

Finding Geographically Separated Paths Through Fiber Networks

Master's Thesis

Author:
M. Boers

First supervisor:
Dr. J.A. Hoogeveen
Second supervisor:
Prof. Dr. H.L. Bodlaender



Universiteit Utrecht

Abstract

The KPN fiber network is a network of fiber cables in The Netherlands. Since only a part of the capacities of the cables are used for KPN's own services, the unused fibers can be interconnected to create and sell fiber connections to clients. Clients often request two connections such that if a connection fails, there is still one working connection left. To reduce the probability of both connections failing simultaneously, these connections need to be geographically separated as much as possible. Currently, finding these routes is a manual task. This thesis introduces and compares various ways to automate this process.

A set of fiber connection problems is modeled as an integer linear program, consisting of an integer multi-commodity flow model with additional constraints. Then, an algorithm is introduced to efficiently solve smaller fiber connection problems to optimality. In addition, a second method of solving the problem based on simulated annealing is introduced. To reduce the magnitude of the problem, four filtering preprocessing steps are introduced to reduce the size of the network. The preprocessing steps accomplish a very large reduction in the size of the problem in a short time.

The ILP solver works well on small problems, but can take a very long time on larger problems. This time can be significantly reduced by solving the problem with only a set of important constraints first, and only adding constraints when needed. The simulated annealing solver finds good solutions in a reasonable time on problems of all sizes.

Two heuristic methods to solve the problem are introduced. A polynomial time heuristic which solves each fiber connection one by one using a shortest path algorithm, and another which solves every problem in an instance to optimality, but one by one instead of simultaneously.

To find the balance between the quality of the solution and the time the algorithm takes to run, the introduced ways of solving the problem are used to introduce a combined heuristic and a combined solver. The results from the algorithms are much better than the results that were found manually.

Contents

1	Introduction	4
2	Problem description	5
2.1	Types of dark fiber products	5
2.2	Network structure	7
3	Analysis of the problem	11
3.1	Complexity	11
3.1.1	NP-completeness of the multiple (single) dark fiber problem	11
3.1.2	NP-completeness of the dual dark fiber problem	12
3.2	Defining the formal problem and solution	15
3.2.1	Network definition	15
3.2.2	Terminology	16
3.2.3	Solution definition	17
4	Solving the problem	18
4.1	Generation of single point of failures	18
4.2	Adding virtual cables	21
4.3	Examination of methods for solving problems with single point of failure constraints	27
4.3.1	Methods based on pathfinding algorithms	27
4.3.2	Augmenting flow algorithms	28
4.3.3	Integer linear programming solutions	30
4.4	A simple heuristic	32
4.5	ILP formulation	33
4.6	Solving the ILP	36
4.7	A more complex heuristic	37
4.8	Preprocessing	38
4.8.1	Preprocessing step based on shortest path distance (SP)	38
4.8.2	Preprocessing step based on availability (FF)	40
4.8.3	Preprocessing step based on hierarchy (HF)	40

4.8.4	Preprocessing step based on straight line distance (SL)	44
4.8.5	Chaining preprocessing steps	44
4.9	Solving with local search	45
4.9.1	Mutation step	45
4.9.2	Simulated annealing	47
4.10	Combining solvers	48
4.10.1	Providing the local search solver with initial solutions	48
4.10.2	Providing the ILP solver with initial solutions	49
4.10.3	Combining all solvers	49
5	Experiments	51
5.1	Instances	51
5.2	Preprocessing	53
5.3	Heuristics	57
5.3.1	Simple heuristic	57
5.3.2	Complex heuristic	58
5.3.3	Combining heuristics	58
5.4	ILP	59
5.4.1	Important single point of failures	59
5.5	Local Search	61
5.5.1	Randomness factor	62
5.5.2	Higher Q	63
5.6	Combined solver	63
5.6.1	Comparison between all solvers	63
5.6.2	Comparison to consultants	66
6	Discussion	67
6.1	Preprocessing	67
6.1.1	Preprocessing step based on hierarchy	68
6.1.2	Preprocessing based on straight line distance	69
6.2	Heuristics	69
6.2.1	Simple heuristic	69
6.2.2	Complex heuristic	70
6.2.3	Combined heuristic	71
6.3	ILP	72
6.3.1	Important single point of failures	72
6.4	Local search	73
6.4.1	Randomness factor	74
6.4.2	Initial solution	75
6.4.3	Higher Q	75
6.5	Combined solver	77

6.6	Reflection	78
6.6.1	Perspective on the interpretation of the experiments	78
6.6.2	Network single point of failures	78
6.6.3	Score definition	79
6.6.4	Other methods	80
7	Conclusion	81
7.1	Overview	81
7.2	Best methods to solve problems	84
7.3	Comparison to the current situation	85
A	Data interpretation	86
A.1	Inter node network	86
A.1.1	Endpoints	86
A.1.2	Geographical information	87
A.1.3	Lengths	87
A.1.4	Hierarchy information	87
A.2	Local network	88
A.2.1	Algorithm to interpret drawings	89
B	Details on the instances	92
C	Details on the experimental results	96

Chapter 1

Introduction

The KPN fiber network is a network of fiber cables that covers the entire country of the Netherlands. A fiber cable consists of several individual fibers. Only part of these fibers are used for KPN's own telecommunication services. The rest of the cable is not used, and thus it is *dark*. The unused fibers can be used for a service called *dark fiber*. Dark fiber is a basic service in which a client requests one or more physical fiber connections between two or more locations. When a dark fiber connection is created, these fibers are physically welded together, such that these fibers are merged into one longer fiber. If an end location is not already connected to the KPN fiber network, an extra cable from the end location to a point on the fiber network has to be added to the network. No further services are being delivered over these connections. This is the responsibility of the client.

To increase the reliability of a connection, clients often request two connections instead of one. Then, if one of the connections fails, the other connection can take over. Connection failures happen mostly when cables are accidentally cut during construction work. Therefore, to lower the probability of both connections failing simultaneously, it is important that the two connections are separated geographically by at least six meters.

Currently, finding out which cables are used to make a dark fiber connection is done manually at KPN. Due to the difficult constraints that dark fiber connections need to satisfy and the large size of the network, this is a time-consuming process, and there is no guarantee about the quality of the results. The goal of this thesis is to find and compare ways to automate the entire process of finding dark fiber connections. This will be done by analyzing the complexity of the problem, examining various existing methods to solve similar problems, formalizing the problem and finally introducing several ways to solve it.

Chapter 2

Problem description

2.1 Types of dark fiber products

KPN sells several dark fiber products. Clients can order one such product, or multiple products at once.

There are several types of dark fiber products:

Single dark fiber

The simplest form of dark fiber product involves a client requesting one physical fiber connection between two locations. See Figure 2.1. A client requests a specific number of fibers for the connection. All these fibers must follow the same path. Clients may have extra demands concerning the quality of the connection. The client will eventually connect lasers to their dark fibers. These lasers need to be stronger for longer routes. Because lasers that are too strong are dangerous, routes that are longer than 100 kilometers are never delivered. Requests by clients for routes that need to be longer than 100 kilometers are always denied.

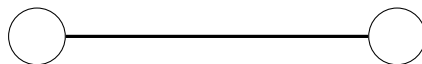


Figure 2.1: Single dark fiber between two locations

Dual dark fiber

A disadvantage of the single dark fiber is that any failure along the route will break the connection. Most often such a failure occurs accidentally when a cable is broken during construction work. To lower the risk of failure, clients can choose for a dual dark fiber. A dual dark fiber consists of two dark fiber connections between two locations. All fibers

within a connection must follow the same path. See Figure 2.2. If one of the connections fails, the other connection can take over. The switching of the connections is the responsibility of the client.

To maximize security, it is necessary for the two paths to be *geographically separated* as much as possible. Fully geographically separated routes are at least a fixed distance of six meters apart from each other along the entire route. A part of the path on which the two connections are not geographically separated is called a *single point of failure*.

The connection length constraint of 100 kilometers still holds. This means that neither of the two connections individually is allowed to be longer than 100 kilometers.

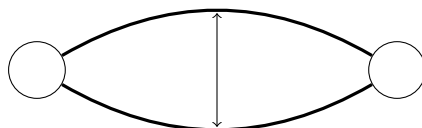


Figure 2.2: Dual dark fiber between two locations. The double arrow represents geographical separation along the two entire edges.

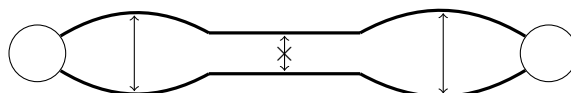


Figure 2.3: Dual dark fiber with a single point of failure between two locations. The crossed double arrow represents a single point of failure.

Network dark fiber

In a network dark fiber, a client requests two geographically separated connections, and both connections share one endpoint. See Figure 2.4

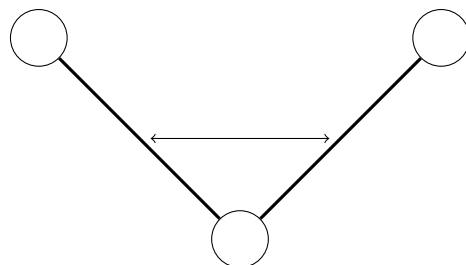


Figure 2.4: Network dark fiber. The double arrows represent geographical separation.

Dark fiber ring

In a dark fiber ring, a client requests dark fiber connections between more than two locations in a ring, i.e. each location is directly connected to exactly two other locations, and each connection is geographically separated from all other connections. See Figure 2.5. If one connection fails, the two locations connected to it are still reachable by switching over all other locations.

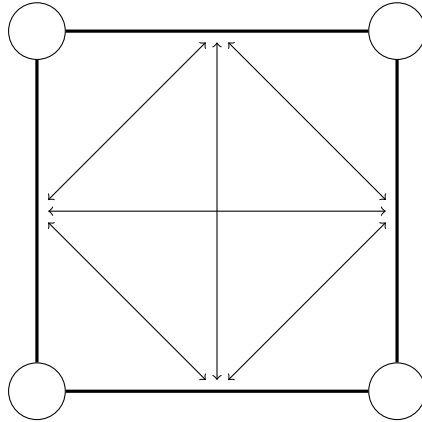


Figure 2.5: Dark fiber ring between four locations. The double arrows represent geographical separation.

2.2 Network structure

The entire KPN fiber network contains 1383 main hubs to which the main cables are connected. There are also several thousands of smaller hubs which interconnect the smaller local cables. There are four special main hubs called *backbone locations*.

Each main hub has two entries which can be used to connect cables to, see Figure 2.6. This constraint limits the possibilities of finding fully geographically separated routes, because all cables that are connected to the same entry form a single point of failure with each other.

