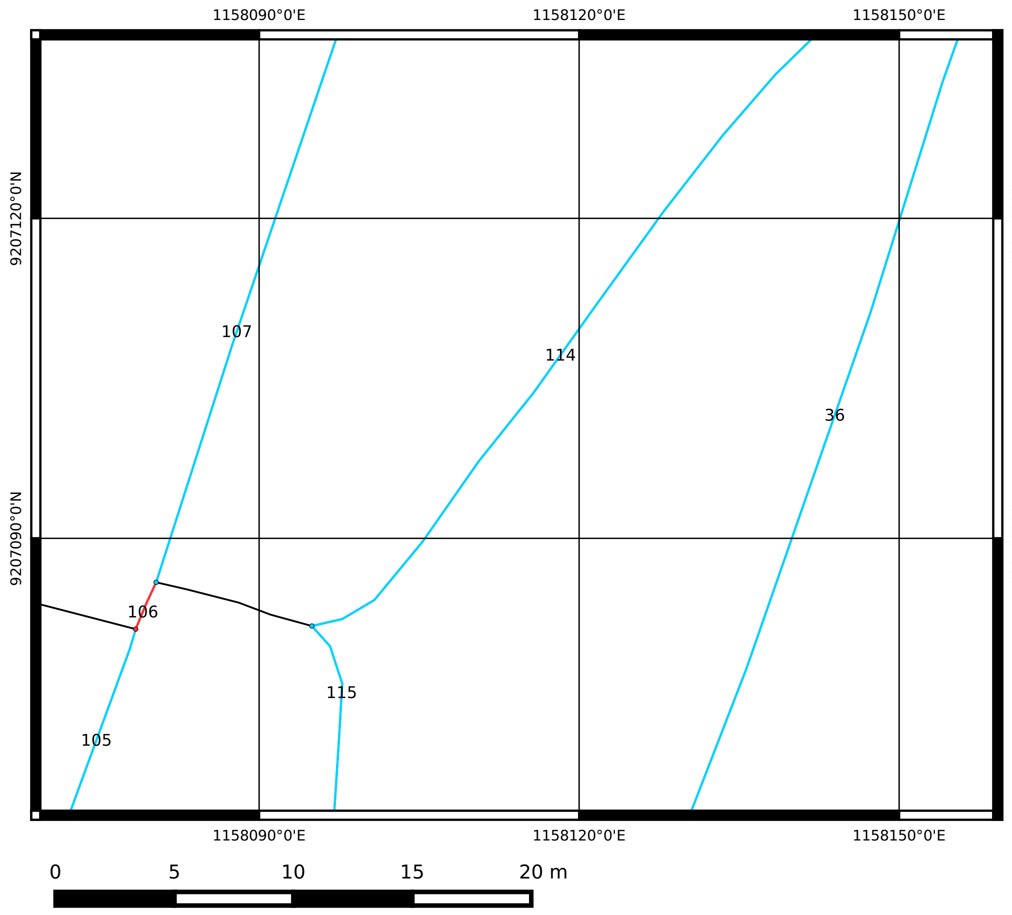


Legend

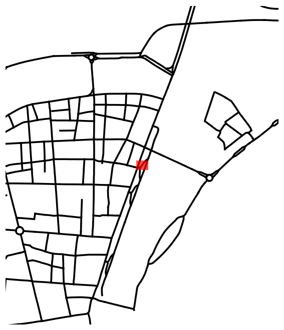
Not Required

Required

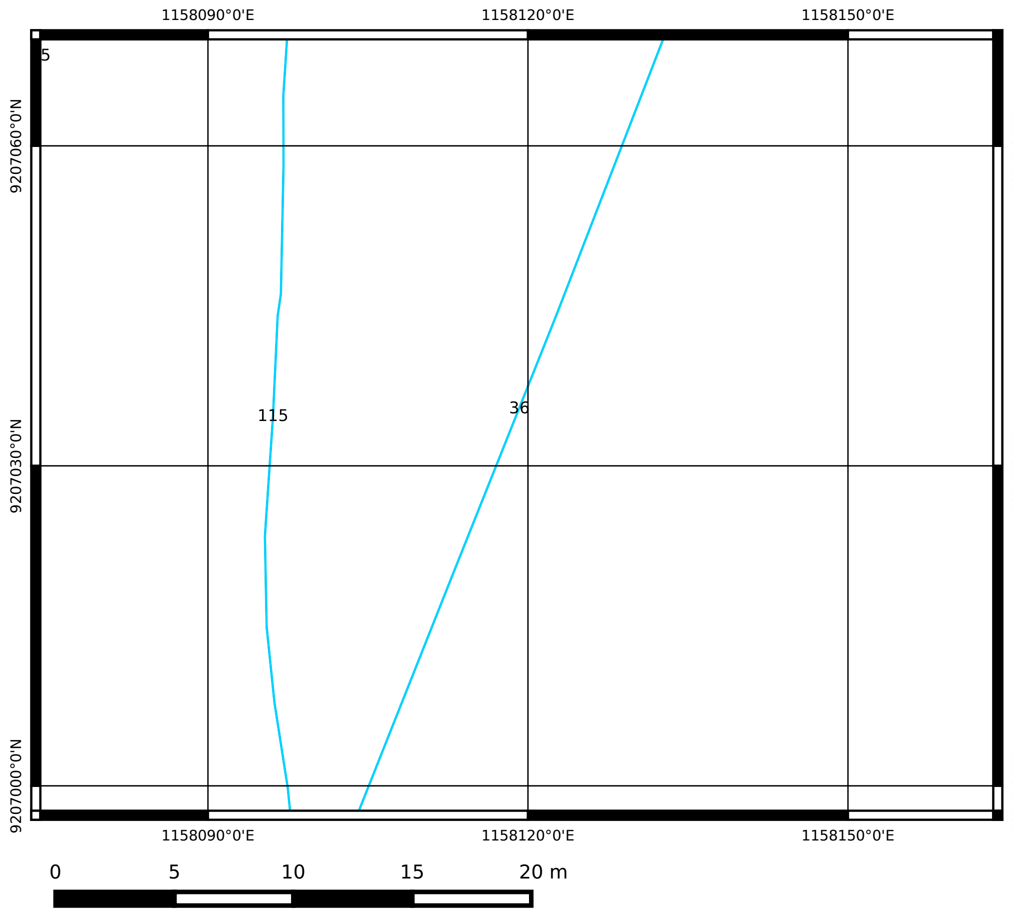
Not used road



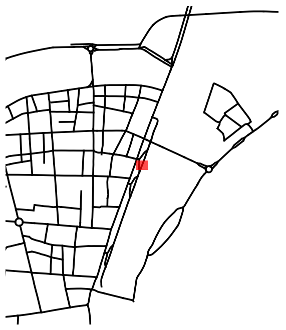
Details for the generated route 310179_KV_H_Midtbyen



Legend

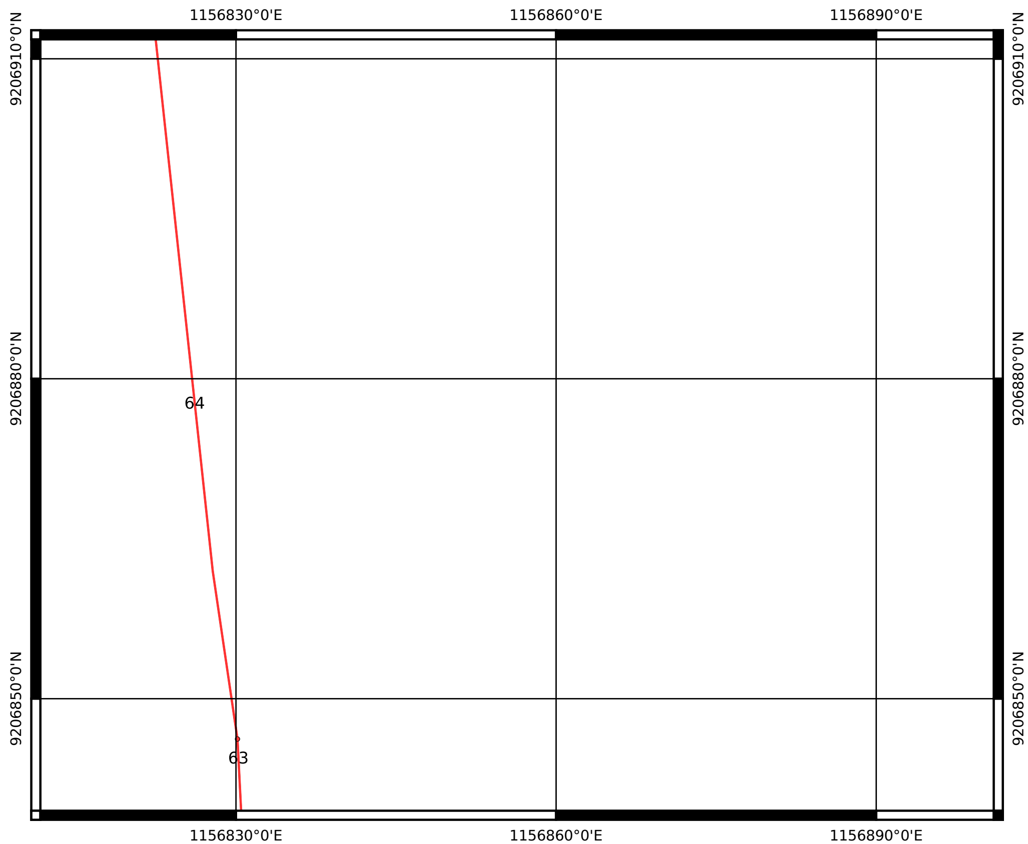


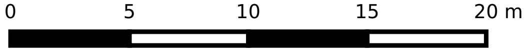
Details for the generated route 310179_KV_H_Midtbyen