Several types of cables are connected to the main hubs:

- INN (*Inter Node Net*) cables

These are cables in between the 1383 main hubs. An inter node cable has two endpoints, both of which are main hubs. Within the inter node network, there are several subnetworks:

- Backbone networks

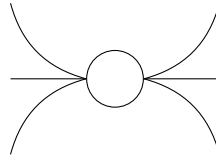


Figure 2.6: A hub with three cables connected to each of its two entries. Clearly cables connected to the same side cannot be fully geographically separated.

There are two subnetworks called *backbone A* and *backbone B*. These two subnetworks both interconnect all four backbone locations. Backbone A and backbone B are guaranteed to be geographically separated from each other. This makes it easy to find two geographically separated routes between backbone locations, by making one route use the backbone A subnetwork, and the other route use the backbone B subnetwork.

- MC rings

MC (*Metro Core*) rings are subnetworks forming rings between main hubs. All cables in an MC ring are supposed to be geographically separated from one another. This would make it easy to find two geographically separated routes between main hubs on the same ring, by making one route go over one side of the ring, and the other route go over the other side. However, in practice, not all MC rings are truly geographically separated.

- Local cables

The country of the Netherlands is divided into 1383 geographical areas, and each main hub is in exactly one of these areas. Local cables form a small local network within the geographical area of the main hub they are connected to. The local cables are interconnected by the smaller hubs. Although these local networks are supposed to stay inside the geographical area of their main hub, there are often exceptions to this in which a local network extends beyond the borders of its area, or in which different bordering local networks are interconnected. There are three types of local cables:

- PAN

PAN (Dutch: *Primair AansluitNet*) cables are connected to one or more main hubs, or to other PAN cables.

- SAN

SAN (Dutch: *Sedundair AansluitNet*) cables are connected to PAN cables or to other SAN cables.

- TAN

TAN (Dutch: *Tertiair AansluitNet*) cables directly connect the client to a SAN cable. These cables often do not exist yet, and have to be created. This is an expensive operation, so we want to minimize the length of the needed TAN cable.

PAN cables cannot be used to connect a TAN cable to, unless the PAN cable has the GBT (Dutch: *geleidebuis techniek*) property, or (part of) the PAN cable is marked as a fictional SAN.

Every cable consist of a number of fibers (the capacity). Multiple clients can be using the same cable. Therefore, some fibers of a cable may be in use by other clients or services. This should be taken into account, such that only cables with enough free fibers are used for new connections.

See Figure 2.7 for an abstract drawing of the structure of the network.

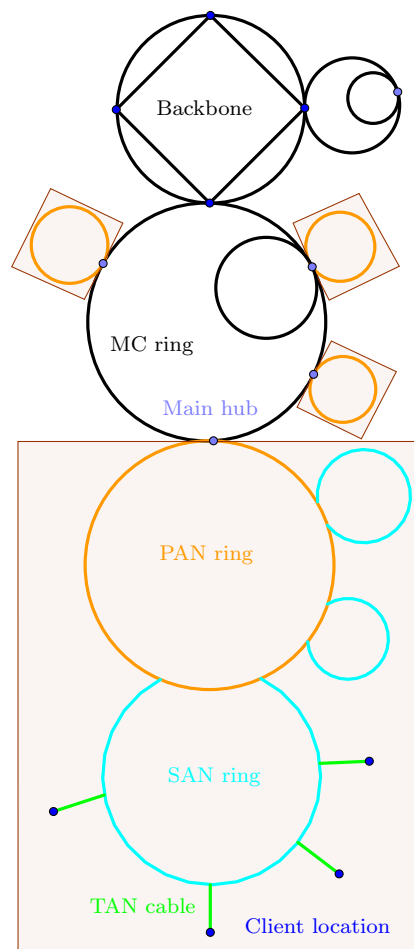


Figure 2.7: Idealized overview of the structure of the network. The brown boxes each represent one of the 1383 geographical areas the country is divided into. Therefore, cables within one of the brown boxes are local cables. Black cables, outside the brown boxes, are INN cables. In reality, many cables do not form closed rings, some cables extend beyond their geographical area, and there are cables between different rings.

Chapter 3

Analysis of the problem

3.1 Complexity

To make informed decisions about which algorithms to use and to consider for solving the problem, the complexity of the problem needs to be analyzed. This section shows proof that finding paths for any of the dark fiber products involving two or more paths is NP-hard.

3.1.1 NP-completeness of the multiple (single) dark fiber problem

Lemma 1. *Given a fiber network, where every cable has a number of available fibers. The problem of finding two or more paths of one or more fibers between different locations is NP-complete.*

Proof. A solution can be verified in polynomial time by verifying for every cable in every path that they share an endpoint with another cable in the path or one of the locations the route runs between, and that the number of fibers of the route is less than or equal to the availability of the cable. Hence, the problem is in NP.

The problem of finding integer multi-commodity flow in an undirected graph, shown by Even et al. [1] to be NP-complete, can be reduced to the multiple single dark fiber problem in polynomial time. Every commodity can be represented as a single dark fiber problem, where the source and sink of the commodity are the end locations of the single dark fiber problem. The flow demand for a commodity can be represented as the number of fibers for the single dark fiber, and the capacity of an edge can be represented as the number of unused fibers in a cable.

A solution to the multiple single dark fiber problem then equals a solution to the integer multi-commodity flow problem. Hence, the multiple single dark fiber problem is NP-complete.

□

3.1.2 NP-completeness of the dual dark fiber problem

As was proven in Section 3.1.1, finding two or more paths between different locations in the dark fiber network is NP-complete. However, in the special case in which the start and end locations of the two paths are equal, the problem can be solved in polynomial time.

The dual dark fiber problem is such a problem in which the start and end locations are equal. However, due to the additional constraint of geographical separation, this problem is still NP-complete.

The following NP-completeness proof for the geographical separation constraint was derived from personal communication with Hans L. Bodlaender.

Lemma 2. *Given the undirected graph $G = (V, E)$ representing a fiber network, and set $S \subseteq \{\{e_1, e_2\} \mid e_1 \in E, e_2 \in E\}$ of unordered pairs of edges in E , representing two cables which are geographically separated from each other. The problem of finding two paths $P_1 \subseteq E, P_2 \subseteq E$ from $s \in V$ to $t \in V$ for which there is no pair of edges in S for which both paths use both edges is NP-complete.*

Proof. A solution can easily be verified in polynomial time by verifying for every pair of edges in S that neither path contains this edge. Hence, the problem is in NP.

The 3-SAT problem, proven by Karp [2] to be NP-complete, can be reduced to this problem in polynomial time. Consider the 3-SAT problem consisting of a set of variables $x_1, x_2 \dots x_n$, their literals $L = x_1, \neg x_1, x_2, \neg x_2 \dots x_n, \neg x_n$ and clauses $c_1, c_2 \dots c_m$ each consisting of three literals $c_i = (l_{i1}, l_{i2}, l_{i3})$ where $l_{i1}, l_{i2}, l_{i3} \in L$. The graph from Figure 3.1 can be created, consisting of:

- Vertices $x_i, x'_i, \neg x_i$ and $\neg x'_i$ for every variable
- Edges (x_i, x'_i) and $(\neg x_i, \neg x'_i)$ for every variable
- Edges $(x'_i, x_{i+1}), (x'_i, \neg x_{i+1}), (\neg x'_i, x_{i+1})$ and $(\neg x'_i, \neg x_{i+1})$ for every variable
- Vertices $l_{i,1}, l_{i,2}, l_{i,3}, l'_{i,1}, l'_{i,2}$ and $l_{i,3}$ for every clause
- Edges $(l_{i,1}, l'_{i,1}), (l_{i,2}, l'_{i,2}), (l_{i,3}, l'_{i,3})$ for every clause
- Edges $(l'_{i,1}, l_{(i+1),1}), (l'_{i,1}, l_{(i+1),2}), (l'_{i,1}, l_{(i+1),3}), (l'_{i,2}, l_{(i+1),1}), (l'_{i,2}, l_{(i+1),2}), (l'_{i,2}, l_{(i+1),3}), (l'_{i,3}, l_{(i+1),1}), (l'_{i,3}, l_{(i+1),2}), (l'_{i,3}, l_{(i+1),3})$

See Figure 3.2 for an example. There are a maximum of three literals in every clause, and each clause is in S exactly once. Hence the reduction is of polynomial size.

Because the two edges from s are in single point of failure with themselves, two paths starting from s that do not violate this constraint will have to go over both these edges. This forces one path to follow the side of the graph containing the variables, and the other path to follow the side of the graph containing the clauses.

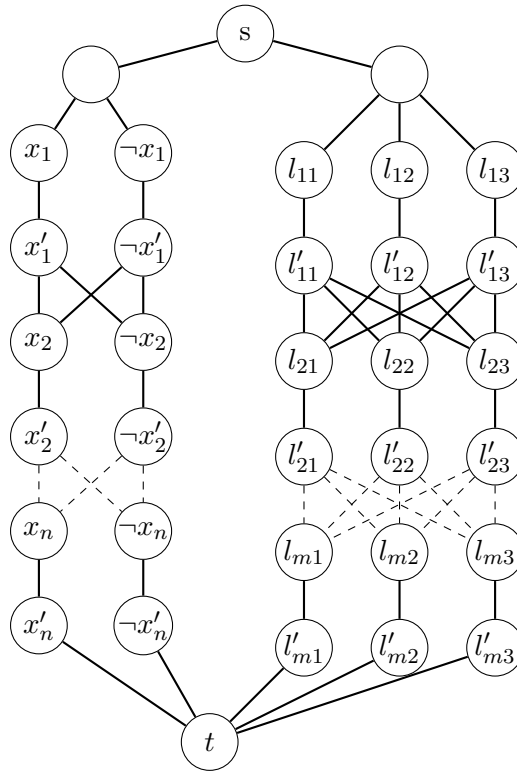


Figure 3.1: Network for reduction from the 3-SAT problem

As can be seen in Figure 3.1 and 3.2, one path from s to t will have to go through exactly one edge representing a literal for every variable, and the other path from s to t will have to go through exactly one edge representing a literal for every clause. This makes sure the \wedge and \vee operators in the 3-SAT formula are satisfied. The single point of failure constraints make sure there can be no contradictions between the literals. Therefore, if there are two paths that do not violate any single point of failure constraint, there must be a configuration of variables for which the 3-SAT formula is satisfied. Without the single point of failure constraints representing contradictions, two paths would always be possible. Therefore, if it is impossible to make two paths satisfying these constraints, it means that it is impossible to satisfy the 3-SAT formula without any contradictions.

If a solution to the problem exists, one of the paths will visit the nodes representing the values of all variables such that all clauses are satisfied in the 3-SAT problem. If no solution to the problem exists, the 3-SAT problem must be unsatisfiable.

□

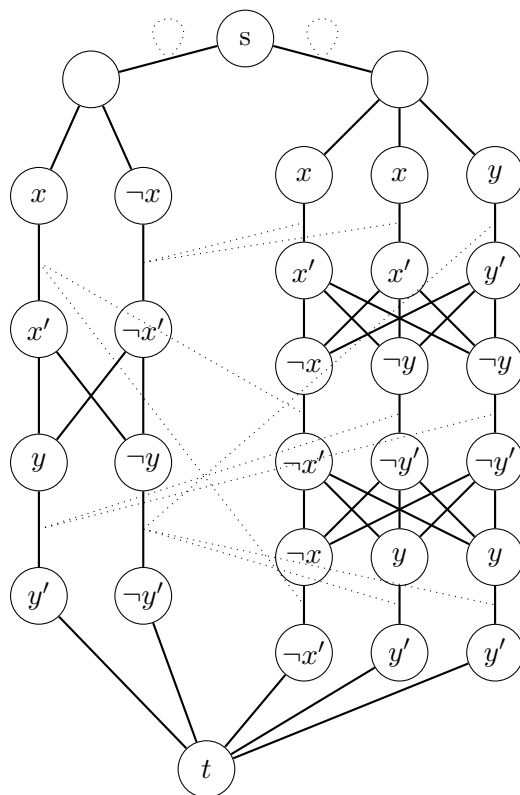


Figure 3.2: Network for reduction from the 3-SAT problem $(x \vee x \vee y) \wedge (\neg x \vee \neg y \vee \neg y) \wedge (\neg x \vee y \vee y)$. Dotted lines represent pairs of edges in the set S .

3.2 Defining the formal problem and solution

As we have seen in Chapter 2, there are several different dark fiber products. We want to be able to create routes for any of these dark fiber products, and also group several of these products into a larger problem. To accomplish this more easily, we make one uniform problem definition for all dark fiber products.

3.2.1 Network definition

The entire network, consisting of both inter node and local cables, is modeled as an undirected graph $G = (V, E)$ and a set of single point of failures S .

Nodes and edges

A node $v \in V$ in the graph represents either a main hub in the inter node network, or a possible welding point in the local network. The possible welding points are certain locations where multiple cables end. Fibers from these cables can be welded together into a new, longer cable at these points to create a connection.

An edge $e = \{u, v\} \in E$ from node $u \in V$ to $v \in V$ in the graph represents a cable or a part of a cable in the network. An edge e has several properties:

- e_{free} , the number of free fibers in the cable corresponding to edge e
- e_{length} , the length of the (part of the) cable corresponding to edge e
- $e_{hierarchy}$, the hierarchy (TAN/SAN/PAN/INN) of the cable corresponding to edge e

Cables which do not exist yet and have to be dug, from the location of the client to a SAN cable, are also represented as edges in the model. We call these cables *virtual cables*. These cables will always be TAN cables. Virtual cables will be further explained in Section 4.2.

Single point of failures

For ensuring geographical separation, extra information about the location of the cables needs to be included into the model. Since this comprises a large amount of data, the model only includes the minimum amount of data required to ensure geographical separation: one set S of single point of failures. The advantage of this approach is that the set will only have to be generated once, and after that the (slow) single point of failure generation algorithm will not have to run anymore. See Section 4.1 for a detailed explanation of the generation algorithm. A single point of failure s consists of an unordered pair of edges $\{e_1, e_2\}$ which are not geographically separated, and a length s_{length} of the distance over which these two edges are not geographically separated. This generates a significantly

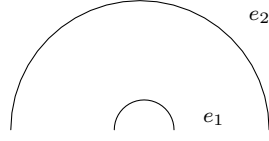


Figure 3.3: Suppose e_1 and e_2 are cables which are near each other, i.e. they are within six meters of each other. The distance over which e_1 is near e_2 is higher than the distance over which e_2 is near e_1 .

smaller set than counting every area at which cables are near each other as a single point of failure, because many cables are in single point of failure with each other multiple times, but this information is not relevant to the solution to the problem.

The distance over which edges $\{e_1, e_2\}$ in a pair of edges are within a certain distance of each other, does not have to be the same for e_1 and e_2 , see Figure 3.3. This constitutes a problem for deciding the length of a single point of failure. Consider the extreme case in which e_1 is very short and e_2 is very long. In this case, when e_2 is broken, the probability of e_1 also being broken is very low. However, if e_1 is broken, the probability of e_2 also being broken is very high. Therefore, the length of the single point of failure is defined to be the minimum of the two distances over which the two cables are near the other cable.

3.2.2 Terminology

Subproblem

A subproblem $r = (s_r, t_r, fibers_r) \in R$ is defined to be a pair of locations (s_r, t_r) between which one dark fiber connection will have to be constructed, and a number of fibers $fibers_r$ that needs to be available along the entire path for this dark fiber connection. A location can be either an existing location in the network, or a new location that needs to be created. A subproblem represents one dark fiber connection. For example, one dual dark fiber contains two subproblems.

Problem

A problem $P \in PS, P \subseteq R$ is defined to be a set of subproblems $r \in R$, for which every subproblem needs to be geographically separated from other subproblems in the same problem as much as possible. A problem represents one order of a dark fiber product. For example, one dual dark fiber order is represented as one problem containing two subproblems.

Problem set

The problem set PS is a set of problems that need to be solved simultaneously. The solutions to these problems are related through the number of fibers that need to be reserved for a solution, i.e. if a solution of a subproblem with $fibers_r = 2$ uses a cable that has only two fibers available, the path for another subproblem cannot use this cable anymore. A problem set represents a set of orders of dark fiber products.

3.2.3 Solution definition

A solution to a problem set is represented as a set of paths, one for every subproblem, and a set of unsolved problems. Unsolved problems occur when one of the subproblems in a problem set is infeasible, or when two or more subproblems cannot simultaneously be solved due to the capacity constraint on cables.

Score calculation

The ultimate goal is to find the best solution to a problem set. To compare the quality of solutions, each solution is assigned a score. This score is the sum of:

- The length of the route in meters, where virtual cables are multiplied by a certain penalty C_{virt}
- The lengths of all violated single point of failures in meters, multiplied by a certain penalty C_{spof}
- The number of unsolved problems, multiplied by a certain penalty $C_{unsolved}$

The values for C_{spof} , $C_{unsolved}$ and C_{virt} are chosen as follows:

- $C_{spof} = 100001$, because the maximum length a route is 100000, and 1 meter without geographical separation is worse than having a route of maximum length
- $C_{unsolved} = 10000100000$, because an unsolved problem is worse than a route that is not geographically separated
- $C_{virt} = 1000$

Chapter 4

Solving the problem

4.1 Generation of single point of failures

Because of the decision to use one set of single point of failures in the model, as explained in section 3.2.1, this set will have to be generated once. The geographical position of every cable is known, in the form of a set of connected line segments. The information that needs to be extracted from this is the distance for which a cable is within six meters of another cable, because if two cables are within six meters of each other, they form a single point of failure. A drawing of a cable is represented by a string of connected line segments.

The main idea of the single point of failure generation algorithm is to consider all individual line segments that make up a cable, and compare each line segment to all other line segments. For each line segment, the part of the line segment that is near another cable is being colored, using a different color for every other cable. It is not possible to simply add the distances for which a certain line segment is close to another cable to each other, because this cable may be made up of multiple line segments that are all nearby. See Figure 4.1.

Even though the running time of the algorithm is not very important, comparing every line segment to every other line segment takes far too much time to make the algorithm terminate in a reasonable time. To reduce the number of comparisons to be made, an approach based on sweep line algorithms [3] is used. The topmost and bottommost points of all line segments are sorted on y-coordinate. Then the algorithm loops over this list. For every line segment, there is a set of line segments that overlap it on the y-axis. It only compares this line segment to the other line segments in this list. To be sure each line segment is being compared to all other relevant line segments, six meters is subtracted from every topmost point, and six meters is added to every bottom-most point. See Algorithm 1 for the pseudocode, and Figure 4.2 for a drawing.

Comparing a line segment to another line segment involves cutting the line segment into three parts. For two of these parts, the closest point on the other line segment is one

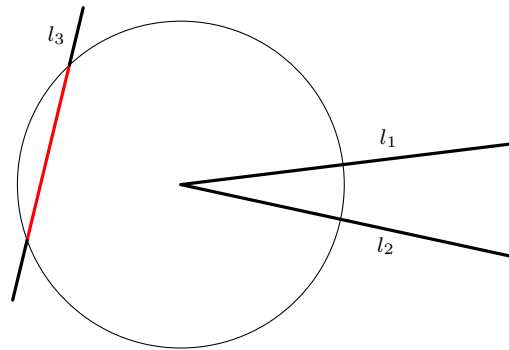


Figure 4.1: Suppose l_1 and l_2 both belong to the same cable c , and l_3 belongs to another cable. The nearest point on c to l_3 is the point where l_1 and l_2 are connected. The circle has a radius of the single point of failure distance, six. Then, if we would add the distance for which l_1 is close to l_3 to the distance for which l_2 is close to l_3 , we would end up with the length of the red area twice, while it is only near c for the distance of the red area once.

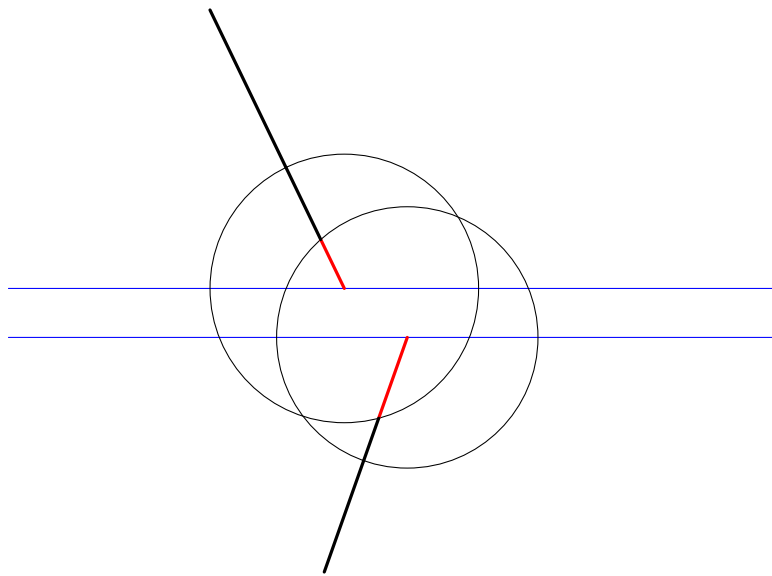


Figure 4.2: Even though the two line segments do not overlap on the y-axis (see blue lines), they are still within six meters from each other on the red areas (the circles have a radius of the single point of failure distance, six), and therefore form a single point of failure. That is why we also compare to line segments within six meters of the top and the bottom.