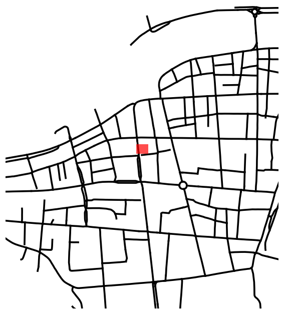
Legend

- 🟥 Not Required
- 🟦 Required
- — Not used road





Details for the generated route 310179_KV_H_Midtbyen

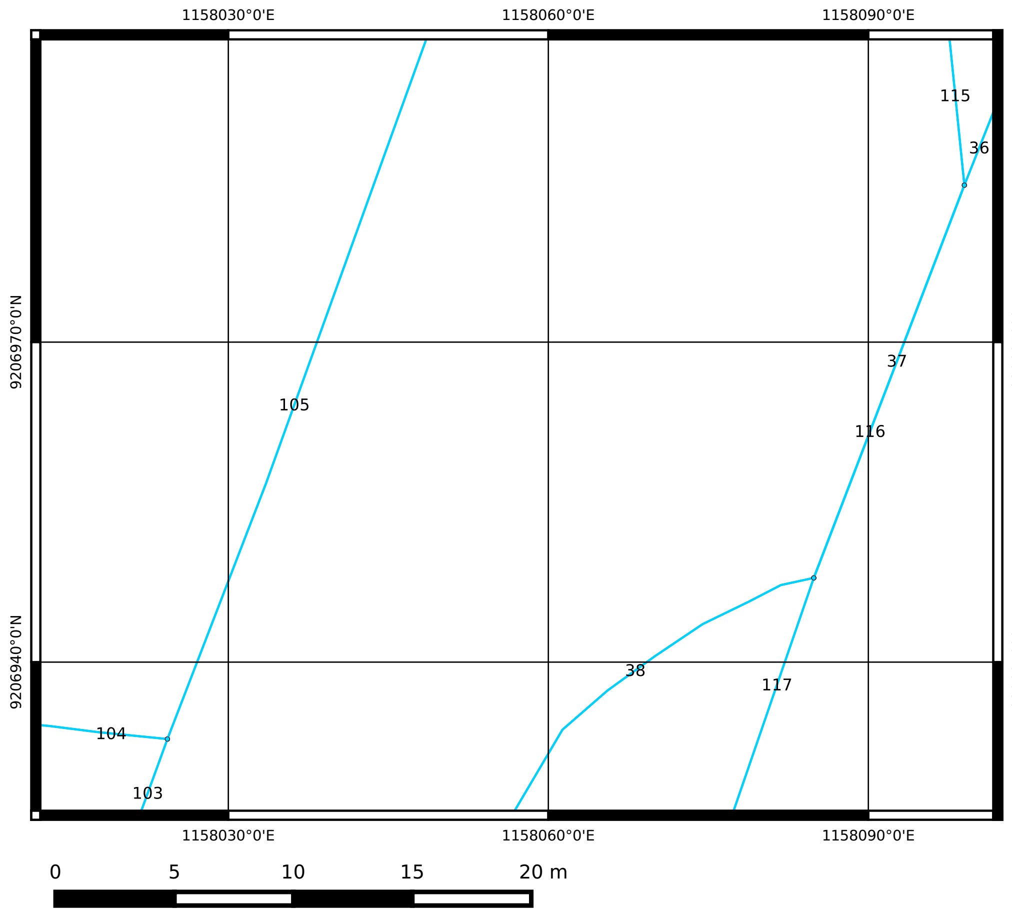


Legend

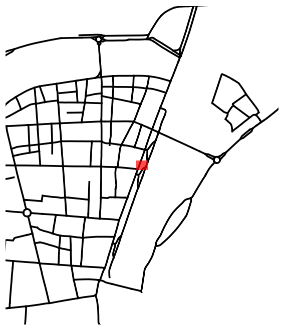
🟥 Not Required

🟦 Required

— Not used road

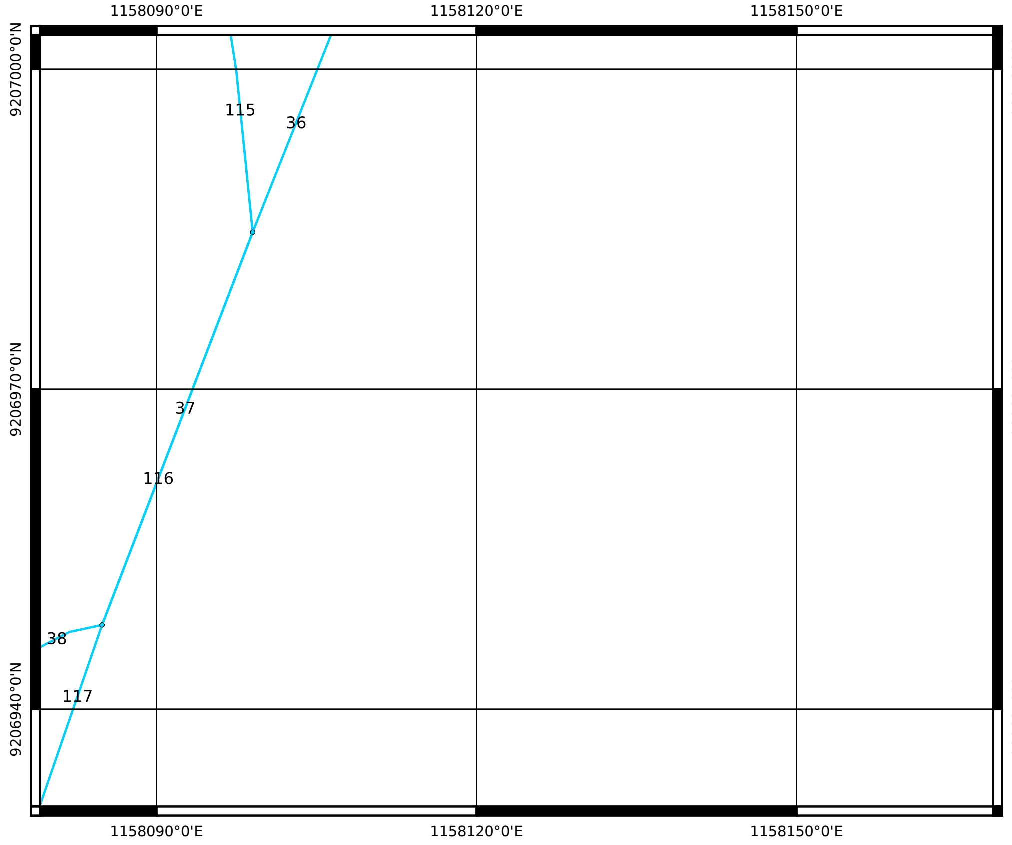


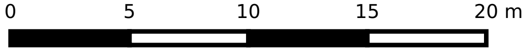
Details for the generated route 310179_KV_H_Midtbyen



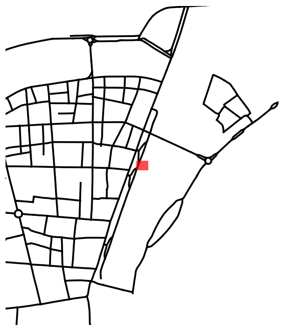
Legend

- Not Required
- Required
- Not used road



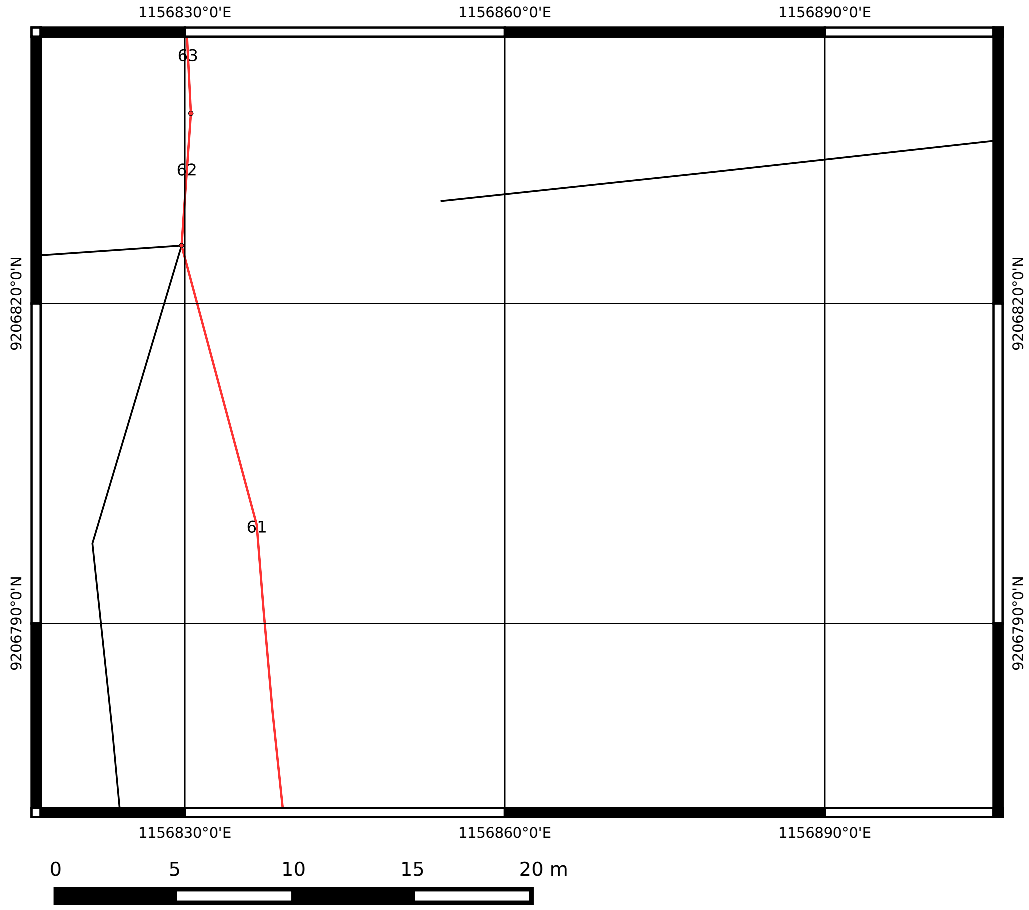


Details for the generated route 310179_KV_H_Midtbyen

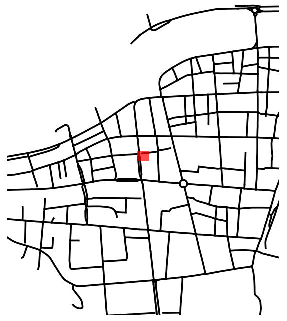


Legend

- Not Required
- Required
- Not used road

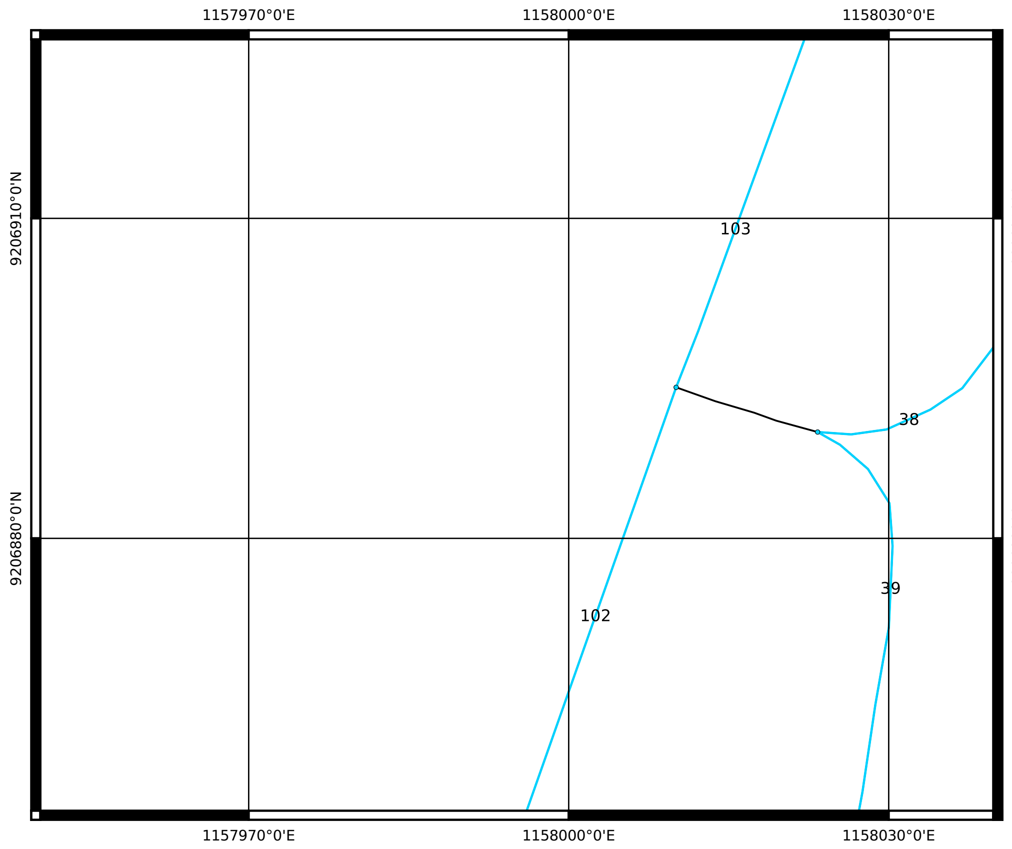


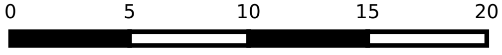
Details for the generated route 310179_KV_H_Midtbyen