Algorithm 1 Compares every line segment of a cable to every relevant line segment of every other cable. The COLOREDDISTANCE function returns the distance for which the line segment in the first argument is colored for the line segment in the second argument.

```

1: function SPOFGENERATION( $G = (V, E)$ )
2:    $S \leftarrow \emptyset$ 
3:    $lineSegments \leftarrow \bigcup_{e \in E} \text{LINESEGMENTS}(e)$ 
4:    $events \leftarrow \emptyset$ 
5:   for all  $l = (p_{top}, p_{bot}) \in lineSegments$  do
6:      $events \leftarrow events \cup \{(p_{top}.Y - 6, l, \text{entry}), (p_{bot}.Y + 6, l, \text{leave})\}$ 
7:   end for
8:   SORT( $events$ )
9:    $current \leftarrow \emptyset$ 
10:  for all  $(p_e, l_1, type) \in events$  do
11:    if  $type = \text{entry}$  then
12:       $current \leftarrow current \cup \{l_1\}$ 
13:      Add  $l_1$  to set of relevant line segments to all line segments in  $current$ 
14:    else if  $type = \text{leave}$  then
15:      for all  $l_2 \leftarrow$  relevant line segments to  $l_1$  do
16:        COMPAREANDCOLOR( $l_1, l_2$ )
17:      end for
18:       $current \leftarrow current \setminus \{l_1\}$ 
19:    end if
20:  end for
21:  for all  $e_1 \in E$  do
22:    for all  $e_2 \in E$  do
23:       $S \leftarrow S \cup (\{e_1, e_2\}, \text{MIN}(\text{COLOREDDISTANCE}(e_1, e_2), \text{COLOREDDISTANCE}(e_2, e_1)))$ 
24:    end for
25:  end for
26:  return  $S$ 
27: end function

```

of the endpoints, and for the remaining part, the closest point on the other line segment is a point on the line segment. See Algorithm 2 on how to cut the line segment into parts, and Figure 4.3 and 4.4 for examples.

Then the single point of failure areas for the top and bottom parts are calculated by Algorithm 3, making a circle of radius six around the top point and coloring the area between the intersection points with l_1 (see Figure 4.5). This is done by first calculating the angle of the two corners which are not the top corner. Then, by using the law of sines, it is calculated what the angle of the remaining corner would be if the sides would have a length of at most six. It is not easily possible to simply use the Pythagorean theorem here, because the triangle may be an obtuse triangle, i.e. one of the angles may be larger than 90 degrees, see Figure 4.6. One could work around this problem, but the current solution is easier and works for every kind of triangle.

The single point of failure area for the middle part is calculated by Algorithm 4, finding the points at which the distance between the two line segments is six by linear interpolation between the distance at the top of the trapezoid and that at the bottom of the trapezoid (see Figure 4.7 and 4.8).

4.2 Adding virtual cables

Cables which do not exist yet, but need to be considered for the solution, need to be added before solving the problem. These are the TAN cables which start at a client and connect the client to a SAN cable. There are eight million possible client locations in the country, so creating all virtual cables in advance is going to increase the size of the network enormously. Therefore, we choose to only add them temporarily when solving a problem. Virtual cables do not exist yet and need to be dug. Digging new cables is more expensive than using existing cables. This is why the length of virtual cable is multiplied by a constant C_{virt} as a penalty for the extra costs. See Section 3.2.3.

The algorithm for adding the virtual cables simply looks at every SAN cable within a radius of 1000 meters of all client locations, and creates a virtual cable following a straight line from the client location to the closest point on the cable. The SAN cable, i.e. the edge representing the SAN cable, is then cut into two parts, and a node is created at the point where the virtual cable intersects the SAN cable.

In reality, the cables will not be following a straight line, because it is not always possible to dig everywhere. An analysis of data from the past shows that the real distance can be approximated fairly well by using the straight line distance times a constant of 1.6. KPN uses a constant of 1.4 when determining the pricing. However, since we already multiply the length of the virtual cable with a constant, C_{virt} , to determine its weight in the score, it is not necessary to adjust the length of the virtual cable.

Virtual cables, unlike other cables, are always considered to be geographically separated, because they can easily be dug in such a way that they do not form a single point of failure,

Algorithm 2 Colors the part of line segment l_1 for which l_1 is near l_2 . To accomplish this, l_1 is being cut into three parts: a top triangle, a middle trapezoid/hourglass, and a bottom triangle. See Figure 4.3 and 4.4 for examples.

```

1: function COMPAREANDCOLOR( $l_1, l_2$ )
2:                                     ▷ The top part
3:    $bl_1 \leftarrow$  the line perpendicular to  $l_2$ , through  $l_{2,top}$ 
4:   if INTERSECTS( $l_1, bl_1$ ) then
5:      $q_1 \leftarrow$  INTERSECTIONPOINT( $l_1, bl_1$ )
6:   else
7:                                     ▷ Side of  $bl_1$  that  $p_0$  is on
8:      $posp \leftarrow$  SIGN( $(p_0.X - bl_{1,top}.X) * (bl_{1,bot}.Y - bl_{1,top}.Y) - (p_0.Y - bl_{1,top}.Y) * (bl_{1,bot}.X - bl_{1,top}.X)$ )
9:                                     ▷ Side of  $bl_1$  that  $p_{l_2,bot}$  is on
10:     $poso \leftarrow$  SIGN( $(l_{2,bot}.X - bl_{1,top}.X) * (bl_{1,bot}.Y - bl_{1,top}.Y) - (l_{2,bot}.Y - bl_{1,top}.Y) * (bl_{1,bot}.X - bl_{1,top}.X)$ )
11:  end if
12:   $a_1 \leftarrow$  the line segment between  $l_{2,top}$  and  $p_1$ 
13:   $b_1 \leftarrow$  the line segment between  $l_{2,top}$  and  $q_1$ 
14:   $t_{top,start}, t_{top,end} \leftarrow$  TRIANGLEPARTSPOF( $a_1, b_1$ )
15:  Color the part between  $t_{top,start}$  and  $t_{top,end}$  on  $l_1$  for the cable belonging to  $l_2$ 
16:                                     ▷ The bottom part
17:   $bl_2 \leftarrow$  the line perpendicular to  $l_2$ , through  $l_{2,bot}$ 
18:  if there is an intersection between  $l_1$  and  $bl_2$  then
19:     $q_2 \leftarrow$  INTERSECTIONPOINT( $l_1, bl_2$ )
20:  else
21:                                     ▷ Side of  $bl_2$  that  $p_2$  is on
22:     $posp \leftarrow$  SIGN( $(p_2.X - bl_{2,top}.X) * (bl_{2,bot}.Y - bl_{2,top}.Y) - (p_2.Y - bl_{2,top}.Y) * (bl_{2,bot}.X - bl_{2,top}.X)$ )
23:                                     ▷ Side of  $bl_2$  that  $p_{l_2,top}$  is on
24:     $poso \leftarrow$  SIGN( $(l_{2,top}.X - bl_{2,top}.X) * (bl_{2,bot}.Y - bl_{2,top}.Y) - (l_{2,top}.Y - bl_{2,top}.Y) * (bl_{2,bot}.X - bl_{2,top}.X)$ )
25:  end if
26:   $a_2 \leftarrow$  the line segment between  $l_{2,bot}$  and  $p_2$ 
27:   $b_2 \leftarrow$  the line segment between  $l_{2,bot}$  and  $q_2$ 
28:   $t_{bot,start}, t_{bot,end} \leftarrow$  TRIANGLEPARTSPOF( $a_2, b_2$ )
29:  Color the part between  $t_{bot,start}$  and  $t_{bot,end}$  on  $l_1$  for the cable belonging to  $l_2$ 
30:                                     ▷ The middle part
31:   $t_{mid,start}, t_{mid,end} \leftarrow$  MIDDLEPARTSPOF(LENGTH( $b_1$ ), LENGTH( $b_2$ ), DISTANCE( $q_1, q_2$ ), INTERSECTS( $l_1, l_2$ ))
32:  Color the part between  $t_{mid,start}$  and  $t_{mid,end}$  on  $l_1$  for the cable belonging to  $l_2$ 
33: end function

```

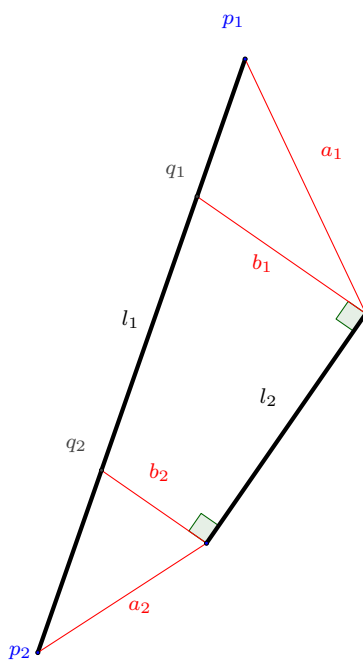


Figure 4.3: l_1 is divided into three parts: a top part between p_1 and q_1 , forming a triangle with an endpoint of l_2 , a middle part between q_1 and q_2 , forming a trapezoid with l_2 , and a bottom part between q_2 and p_2 , forming a triangle with an endpoint of l_2 . For all points on the top and bottom parts, the shortest distance to l_2 is the distance to the point on l_2 with which the part forms a triangle. For all points on the middle part, the closest distance to l_2 is the distance to the intersection of the line perpendicular on l_2 with l_2 . Initially, the locations of both line segments are known. In this example, Algorithm 2 determines the locations of q_1 and q_2 .

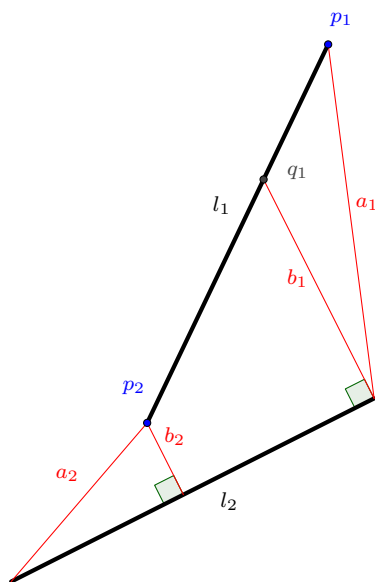


Figure 4.4: l_1 is divided into two parts: a top part between p_1 and q_1 , forming a triangle with an endpoint of l_2 , and a middle part between q_1 and p_2 , forming a trapezoid with a part of l_2 . The third part, between q_2 and p_2 , is empty, because $p_2 = q_2$. Initially, the locations of both line segments are known. In this example, Algorithm 2 determines the locations of q_1 and q_2 .

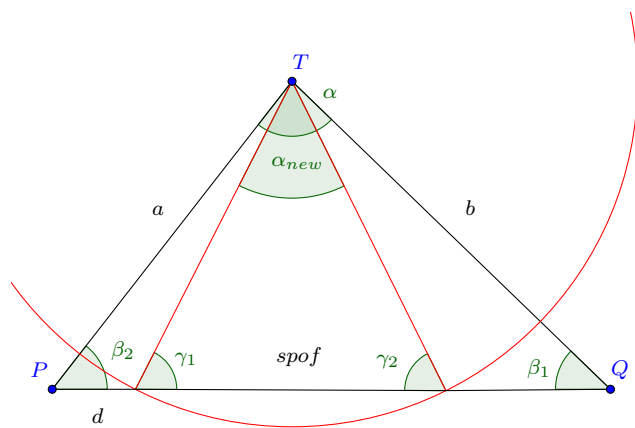


Figure 4.5: Complementary drawing to Algorithm 3. The red circle is centered around T and has a radius of the single point of failure distance, six. The locations of T , P and Q are known. The goal is to determine the length of the part marked d and the length of the part marked $spof$.

Algorithm 3 Calculates the distances from point p along line segment (p, q) at which the distance to top equals six. The difference between the outcome values is the distance of the single point of failure on this part. See Figure 4.5 for a drawing of the calculations that are being made. Note that \sin^{-1} has two solutions, but only one is possible in this case. An implementation needs to include steps to find which one is possible.

```

1: function TRIANGLEPARTSPOF( $a = (T, P)$ ,  $b = (T, Q)$ )
2:    $\alpha \leftarrow \text{ANGLEBETWEEN}(a, b)$  ▷ The top angle of the triangle
3:    $\beta_1 \leftarrow \text{ANGLEBETWEEN}(b, (P, Q))$ 
4:    $\beta_2 \leftarrow \text{ANGLEBETWEEN}(a, (P, Q))$  ▷ The other angles of the triangle
5:    $\gamma_1 \leftarrow \sin^{-1}(\text{LENGTH}(b) * \sin(\beta_1) / \text{MIN}(\text{LENGTH}(a), 6))$ 
6:    $\gamma_2 \leftarrow \sin^{-1}(\text{LENGTH}(a) * \sin(\beta_2) / \text{MIN}(\text{LENGTH}(b), 6))$ 
▷ The other angles if the sides were at most 6 long
7:    $\alpha_{new} \leftarrow \pi - \gamma_1 - \gamma_2$ 
8:    $spof \leftarrow [\text{MIN}(\text{LENGTH}(a), 6)^2 + \text{MIN}(\text{LENGTH}(b), 6)^2 - 2 * \text{MIN}(\text{LENGTH}(a), 6) -$ 
▷ Length of the single point of failure (law of cosines)
    $\text{MIN}(\text{LENGTH}(b), 6) * \cos \alpha_{new}]^{1/2}$ 
9:    $d \leftarrow [\text{LENGTH}(a)^2 + \text{MIN}(\text{LENGTH}(a), 6)^2 - 2 * \text{LENGTH}(a) * \text{MIN}(\text{LENGTH}(a), 6) *$ 
▷ Distance from  $P$  to the start of the single point of failure (law of cosines)
    $\cos(\pi - \beta_1 - \gamma_1) - \alpha_{new}]^{1/2}$ 
   return ( $d, d + spof$ )
10: end function

```

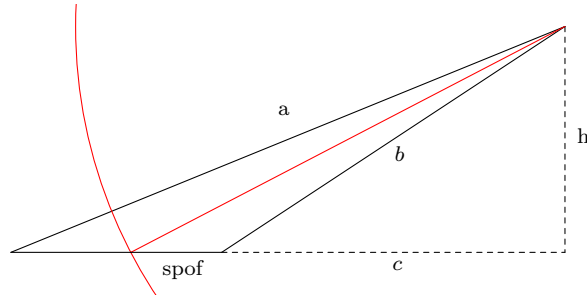


Figure 4.6: Example in which the triangle is obtuse. Algorithm 3 still works. However, if you use the Pythagorean theorem, you would need to find the difference between the two results (c and $c + spof$) in this case, but the sum in the non-obtuse case. This would make the algorithm unnecessarily complicated. Another difficulty is that you would need to calculate the height h .

Algorithm 4 Calculates the minimum and maximum height at which the distance in a trapezoid of height $height$, top width d_{top} and bottom width d_{bottom} equals six. The difference between the outcome values is the distance of the single point of failure on this part. See Figure 4.8 and 4.7 for examples.

```

1: function MIDDLEPARTSPOF( $d_{top}$ ,  $d_{bottom}$ ,  $height$ ,  $intersection$ )
2:   if  $intersection$  then
3:      $d_{top} \leftarrow -d_{top}$ 
4:   end if
5:    $f_{top} \leftarrow (6 - d_{top}) / (d_{bottom} - d_{top}) * height$ 
6:    $s_{top} \leftarrow f_{top} * height$  ▷ Height at which distance is 6
7:    $f_{bottom} \leftarrow (-6 - d_{top}) / (d_{bottom} - d_{top}) * height$ 
8:    $s_{bottom} \leftarrow f_{bottom} * height$  ▷ Height at which distance is -6
9:    $low \leftarrow 0 \leq \text{MIN}(s_{top}, s_{bottom}) \leq height$ 
10:   $high \leftarrow 0 \leq \text{MAX}(s_{top}, s_{bottom}) \leq height$ 
    return ( $low$ ,  $high$ )
11: end function

```

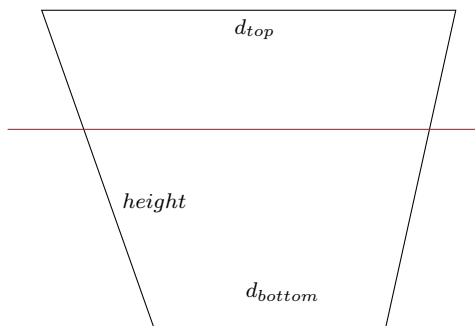


Figure 4.7: Complementary drawing to Algorithm 4. The upper and lower red lines represent respectively the $high$ and low values.

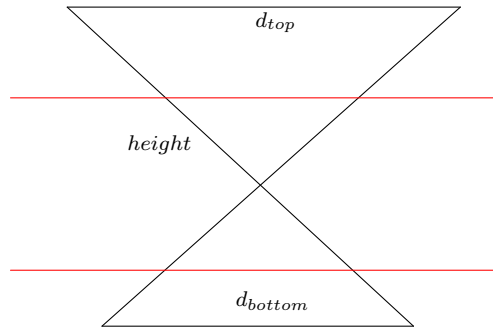


Figure 4.8: Complementary drawing to Algorithm 4. The upper and lower red lines represent respectively the *high* and *low* values.

namely by digging the two cables in opposite directions for three meters before digging towards the SAN cable. This will not be considered for the length of the virtual cable, because three meters is a negligible distance relative to the difference between the assumed distance of the virtual cable (the straight line distance) and the real distance.

Note that for the cable that is being cut, the single point of failures need to be regenerated, because the cable that used to be represented by one edge is now being represented by two edges.

4.3 Examination of methods for solving problems with single point of failure constraints

The most important and most difficult constraint is the one for geographical separation. Several methods for solving similar constraints already exist in literature. The usefulness of these ideas to our problem is being examined.

4.3.1 Methods based on pathfinding algorithms

Suurballe's algorithm

Obviously the single dark fiber problem could easily be solved by using a shortest path algorithm like Dijkstra's algorithm [4], when not expanding nodes with insufficient availability, and terminating after 100 kilometers.

Suurballe's algorithm [5] uses Dijkstra's algorithm to find two shortest edge-disjoint paths in polynomial time. This is a special case of the NP-hard dual dark fiber problem mentioned in Section 3.1, in which all edges are representing cables which are all geograph-

ically separated from each other. Even though this is similar to single point of failures, the NP-hardness of our problem shows that our problem is not polynomial-time reducible to the problem of finding two shortest edge-disjoint paths. To still use Suurballe's algorithm, one could think of simplifying the problem by merging all edges forming a single point of failure into a single edge. Consider the case in which an edge forms a single point of failure with two other edges, but these two other edges do not form a single point of failure with each other. Then all three edges would be joined into one edge, and it would be impossible to use the two edges that are not in single point of failure with each other separately. This might be acceptable if this does not happen often, but because every main hub has only two entries (see Figure 2.6), all cables connected to an entry will be merged. This is why solutions to our problem based on Suurballe's algorithm will lead to bad results, and we will not use it.

4.3.2 Augmenting flow algorithms

Because the dual dark fiber problem can be viewed as a more constrained version of a minimum-cost flow problem, several methods for incorporating the single point of failure constraint into existing flow algorithms are being examined.

Since flow networks use directed graphs, since our network $G = (V, E)$ is undirected, and since it is not possible to define a proper flow conservation constraint on an undirected graph, the network has to be converted to a directed graph $G = (V, A)$. A consequence of this is that the set of single point of failures S also needs to be adjusted for directed edges, resulting in S_{dir} .

We look specifically at the geographical separation constraint. We have the set of single point of failures S_{dir} and their lengths, and for every single point of failure $s = \{(u, v), (w, x)\} \in S_{dir}$ we want flow over arc $(u, v) \in A$, over arc $(w, x) \in A$, or over neither arc, but we want to avoid flow over both if possible.

Branching

One option is to simply solve the problem without looking at the single point of failure constraints. This reduces the problem to a polynomial-time solvable flow problem. Then we branch every single point of failure $s = \{(u, v), (w, x)\} \in S_{dir}$ constraint into two subproblems: one in which there is no flow on arc (u, v) (i.e. remove the arc from the graph), and another one in which there is no flow on arc (w, x) . In the case of one dual dark fiber, the remaining problems can then be solved using a polynomial time min-cost flow algorithm [6]. However, there are $O(2^{|S|})$ such problems, so this results in an algorithm with a worst-case time complexity of $O(2^{|S|})$ times the complexity of a min-cost flow algorithm. Due to the high possible number of single point of failures, this can result in a very slow algorithm.

Another large disadvantage is that this is a hard constraint, and it is impossible to find

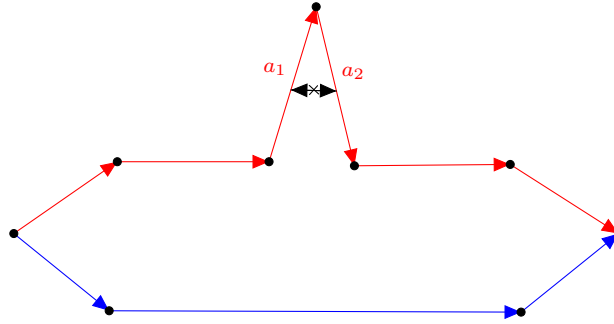


Figure 4.9: Suppose the red and the blue routes represent two connections of a dual dark fiber solution, and arcs a_1 and a_2 form a single point of failure in the set S_{dir} , $\{a_1, a_2\} \in S_{dir}$. Even though both a_1 and a_2 are in use, this should not be considered a violated single point of failure, and should not result in a penalty, because the edges are part of the same connection. It does not matter if a failure of one cable in a connection would make another cable in the same connection fail as well, because the connection would already be interrupted.

the optimal solution if the optimal solution contains at least one single point of failure. This could be fixed by also branching into the problem where s is violated, but this increases the worst-case time complexity to $O(3^{|S|})$ times the complexity of a min-cost flow algorithm. The running time in practice can be improved by reducing the number of single point of failures considered, and by carefully choosing which branches to expand first, but it would still not be possible to solve network dark fibers and dark fiber rings, unless the min-cost flow algorithm would be replaced by a multi-commodity flow variant, increasing the complexity even further. Another problem is that it is not possible for one route to use two arcs which form a single point of failure, while this should not be a problem, see Figure 4.9. This could also be solved by using multiple graphs or using multi-commodity flow algorithms, but then we lose the advantages of this method. That is why we do not use this method.