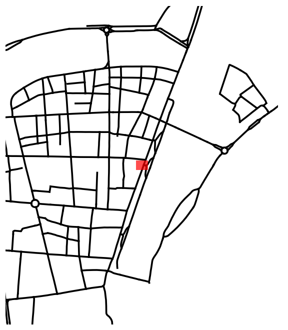
Legend

🟥 Not Required
🟦 Required
— Not used road





Details for the generated route 310179_KV_H_Midtbyen

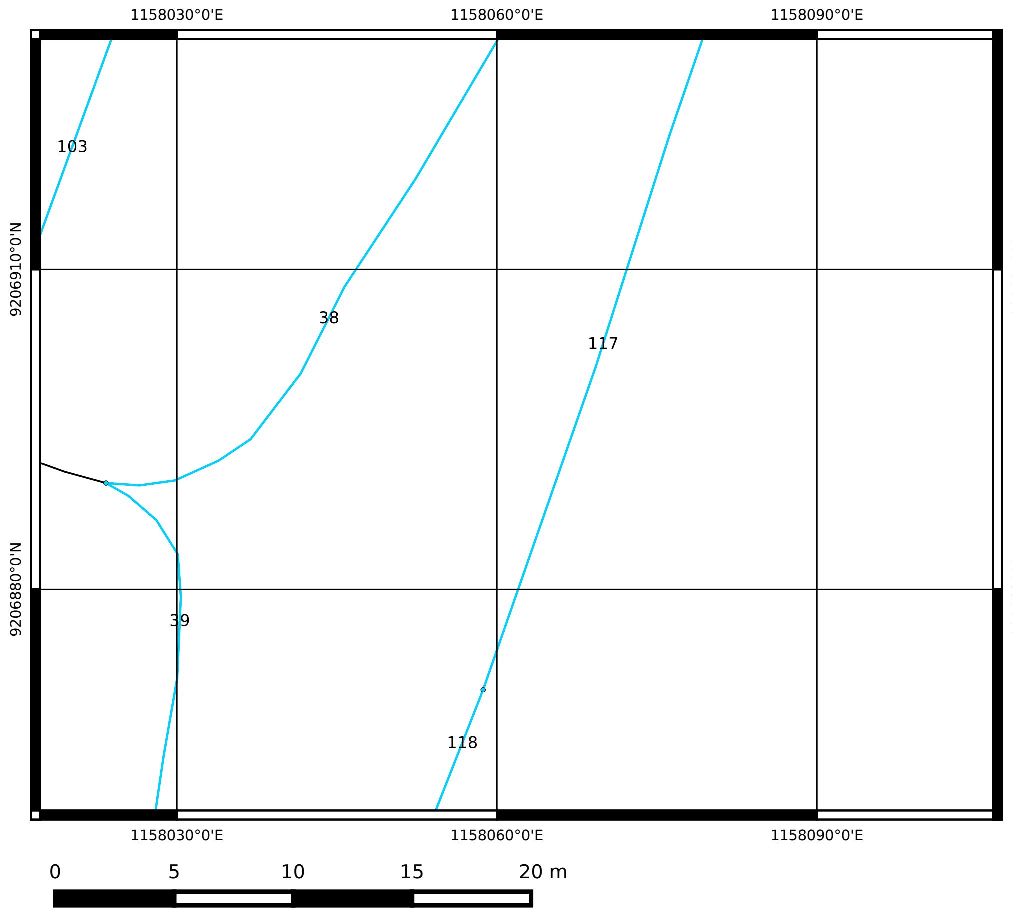


Legend

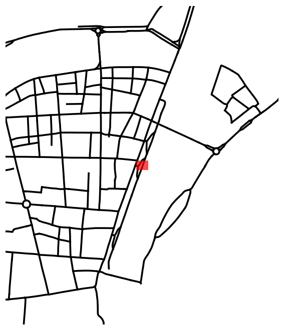
🟥 Not Required

🟦 Required

— Not used road



Details for the generated route 310179_KV_H_Midtbyen

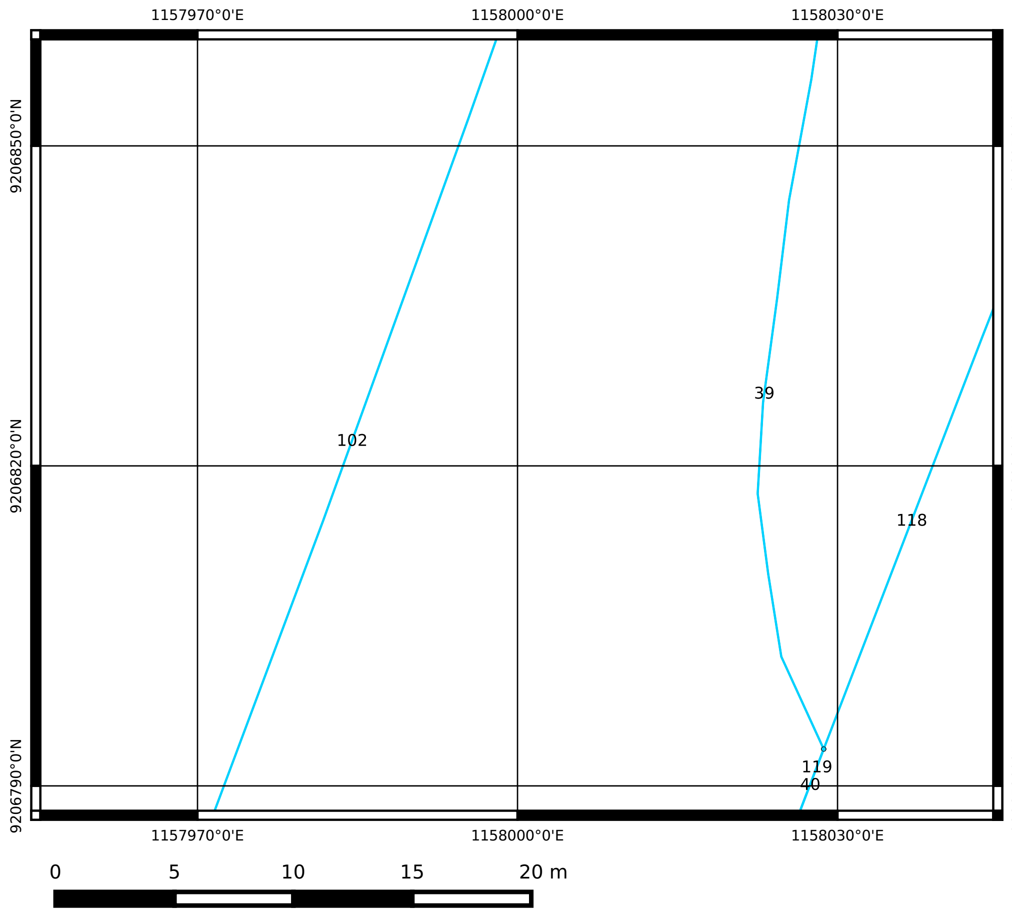


Legend

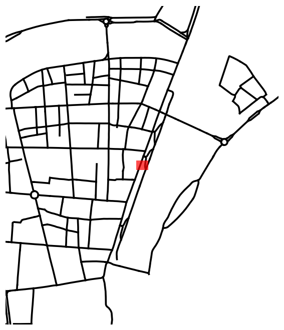
🟥 Not Required

🟦 Required

— Not used road



Details for the generated route 310179_KV_H_Midtbyen

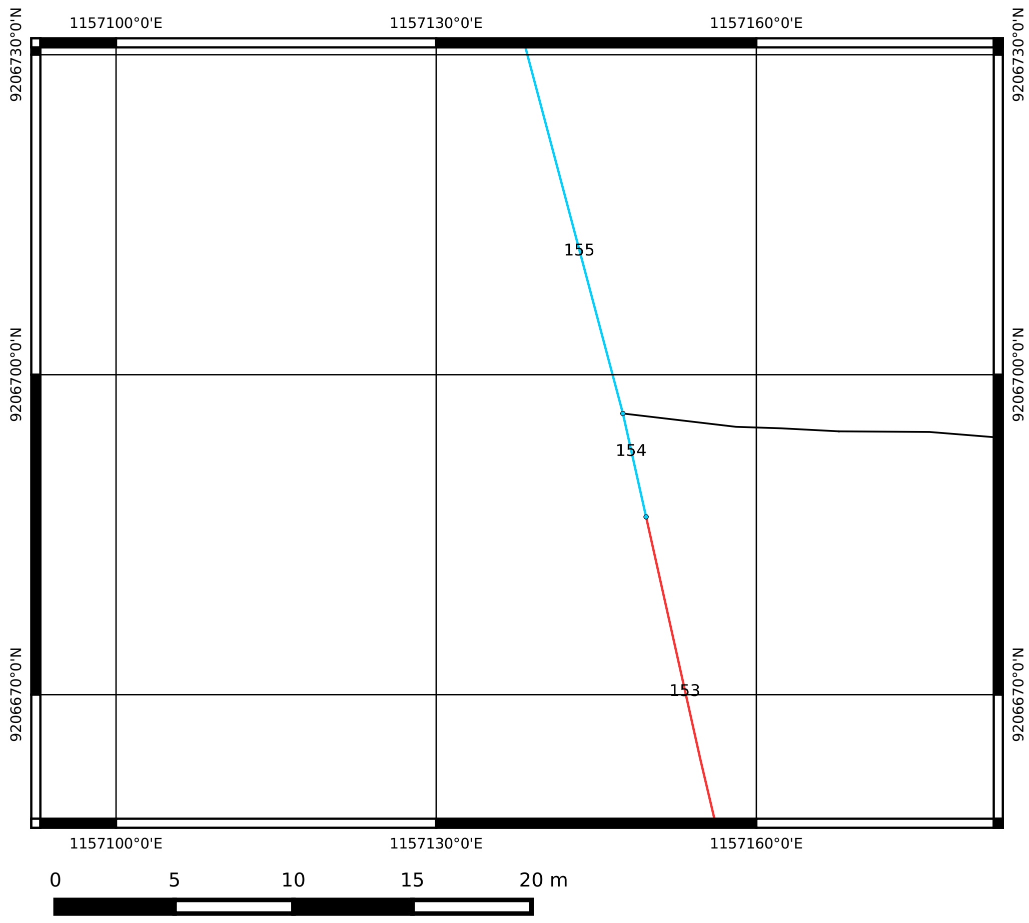


Legend

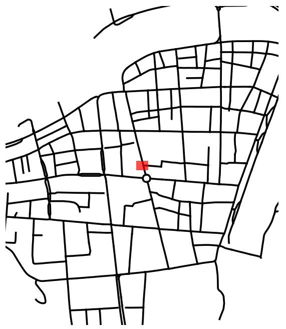
🟥 Not Required

🟦 Required

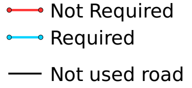
— Not used road

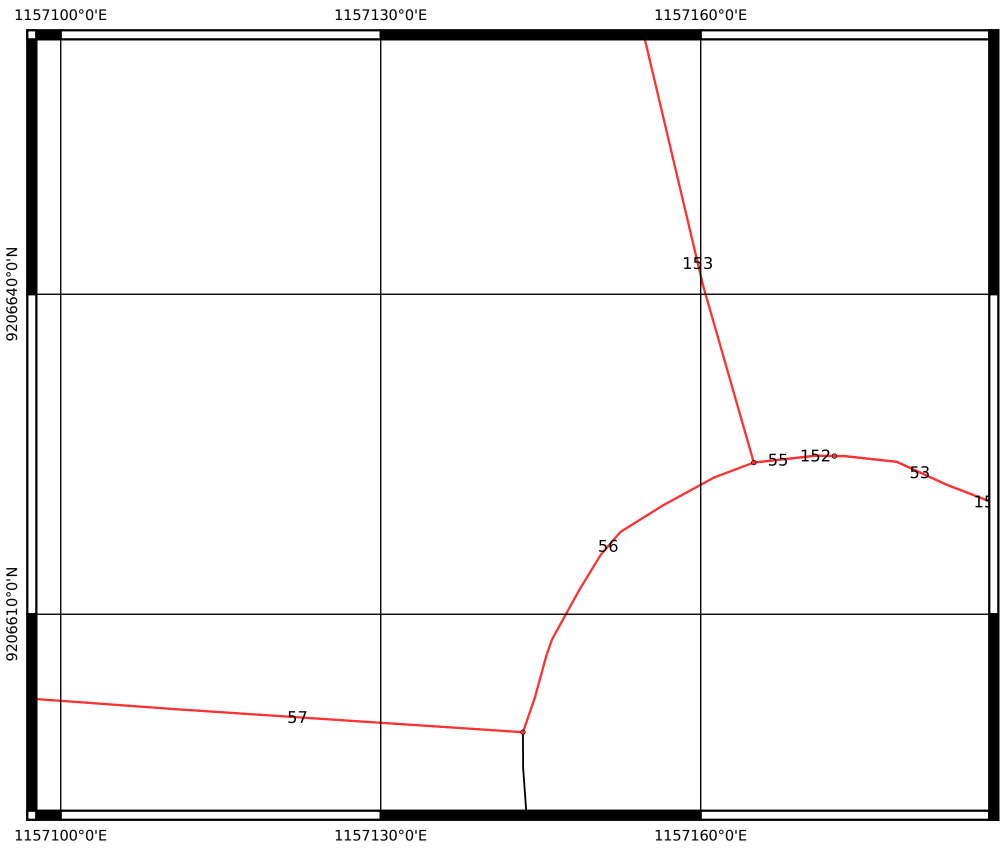


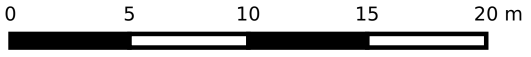
Details for the generated route 310179_KV_H_Midtbyen



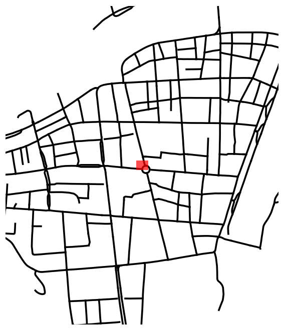
Legend





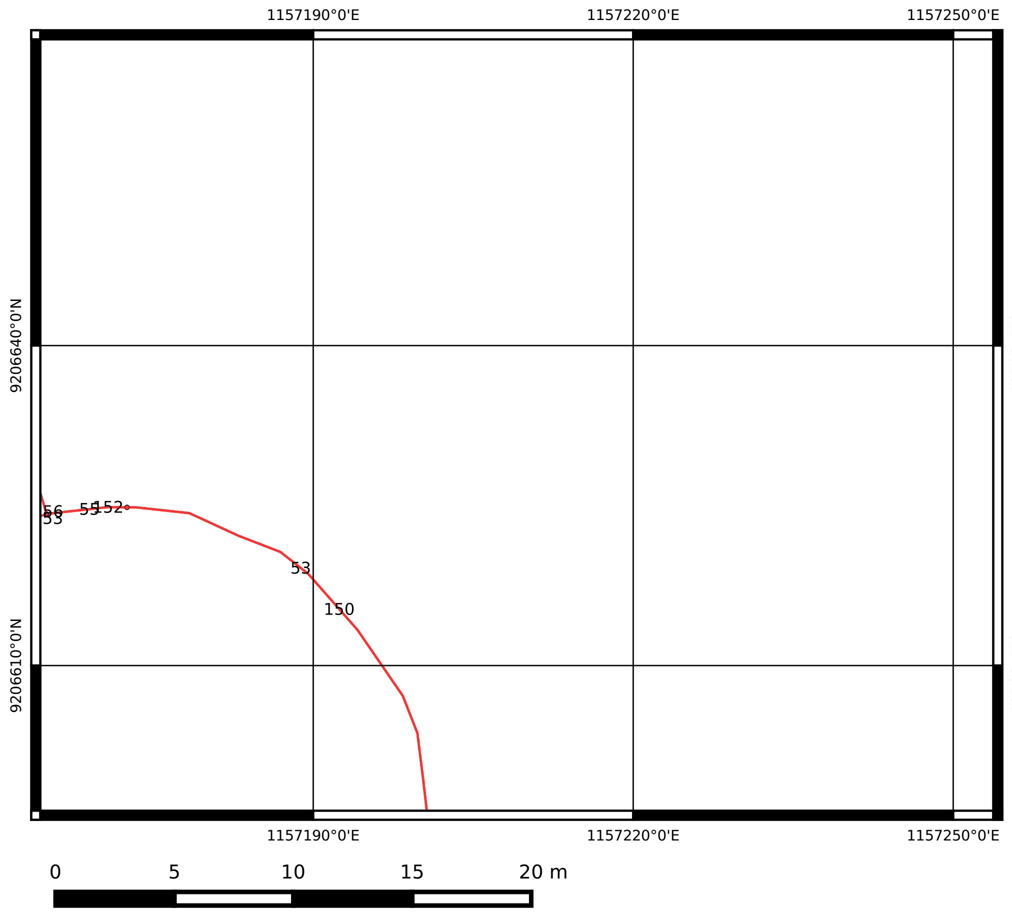


Details for the generated route 310179_KV_H_Midtbyen

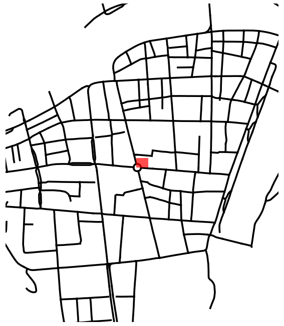


Legend

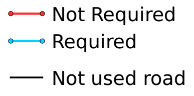
🟥	Not Required
🟦	Required
—	Not used road