Adjusting the network

Fößmeier and Kaufmann [7] mention a method of adjusting the network to forbid certain combinations of arcs being used. The network is adjusted such that every single point of failure is remodeled as in Figure 4.10, where M is a high number. This way, using these two edges simultaneously will result in a high cost. A large advantage is that this will result in a soft constraint, but there are a number of disadvantages. One is that there is a very large number of single point of failures, and constructing them all is going to have a significant impact on the running time of the algorithm.

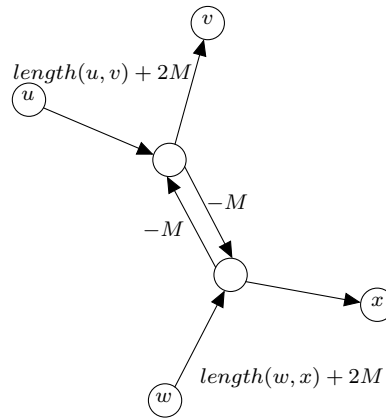


Figure 4.10: Modeling of a single point of failure constraint $s = \{(u, v), (w, x)\}$ by adjusting the network according to the method of Fößmeier and Kaufmann [7]

Fößmeier and Kaufmann recommend solving the flow problem as a linear programming (LP) problem, and using the total unimodularity of the constraint matrix to find an integer flow in polynomial time. This makes this method very attractive for our problem because of the speed and the ease with which extra constraints can be added. However, when adding these extra constraints to the LP, it breaks the total unimodularity of the constraint matrix, and the solution will not be guaranteed to be integer anymore. Another disadvantage is that using two arcs in a single point of failure within the same route, such as in Figure 4.9, is still considered to be a violated single point of failure while it should not be. In addition, the network could become very large due to the large number of single point of failures. Fößmeier and Kaufmann mention that in networks with many constraints, adding the constraints explicitly to the (I)LP will lead to better results.

4.3.3 Integer linear programming solutions

Modeling the entire problem as an integer linear program (ILP) offers some significant advantages over using standard min-cost flow algorithms. When using a model similar to multi-commodity flow, it will be easy to model network dark fibers and dark fiber rings, to make a single point of failure constraint which only holds for separate routes, and to model the availability constraint.

Either-or constraint

An obvious way to model a single point of failure constraint for the single point of failure $s = \{(u, v), (w, x)\} \in S_{dir}$ in an ILP-problem is by using an either-or constraint:

- Variables

$$f_r(u, v) = \begin{cases} 1 & \text{if arc } (u, v) \in A \text{ is used in subproblem } r \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

The same variable would exist for edge (w, x) , and for all subproblems.

$$y_{r,q}(s = \{(u, v), (w, x)\}) = \begin{cases} 0 & \text{if } (u, v) \in A \text{ is not used in subproblem } r \in R \\ 1 & \text{if } (w, x) \in A \text{ is not used in subproblem } q \in R \end{cases} \quad (4.2)$$

- Constraints

$$f_r(u, v) \leq y_{r,q}(s) \quad (4.3)$$

$$f_q(w, x) \leq 1 - y_{r,q}(s) \quad (4.4)$$

If both edges are unused, the value of the variable does not matter. The constraints will always hold.

The goal is to create a constraint that, for a single point of failure $s = \{(u, v), (w, x)\} \in S_{dir}$ and subproblems r and q will disallow flow through both (u, v) and (w, x) in different subproblems, e.g. $f_r(u, v) = 1 \wedge f_q(w, x) = 1$.

If $f_r(u, v) = 1$, and $f_q(w, x) = 1$, the constraints can never both be satisfied. If $y_{r,q}(s) = 1$, then the second constraint fails to hold, and if $y_{r,q}(s) = 0$, the first constraint fails to hold. However, for example the case in which $f_r(u, v) = 1$ and $f_r(w, x) = 1$, with all other flows being 0, will not violate the constraint. $y_{r,q}(s)$ will need to be 1, because $f_r(u, v) = 1$. The first constraint will hold, because $f_r(u, v) = 1$, and the second constraint will hold as well, because $f_q(w, x) = 0$. This solves the issue of Figure 4.9. The main disadvantage is again that this is a hard constraint. Another disadvantage is that the y variable needs to be an integer variable, and because of the high number of single point of failures, there will be many such variables. This is why we also do not choose this method.

Violation constraint

Another way to construct this constraint is by using a variable for violation:

- Variables

$$f_r(u, v) = \begin{cases} 1 & \text{if arc } (u, v) \in A \text{ is used in subproblem } r \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

$$sv_{r,q}(s = \{(u, v), (w, x)\}) = \begin{cases} 0 & \text{if } s \in S_{dir} \text{ is violated} \\ 1 & \text{if } s \in S_{dir} \text{ is not violated} \end{cases} \quad (4.6)$$

- Constraints

$$f_r(u, v) + f_q(w, x) - sv_{r,q}(s) \leq 1 \quad (4.7)$$

$$f_r(w, x) + f_q(u, v) - sv_{r,q}(s) \leq 1 \quad (4.8)$$

Here, all flow variables are integer. Now that we have a variable for violation, we can add $sv_{r,q} * C_{spof}$ to the objective function to penalize violating a single point of failure. This makes it a soft constraint. When using this constraint, $sv_{r,q}$ will always be integer if all flow variables are integer variables. We would rather require all flow variables to be integer than all single point of failure variables to be integer, because the number of single point of failures is much larger than the number of edges, and there are violation variables for all combinations of subproblems, while there is only one flow variable for every arc for every subproblem. The difference becomes especially large when there are many routes, such as in a network dark fiber or dark fiber ring.

Modeling the single point of failures as violation constraints allows one route to use two arcs which form a single point of failure, solving the issue of Figure 4.9. It also allows the constraint to be a soft constraint, which is what we want. The number of integer variables stays acceptable because we only have to constrain every flow variable to be integer, and not every single point of failure violation variable. That is why we choose this method.

4.4 A simple heuristic

Before trying to solve the problem to optimality, we introduce a simple heuristic to find a solution to the problem, based on how the problem is currently solved by hand. The simple heuristic works by performing the following steps:

1. Modify the graph of the network, multiplying the length of all virtual edges, i.e. edges that belong to virtual cables, by C_{virt} .¹
2. Pick a problem
 - (a) Pick a subproblem within this problem
 - (b) Use a shortest path algorithm to find a solution to the subproblem. Do not use edges with insufficient capacity. Do not create routes that are longer than 100 kilometers (see below). If there is no path shorter than or equal to 100 kilometers, do not solve this problem, and continue to step 3.
 - (c) Update the graph of the network. For all edges that would form a single point of failure with the path that was just found, add $s_{length} * C_{spof}$ to the length of the edge, where s is the possible single point of failure. Subtract the capacity of all the used edges by the number of fibers for this subproblem.

¹See Section 4.2 on why a penalty is added for virtual cables.

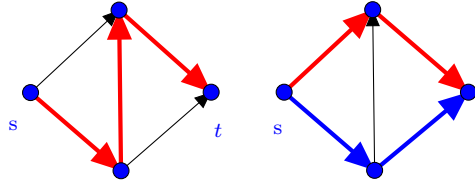


Figure 4.11: Suppose the red path on the left is the shortest path for a subproblem $(s, t, 1)$ in a network with unit capacities. If we solve a problem with two such subproblems on this network using the simple heuristic from Section 4.4, this problem will remain unsolved. However, there is a feasible solution to this problem, shown on the right.

- (d) Return to step 2a, and continue until all subproblems of this problem are solved
3. Undo the changes to the lengths of the edges made during the solving of the last problem (step 2), but keep the changes to the capacity of the edges (step 1)
4. Return to step 2, and continue until all problems in the problem set are solved

This heuristic runs in polynomial time, while the problem is NP-hard (see Section 3.1). Note that if a problem cannot be solved using this heuristic, this does not necessarily mean that it is impossible to solve the problem. See Figure 4.11.

Dijkstra's algorithm [4] can easily be modified to also remember the actual, non-adjusted distances to nodes. Then, simply do not expand edges that would lead to routes that are too long. This both speeds up the algorithm and makes sure none of the found routes violates the route length constraint.

4.5 ILP formulation

The examination of different methods to model the single point of failure constraint brings us to the following integer linear programming formulation for solving the entire main problem, in which, if the flow variables are integer, all variables are integer:

Parameters

- $G = (V, A)$
Directed graph induced by $G = (V, E)$ where $A = \{(u, v), (v, u) \mid \{u, v\} \in E\}$
- S_{dir}
The set of directed single point of failures, $S_{dir} = \{\{(u, v), (w, x)\} \mid (u, v) \in A \wedge (w, x) \in A \wedge \{\{u, v\}, \{w, x\}\} \in S\}$

- s_{length}
The length of single point of failure $s \in S$
- $free(u, v)$
The number of free fibers, e_{free} , in cable corresponding to edge $e = \{u, v\} \in E$
- $length(u, v)$
The length of a cable, e_{length} , corresponding to edge $e = \{u, v\} \in E$
- R
The set of subproblems to be solved
- PS
The set of problems $P \in PS, P \subseteq R$
- s_r, t_r
Source $s_r \in A$ and sink $t_r \in A$, i.e. endpoints, of subproblem $r \in R$
- $fiber_r$
The number of required fibers for subproblem $r \in R$
- $A_{virt} \subseteq A$
All arcs $(u, v) \in A_{virt}$ for which there is an $\{u, v\} \in E_{virt}$, i.e. all arcs which represent virtual cables
- C_{spof}
The penalty for a single point of failure, per meter
- $C_{unsolved}$
The penalty for an unsolved problem
- C_{virt}
The penalty for using a virtual cable, per meter

Variables

$$f_r(u, v) = \begin{cases} 1 & \text{if arc } (u, v) \text{ is used in subproblem } r \in R \\ 0 & \text{otherwise} \end{cases} \quad (4.9)$$

$$sv_{r,q}(s) = \begin{cases} 1 & \text{if spof } s \text{ is violated between subproblems } \{r, q\} \subseteq R \\ 0 & \text{otherwise} \end{cases} \quad (4.10)$$

$$pu(P) = \begin{cases} 1 & \text{if problem } P \subseteq R \text{ is unsolved} \\ 0 & \text{otherwise} \end{cases} \quad (4.11)$$