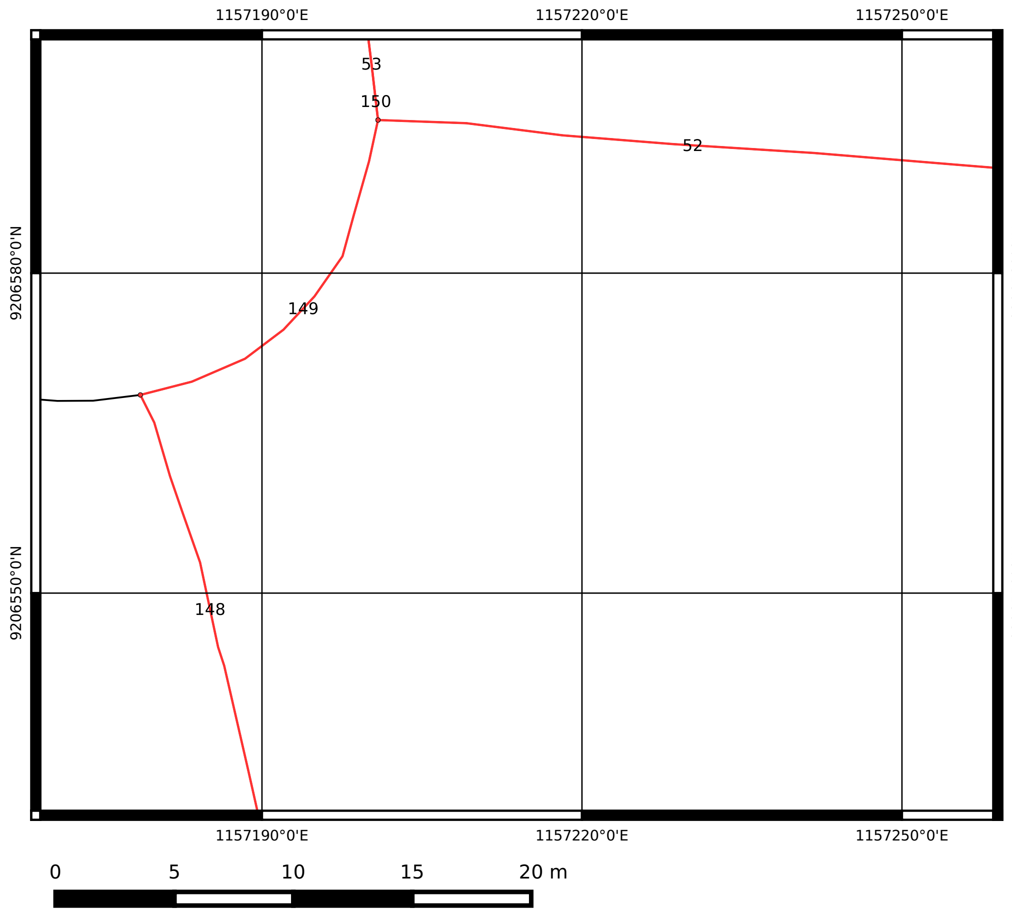


Details for the generated route 310179_KV_H_Midtbyen

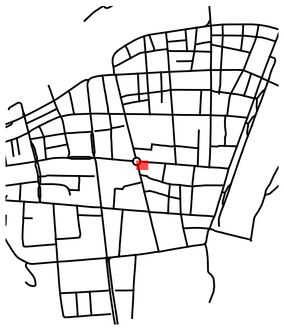


Legend



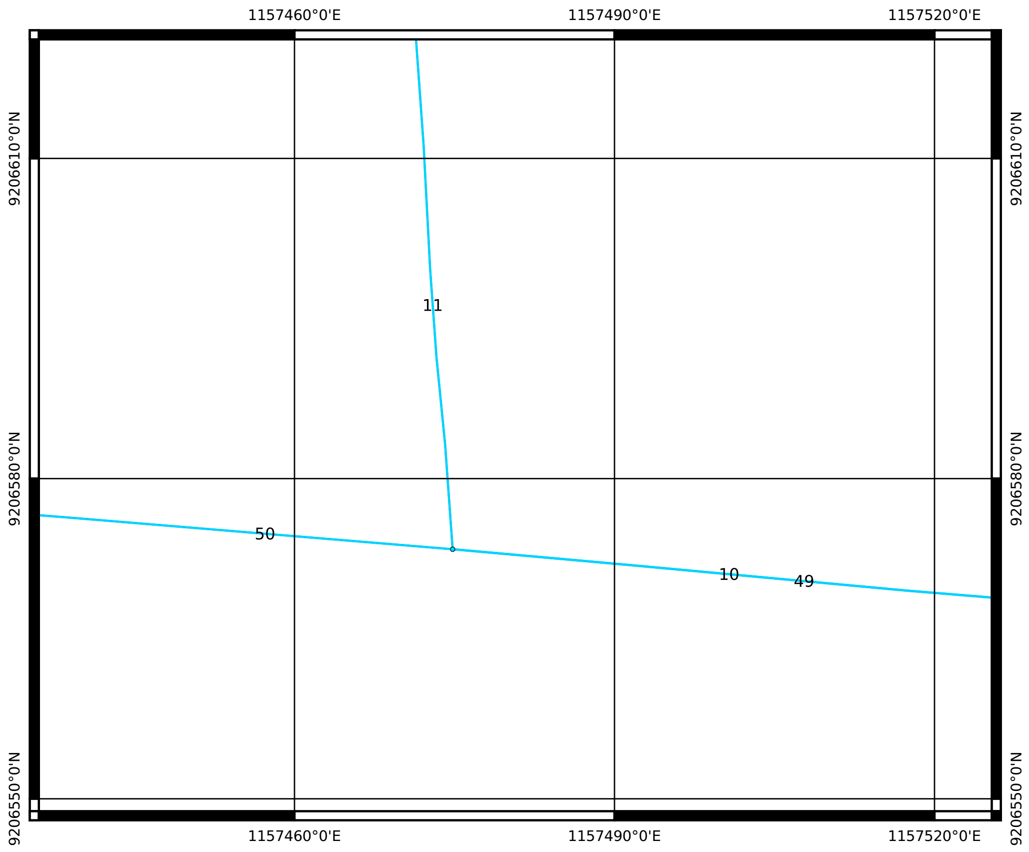


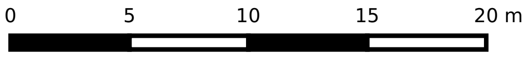
Details for the generated route 310179_KV_H_Midtbyen