Objective

$$\begin{aligned} \min & \sum_{r \in R} \sum_{(u,v) \in A \setminus A_{virt}} length(u,v) * f_r(u,v) \\ & + C_{virt} * \sum_{r \in R} \sum_{(u,v) \in A_{virt}} length(u,v) * f_r(u,v) \\ & + C_{spof} * \sum_{\{r,q\} \subseteq R} \sum_{s \in S} slength * sv_{r,q}(s) \\ & + C_{unsolved} * \sum_{P \in PS} pu(P) \end{aligned} \quad (4.12)$$

Constraints

- Conservation

$$\begin{aligned} \sum_{(u,v) \in A} f_r(u,v) &= \sum_{(v,w) \in A} f_r(v,w) \\ \forall v \in V \setminus \{s_r, t_r\} \quad \forall r \in R \end{aligned} \quad (4.13)$$

- Required flow sink and source

$$\begin{aligned} \sum_{(u,s_r) \in A} f_r(u,s_r) - \sum_{(s_r,w) \in A} f_r(s_r,w) - pu(P) &= -1 \\ \forall r \in R, r \in P \end{aligned} \quad (4.14)$$

$$\begin{aligned} \sum_{(u,t_r) \in A} f_r(u,t_r) - \sum_{(t_r,w) \in A} f_r(t_r,w) + pu(P) &= 1 \\ \forall r \in R, r \in P \end{aligned} \quad (4.15)$$

- Availability

$$\begin{aligned} \sum_{r \in R} (fiber_r * (f_r(u,v) + f_r(v,u))) &\leq free(u,v) \\ \forall (u,v) \in A \end{aligned} \quad (4.16)$$

- Route length

$$\sum_{(u,v) \in A} f_r(u,v) * length(u,v) \leq 100000 \quad (4.17)$$

$$\forall r \in R$$

- Single point of failure

$$f_r(u,v) + f_q(w,x) - sv_{r,q}(s) \leq 1 \quad (4.18)$$

$$\forall \{r,q\} \subseteq P \quad \forall P \in PS \quad \forall s = \{(u,v), (w,x)\} \in S_{dir}$$

Then, every edge $\{u,v\} \in E$ for which $f_r(u,v) = 1$ or $f_r(v,u) = 1$ is used in route r .

4.6 Solving the ILP

The problem is still far too large to efficiently solve as a mixed integer linear programming problem, all flow variables integer, with a simple branch and bound solver.

The first problem is that the memory requirement of branch and bound is relatively high for a large number of integer variables. To reduce this, we use a cutting-plane method based step to the solver, effectively turning it into a variant of a branch and cut solver. Only variables which have not always been integer in earlier solutions are branched on. This reduces the number of integer variables considerably, because the vast majority of edges will never have any flow at all, and thus will never be branched on.

The second problem is that due to the high number of single point of failure constraints, the number of violation variables is also very high. Even though these do not need to be integer, it still slows down solving the LP relaxations considerably. Like the flow variables, the vast majority of single point of failure violation variables will also be zero. Therefore, we introduce another step based on the cutting-plane method, such that only single point of failure constraints and variables which have been violated in previous solutions are considered. We mark these single point of failures as *important* single point of failures. One starts with an initial set of important single point of failures, which could be empty, and solves the ILP with only these single point of failure constraints. Then, the resulting set of solutions is verified against the complete set of single point of failures, and the violated single point of failures are added until there is no single point of failure left which is not added to the ILP model.

All important single point of failures were on a previously considered feasible route. When there are no unconsidered violated constraints left, the solution is guaranteed to be optimal, because the relaxation is a lower bound on the optimal solution, and the optimal solution is the solution to the most constrained problem.

Some MILP solvers support branch and cut algorithms. In these cases, it is not necessary to explicitly use the cutting-plane based method on the flow variables, as the solver

will do this already. Incrementally adding single point of failures, however, reduces the time and space requirements of the LP relaxation, and is therefore useful even when using branch and cut solvers.

Combining these two solutions, we come to Algorithm 5.

Algorithm 5 Cutting-plane method based ILP solver function for the problem. For smaller problems or MILP solvers which already use cutting planes, the steps to find an integer solution may be omitted, i.e. $integerEdges \leftarrow E$. $integerEdges$ are edges which are constrained to have integer flow. $importantSpofs$ is the initial set of important single point of failures, which is usually empty.

```

1: function SOLVE( $G = (V, E)$ ,  $S$ ,  $integerEdges \subseteq E$ ,  $importantSpofs \subseteq S$ )
2:    $solution \leftarrow$  BRANCHANDBOUNDSOLVER( $G = (V, E)$ ,  $integerEdges$ ,  $importantSpofs$ )
3:   if  $solution$  is infeasible then
4:     return infeasible
5:      $\triangleright$  The relaxation is a lower bound to the main problem, so if the relaxation is
       infeasible, the main problem is infeasible.
6:   end if
7:   if  $solution$  contains non-integer edges then
8:      $integerEdges \leftarrow integerEdges \cup$  non-integer edges in  $solution$ 
9:     return SOLVE( $G = (V, E)$ ,  $S$ ,  $integerEdges$ ,  $importantSpofs$ )
10:  end if
        $\triangleright$  At this point the solution is integer and feasible
11:  if  $solution$  contains violated single point of failures in  $S \setminus importantSpofs$  then
12:     $importantSpofs \leftarrow importantSpofs \cup$  violated single point of failures in  $solution$ 
13:    return SOLVE( $G = (V, E)$ ,  $S$ ,  $integerEdges$ ,  $importantSpofs$ )
14:  end if
        $\triangleright$  At this point the solution is guaranteed to be optimal
       return  $solution$ 
15: end function

```

4.7 A more complex heuristic

Using the method to solve the ILP from Section 4.6, we can create a more advanced heuristic:

1. Pick a problem
2. Solve only this problem to optimality using the method to solve the ILP from Section 4.6
3. Update the graph by subtracting all used fibers in the solution from the capacities
4. Return to step 1, and continue until all problems are solved

The only way in which problems are dependent on each other, is by the capacity constraint. If there were no capacity constraint, this heuristic would always find optimal solutions. Therefore, if the cables in the optimal solution have sufficient capacity, or the optimal solutions to the individual problems do not share any cables, this heuristic will also find the optimal solution. This is often the case. Even if this is not the case, often only a few cables have a low capacity, so the solution from the heuristic will still be good.

A downside is that, just as with the simple heuristic, it could happen that this heuristic cannot find a solution to a problem, even though one does exist. Another downside is that there is no known algorithm to compute this heuristic in polynomial time, because solving just one problem to optimality is already NP-hard (see Section 3.1.2). Therefore, in some cases, this heuristic could take a very long time to complete. However, the scale of the problem is being reduced significantly, and it will still be faster than solving the entire problem set at once.

4.8 Preprocessing

Even with the cutting-plane methods, the number of variables is still too large to solve the problem efficiently. This is why a number of filtering preprocessing steps are introduced to reduce the size of the network.

4.8.1 Preprocessing step based on shortest path distance (SP)

Because of the constraint that routes cannot be longer than 100 kilometers, nodes which can never be reached within that distance can be left out. A naive implementation of this would be to filter out all nodes that are more than 100 kilometers away from the start- and endpoint. Unfortunately, if the start- and endpoint are close to each other, this removes very few nodes. A better solution is to only consider nodes of which the sum of the shortest distance to both endpoints is less than 100. This will not change the optimal solution. See Figure 4.12.

This gives a good improvement on single subproblems, but if we have multiple subproblems which are spread over different distant locations, the number of cables that can be removed is much smaller, and flow variables for every subproblem will still be created, while in some subproblems, these cables can never be used. Therefore, the network $G = (V, E)$ is being split into several subsets $G_r \subseteq G = (V_r \subseteq V, E_r \subseteq E)$ of relevant parts of the network to subproblem $r \in R$. Similarly, the set of single point of failures S is also being split into subsets $S_r \subseteq S$ of relevant single point of failures to a certain some problem, i.e. single point of failures of which both edges are in E_r . Then, when constructing the ILP solver, we only create variables $f_r(u, v)$ and $f_r(v, u)$ of which $\{u, v\} \in E_r$, and $sv_{r,q}(s)$ of which $s \in S_r \wedge s \in S_q$.

This leads us to Algorithm 6.

Algorithm 6 Pseudocode for the preprocessing step based on the shortest path distance between a node in the network and the endpoints of a subproblem. The SINGLE-SOURCESHORTESTPATH algorithm only has to expand until 100 kilometers. The optimal solution to the problem after the preprocessing step is guaranteed to be equal to the optimal solution to the problem before the preprocessing step. See Figure 4.12.

```

1: function SHORTESTPATHFILTER( $G = (V, E), S, R$ )
2:   for all  $r \in R$  do
3:      $distanceToS \leftarrow$  SINGLESOURCESHORTESTPATH( $s_r, G_r$ )
4:      $distanceToT \leftarrow$  SINGLESOURCESHORTESTPATH( $t_r, G_r$ )
5:      $V_r, E_r, S_r \leftarrow \emptyset$ 
6:     for all  $v \in V$  do
7:       if  $distanceToS(v) + distanceToT(v) > 100$  km then
8:          $V_r \leftarrow V_r \setminus \{v\}$ 
9:         for all  $u \in V$  do
10:           $E_r \leftarrow E_r \setminus \{u, v\}$ 
11:          for all  $e_2 \in E$  do
12:             $S_r \leftarrow S_r \setminus \{\{u, v\}, e_2\}$ 
13:          end for
14:        end for
15:      end if
16:    end for
17:  end for
18: end function

```

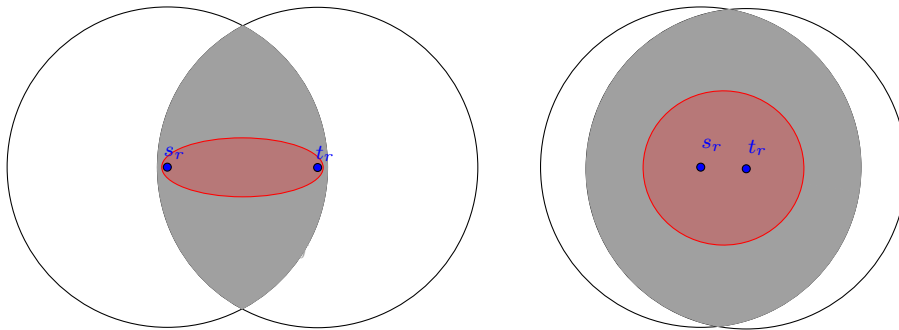


Figure 4.12: Consider subproblem r between locations s_r and t_r . The circles around the locations represent all nodes within a distance of 100 kilometers. Naive interpretations of a shortest path distance filter would only consider the white or grey area, but only the nodes within the red elliptical area with a major axis of 50 need to be considered.