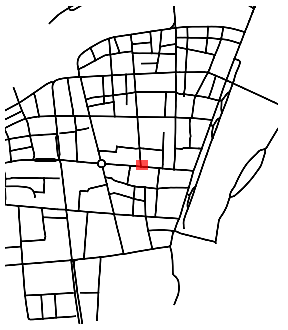
Legend

🟥	Not Required
🟦	Required
—	Not used road



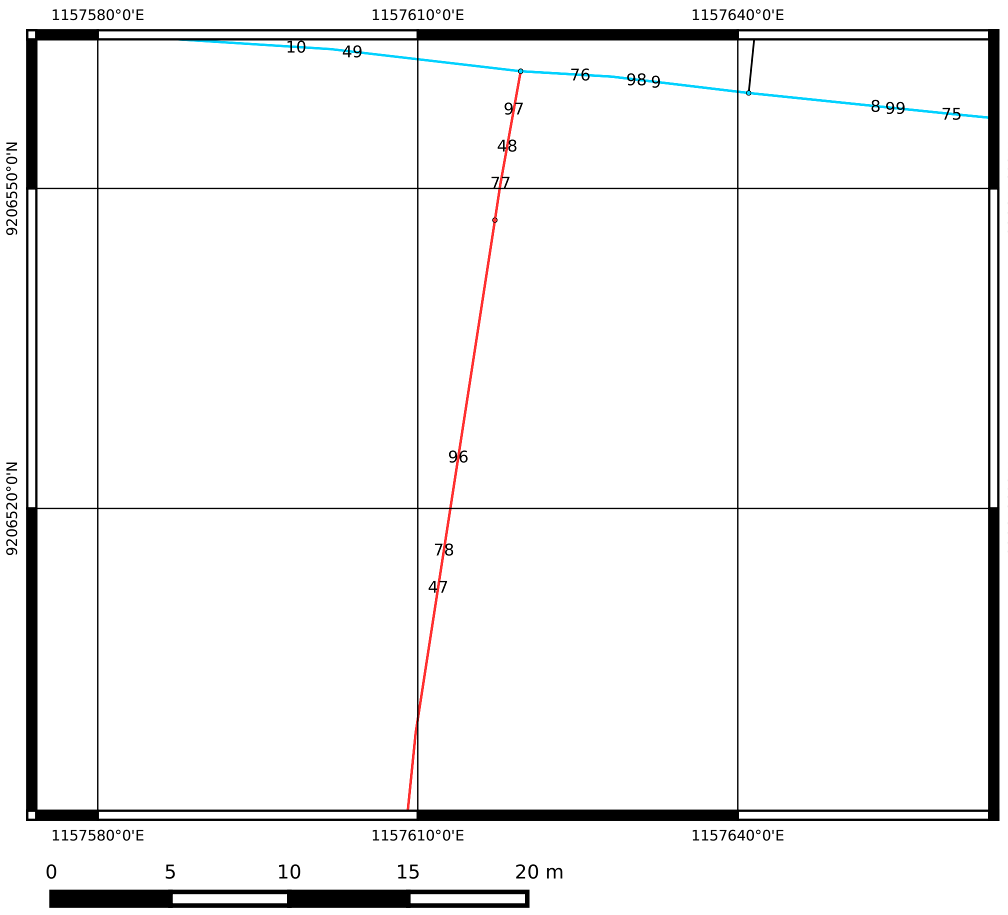


Details for the generated route 310179_KV_H_Midtbyen

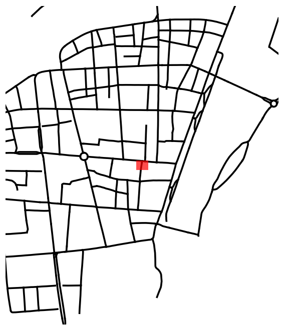


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310179_KV_H_Midtbyen



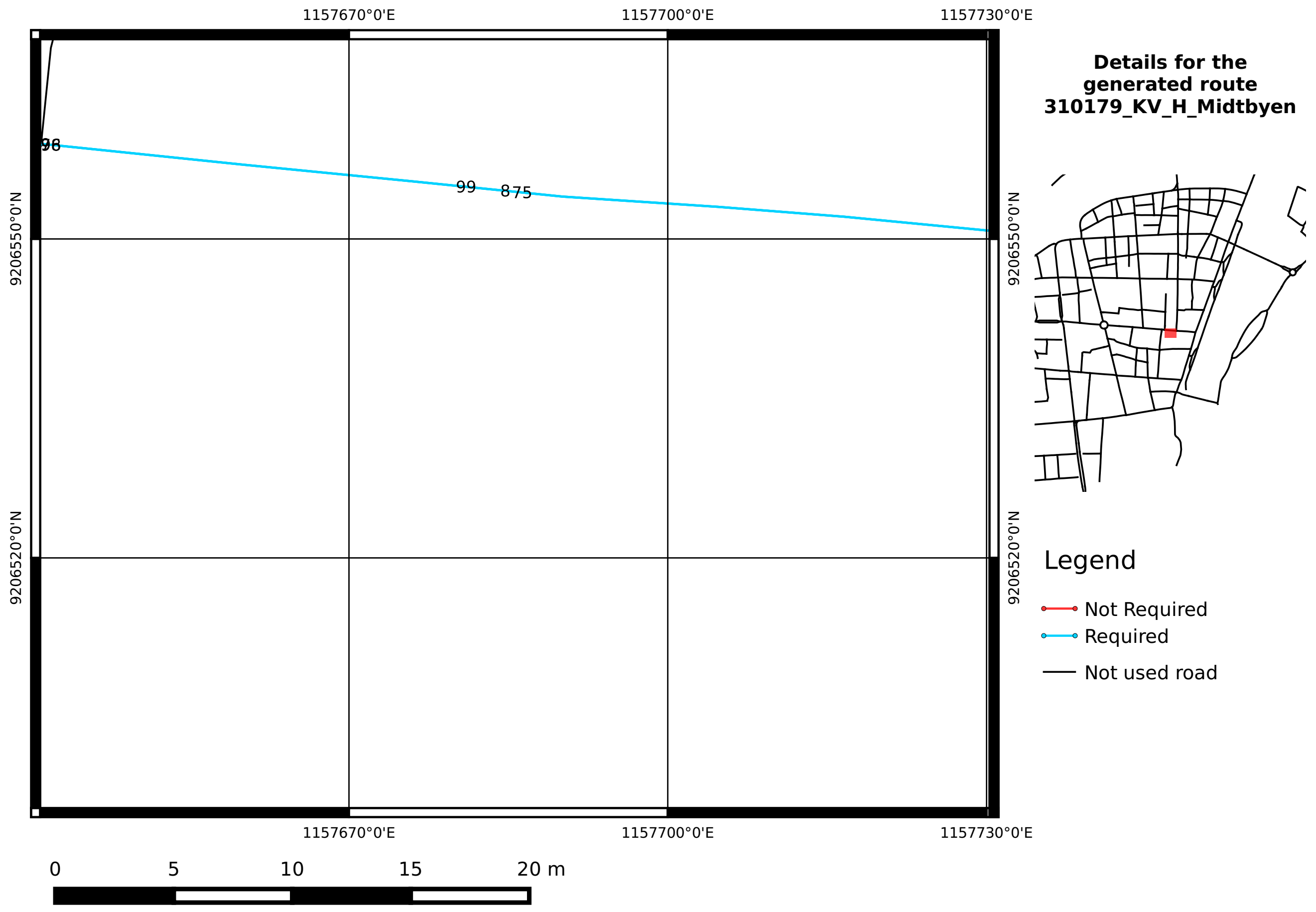
Legend

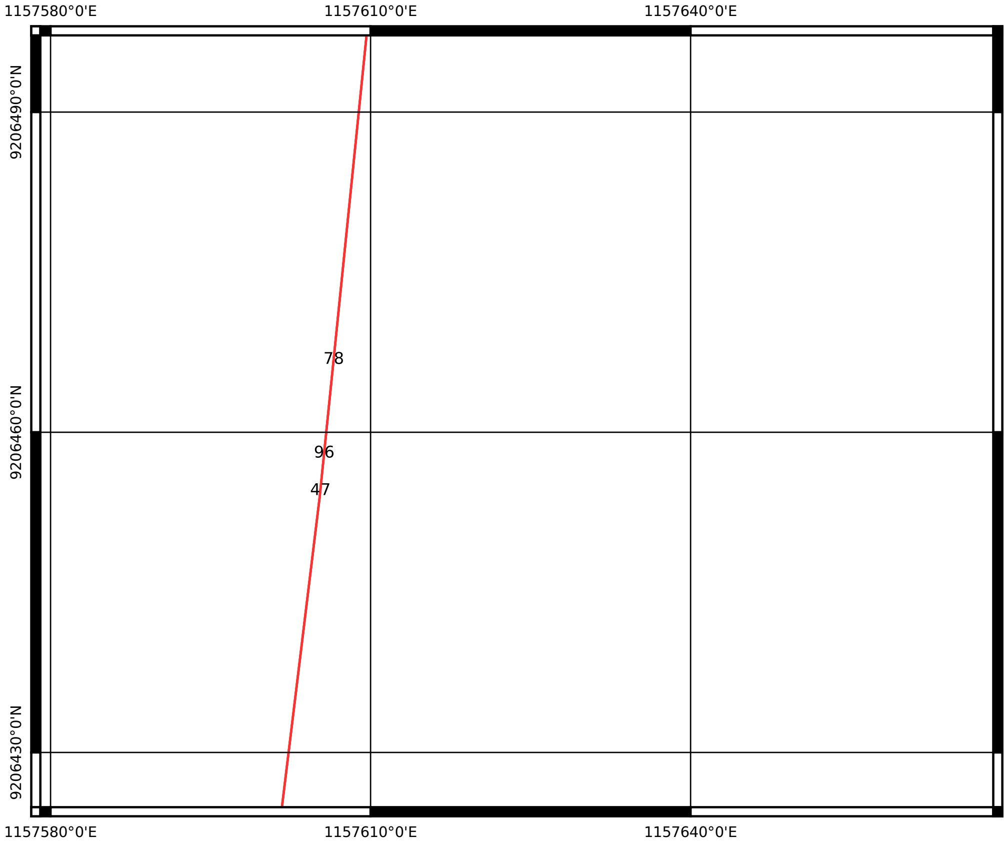
🟥 Not Required

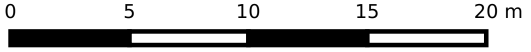
🟦 Required

— Not used road

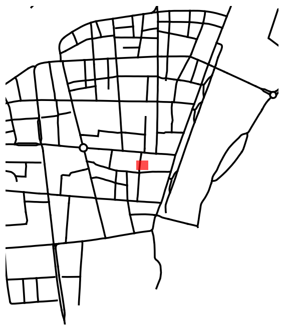
Details for the generated route 310179_KV_H_Midtbyen





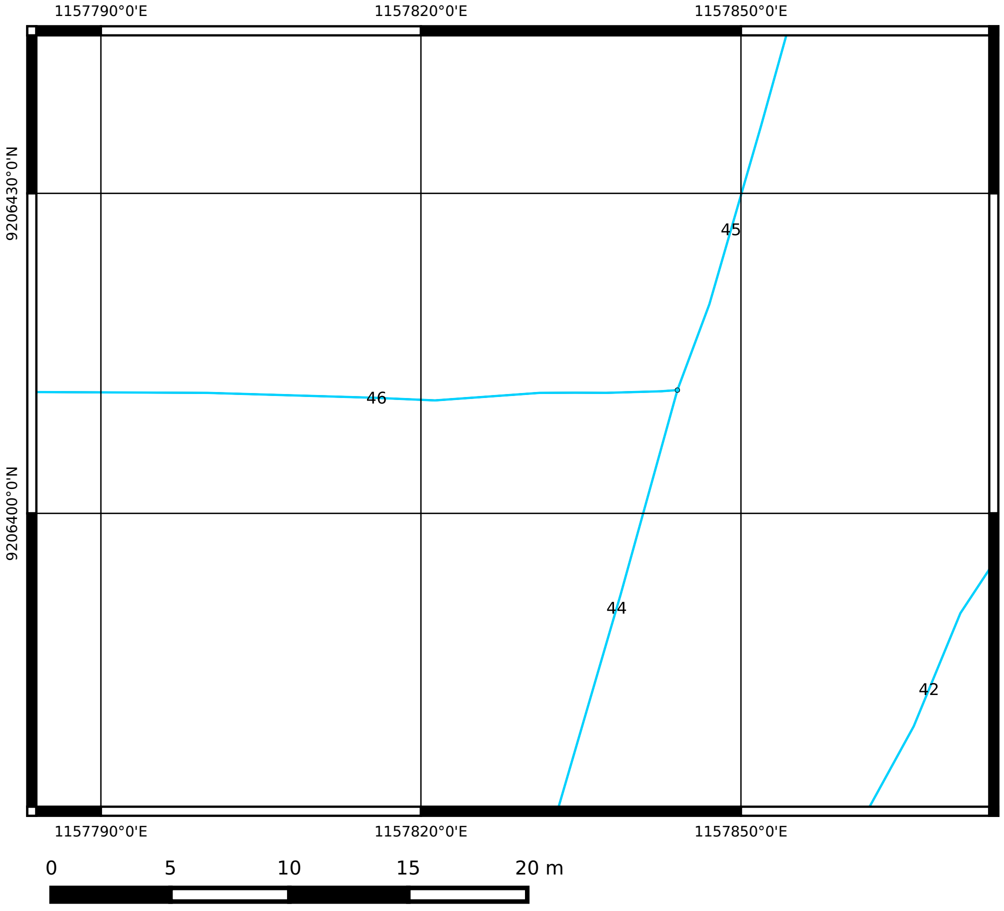


Details for the generated route 310179_KV_H_Midtbyen

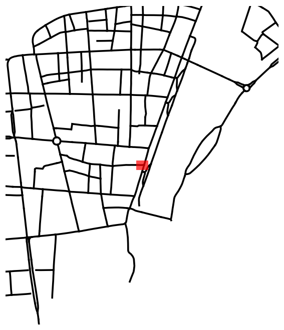


Legend

🟥	Not Required
🟦	Required
—	Not used road

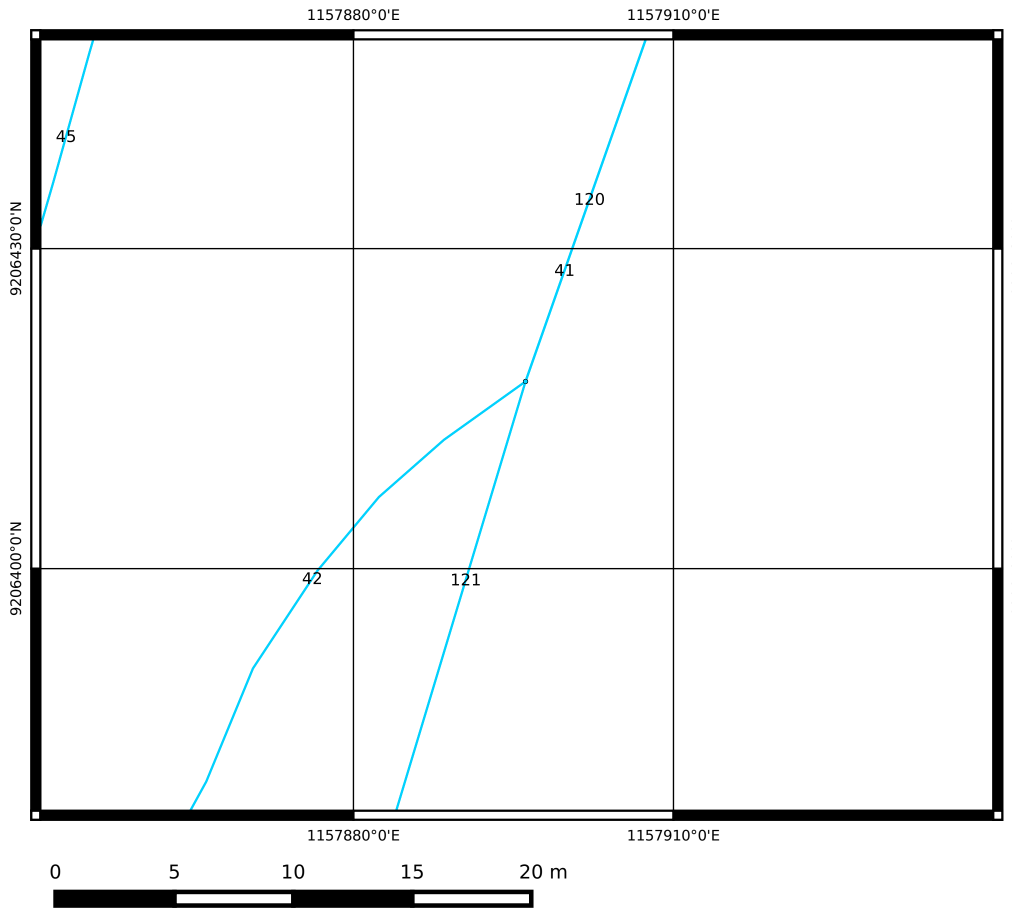


Details for the generated route 310179_KV_H_Midtbyen

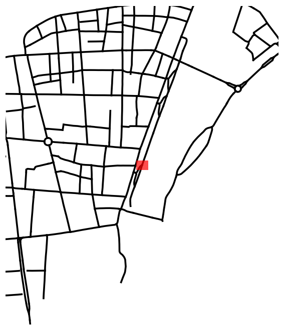


Legend

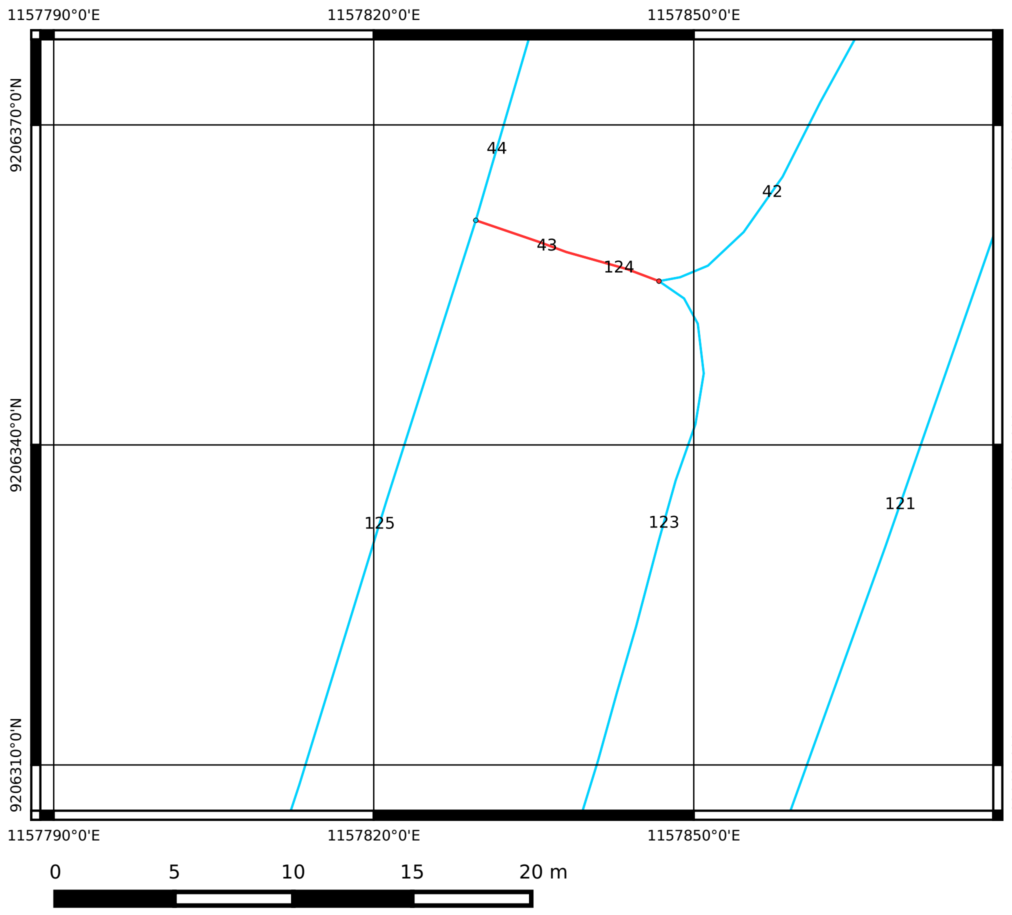
🟥	Not Required
🟦	Required
—	Not used road



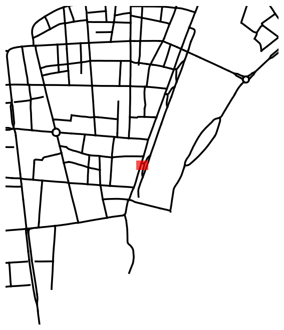
Details for the generated route 310179_KV_H_Midtbyen



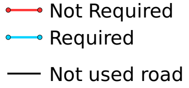
Legend

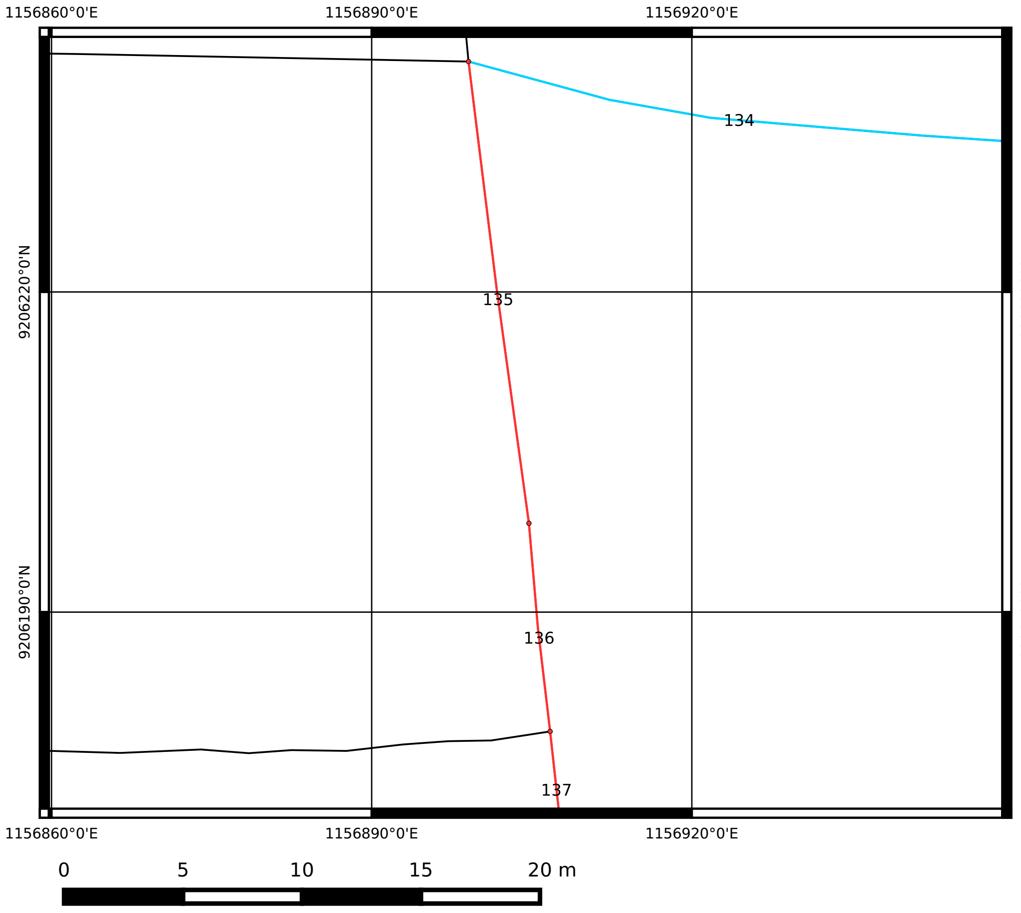


Details for the generated route 310179_KV_H_Midtbyen

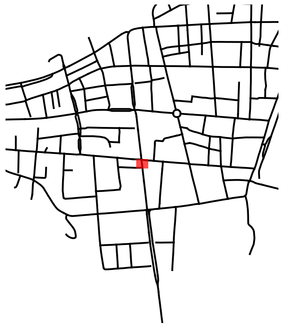


Legend

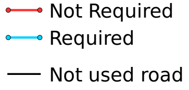


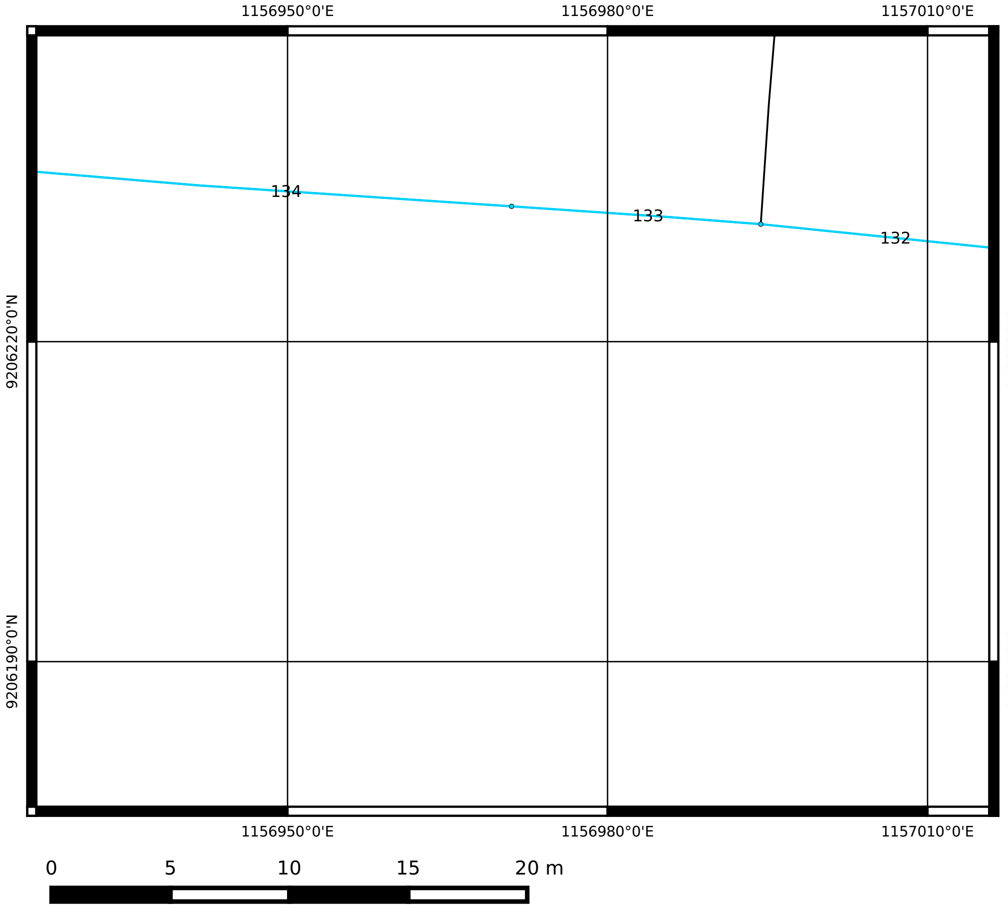


Details for the generated route 310179_KV_H_Midtbyen

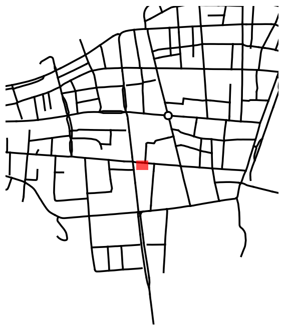


Legend



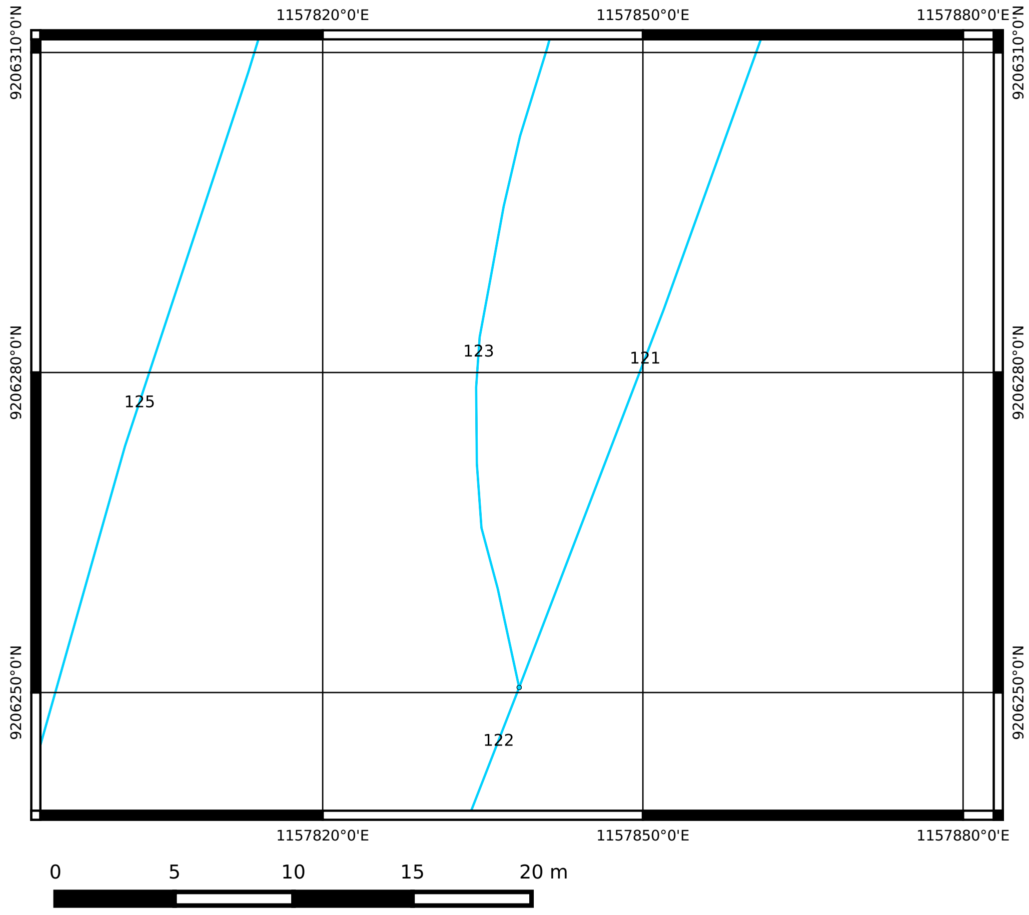


Details for the generated route 310179_KV_H_Midtbyen

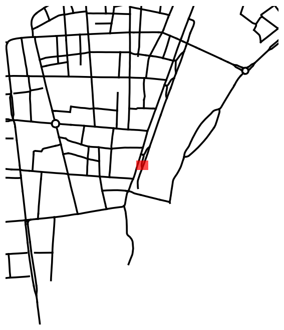


Legend

● Not Required
● Required
— Not used road

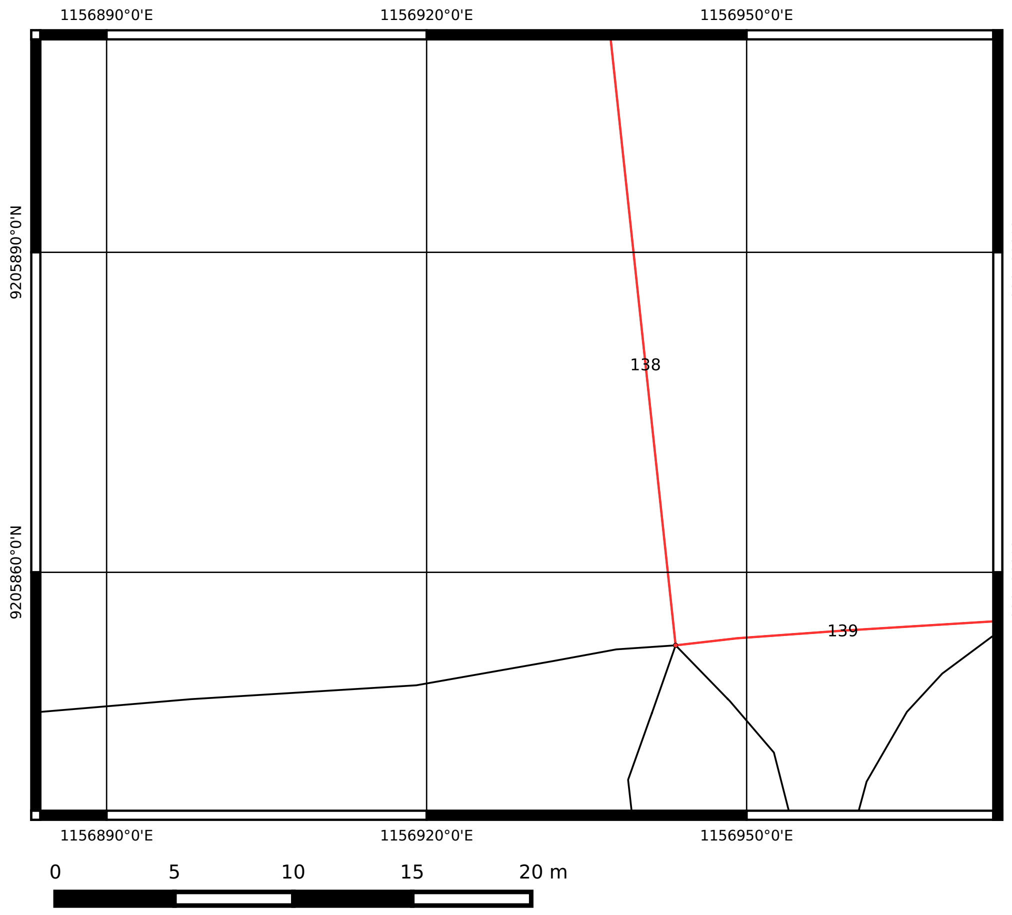


Details for the generated route 310179_KV_H_Midtbyen

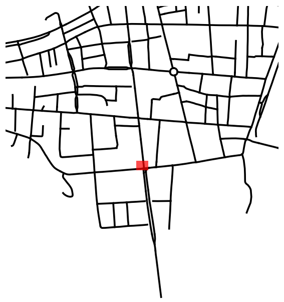


Legend

🟥	Not Required
🟦	Required
—	Not used road



Details for the generated route 310179_KV_H_Midtbyen

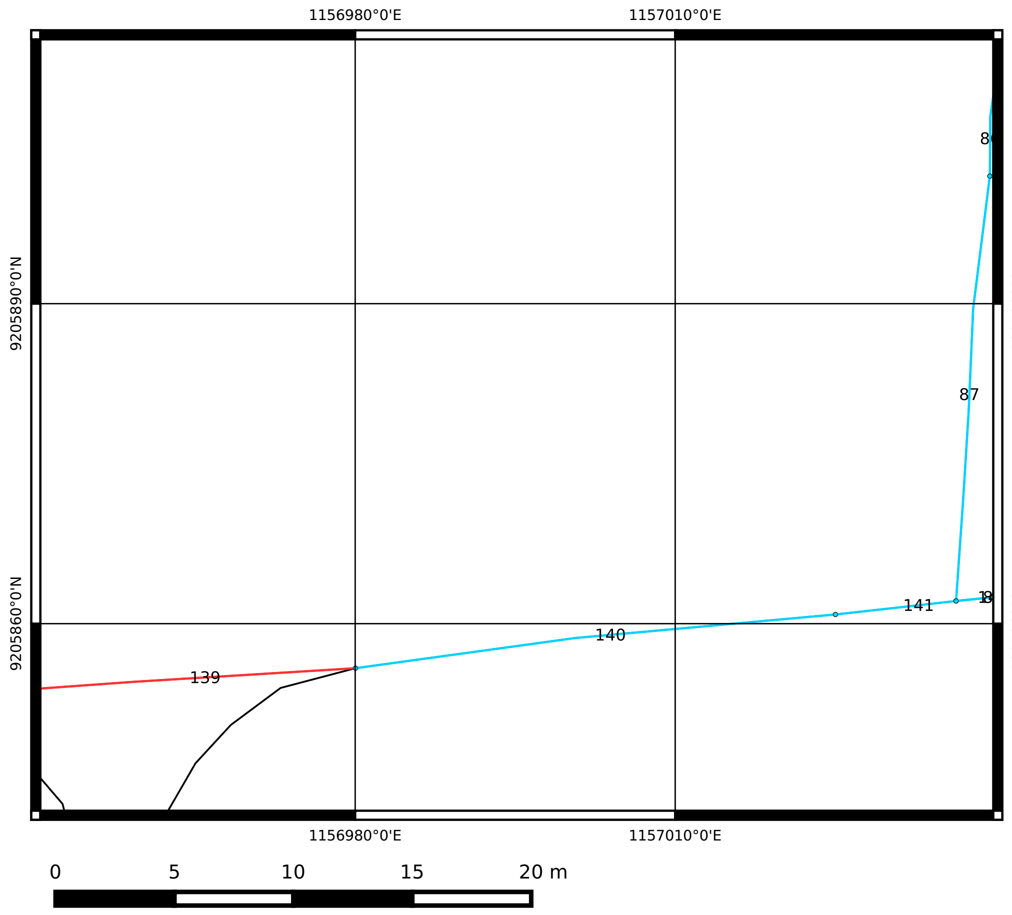


Legend

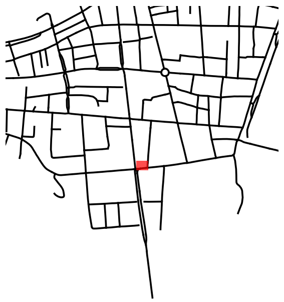
🟥 Not Required

🟦 Required

— Not used road



Details for the generated route 310179_KV_H_Midtbyen

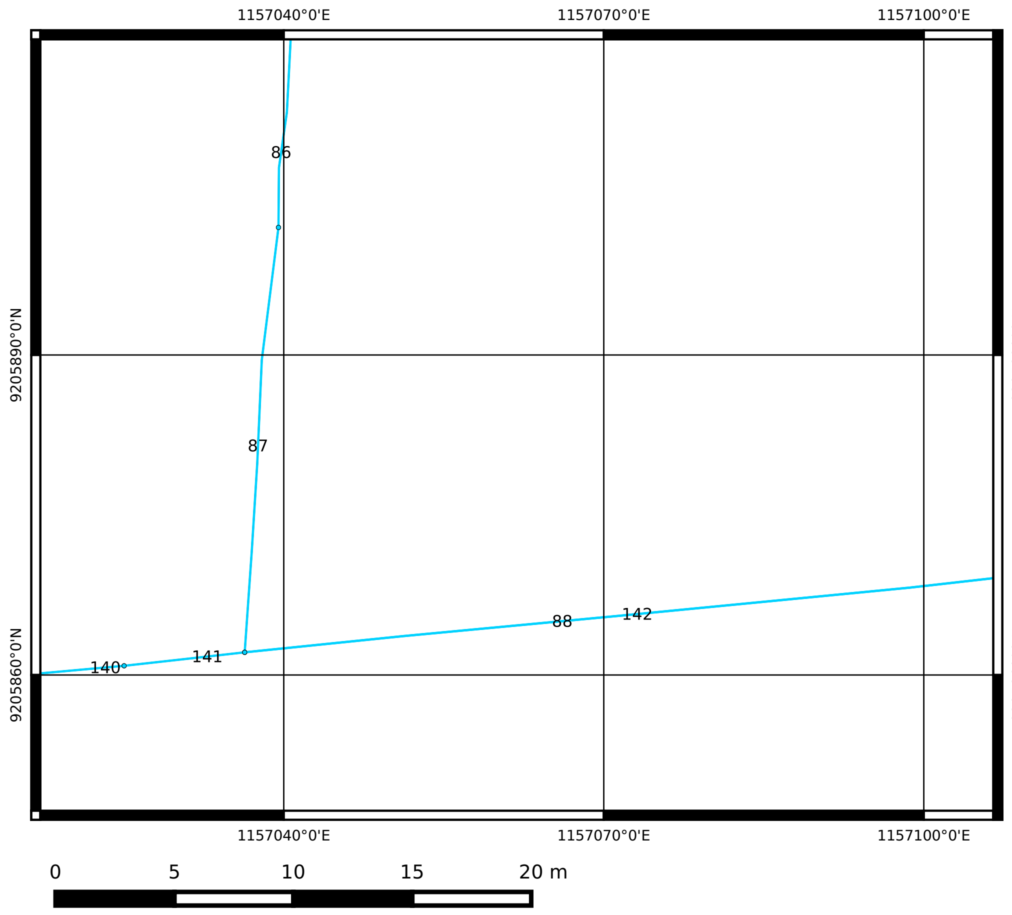


Legend

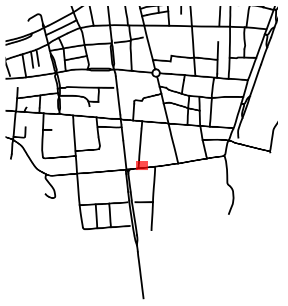
🟥 Not Required

🟦 Required

— Not used road



Details for the generated route 310179_KV_H_Midtbyen



Legend

🟥 Not Required

🟦 Required

— Not used road