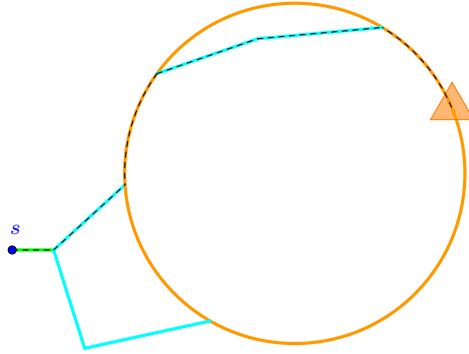


Figure 4.13: Suppose the green cable is a TAN cable, the blue cables are SAN cables, and the orange circle is a ring of PAN cables. The shortest path from s to the triangle is the dashed path, which descends in hierarchy after the first PAN section, and then ascends again. While this is a valid path, and it is strictly speaking the best path, eliminating the second SAN section on the path and instead staying on the PAN ring is more logical and therefore acceptable too.

4.8.2 Preprocessing step based on availability (FF)

If a subproblem requires more fibers than are free in a certain cable, this cable can never be used in a route for this subproblem, and it can therefore be left out without a change in the optimal solution.

4.8.3 Preprocessing step based on hierarchy (HF)

While the algorithm with only optimality-preserving preprocessing steps is fast enough for single dark fibers, even one dual dark fiber will use a large amount of resources. Also, the route you end up with may be optimal in the strict sense, but the optimal route can be very impractical and complicated if it is a route that ascends and descends hierarchy several times. TAN cables are of the lowest hierarchy, then SAN cables, then PAN cables, and finally INN cables are of the highest hierarchy. For example, a connection in a solution may start at a SAN cable, then ascend in hierarchy to a PAN cable, and then use a shortcut over another SAN cable to arrive back at the same PAN cable, see Figure 4.13. While this is a better solution than staying at the PAN cable, it is also unpractical, so using a worse but more practical solution instead would be acceptable too. For these reasons, we introduce a preprocessing step based on the hierarchy of cables.

The idea behind this step is to only include cables which are reachable from one of the end locations through a lower hierarchy. We start with the two end locations of the subproblem, and visit all cables which are of the same hierarchy as the locations, and are directly reachable from the locations. In the example of Figure 4.13, looking only at the

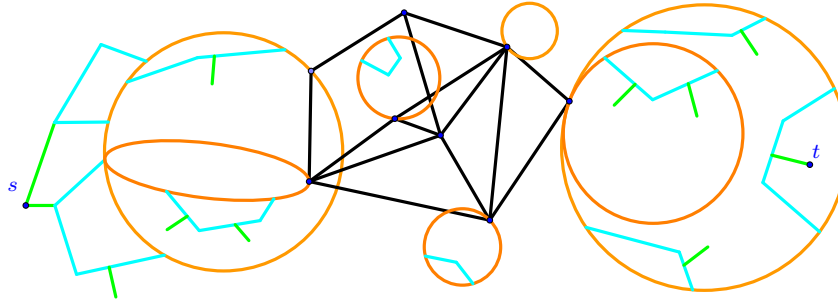


Figure 4.14: Example of a network without preprocessing based on hierarchy applied. The green cables are TAN cables, the blue cables are SAN cables, the orange cables are PAN cables, and the black cables are INN cables.

expansion starting at s , this would mean that the TAN cable would be visited. Then all cables which are of the hierarchy directly above, and are directly reachable from earlier visited nodes are visited. In the example of Figure 4.13, the two SAN sections that are connected to the TAN cable are visited, but the SAN section that is a shortcut to the PAN ring is not visited, because it is not directly connected to the TAN cable that was visited in the previous step. These steps are continued until the maximum hierarchy is reached. On the maximum hierarchy level, all reachable cables are allowed to be visited. See Algorithm 7 and Figure 4.15. At the end, all cables which were not visited are removed from the problem. The hierarchy of a location is defined to be the maximum hierarchy of all cables connected to it.

This step preserves optimality for some instances, but not for all instances. In many cases, only cables which are not part of the optimal solution are removed by the preprocessing step based on hierarchy. In these cases, optimality is preserved. However, consider the example from Figure 4.13 again. In this case, the optimal route before the preprocessing step based on hierarchy is not the same as after the step, so optimality is not preserved. Fortunately, the instances for which optimality is not preserved, must also have been instances for which the optimal solution had illogical hierarchy changes.

Note that the order in which the preprocessing steps are applied is relevant to the outcome. If certain cables are filtered out by one preprocessing step, the shortest path distances can change, and some locations might become disconnected from the network due to the removal of cables. Because of this, the outputs of the `SINGLESOURCESHORTESTPATH` and `DISCOVER` functions can be different for different subproblems. This makes it impossible to speed up the preprocessing steps by saving the results of these functions.

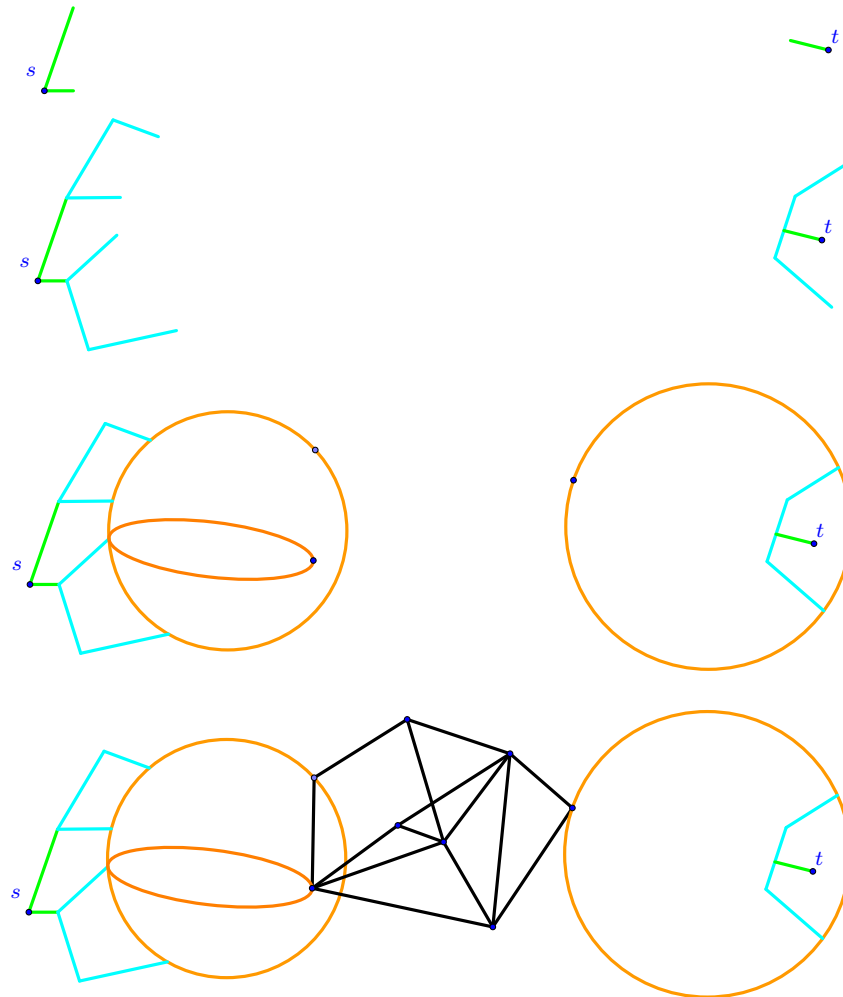


Figure 4.15: Example of the preprocessing step based on hierarchy being applied to the example network from Figure 4.14. The SAN cables that are not connected to the three relevant TAN cables are removed because they would need a descent in hierarchy to be reachable. The smaller PAN ring on the right is removed because it is not directly connected to one of the SAN cables connected to t . The INN network is completely preserved because it is the top hierarchy, but all underlying cables of lower hierarchy are removed. In reality, there are many such cables of lower hierarchy connected to the INN cables. The vast majority of these cables will most likely not be used in a connection from s to t , and this preprocessing step removes them.

Algorithm 7 Pseudocode for the preprocessing step based on hierarchy. The optimal solution to the problem after this preprocessing step is not necessarily equal to the optimal solution to the problem before the preprocessing step.

```

1: function HIERARCHYFILTER( $G = (V, E), S$ )
2:   for all  $r \in R$  do
3:      $E_r \leftarrow \text{DISCOVER}(s_r, G_r) \cup \text{DISCOVER}(t_r, G_r)$ 
4:      $V_r \leftarrow \bigcup \{u, v\} \forall \{u, v\} \in E_r$ 
5:      $S_r \leftarrow \bigcup \{\{e_1, e_2\}\} \forall e_1 \in E_r, e_2 \in E_r, \{e_1, e_2\} \in S$ 
6:   end for
7: end function
8: function DISCOVER( $source, G_r = (V_r, E_r)$ )
9:    $Q \leftarrow$  FIFO queue
10:   $Q.\text{ENQUEUE}(source)$ 
11:   $discovered \leftarrow \emptyset$ 
12:   $discoveredThisHierarchy \leftarrow \emptyset$ 
13:   $hierarchyCables \leftarrow \emptyset$ 
14:   $expanded \leftarrow \emptyset$ 
15:   $currentHierarchy \leftarrow$  maximum hierarchy of cables connected to  $source$ 
16:   $hierarchyCables \leftarrow \{e = (source, v) \mid e \in E_r\}$ 
17:  while  $currentHierarchy \neq maximumHierarchy$  do
18:    while  $Q \neq \emptyset$  do
19:       $u \leftarrow Q.\text{DEQUEUE}$ 
20:      if  $u \notin discoveredThisHierarchy$  then
21:         $discoveredThisHierarchy \leftarrow discoveredThisHierarchy \cup \{u\}$ 
22:         $discovered \leftarrow discovered \cup \{u\}$ 
23:        for all  $e = \{u, v\} \in E_r$  do
24:          if  $e_{hierarchy} = currentHierarchy \wedge currentHierarchy =$ 
 $maximumHierarchy \vee e \in hierarchyCables$  then
25:             $expanded \leftarrow expanded \cup \{e\}$ 
26:             $Q.\text{ENQUEUE}(v)$ 
27:          end if
28:        end for
29:      end if
30:    end while
31:    INCREMENT( $currentHierarchy$ )
32:     $Q \leftarrow Q \cup discovered$ 
33:     $discoveredThisHierarchy \leftarrow \emptyset$ 
34:     $hierarchyCables \leftarrow \{e = (u, v) \mid e \in E_r \wedge u \in discoveredThisHierarchy\}$ 
35:  end while
36:  return  $expanded$ 
37: end function

```

4.8.4 Preprocessing step based on straight line distance (SL)

There could be a high number of cables, which could make preprocessing take a long time if there are many subproblems. Even when using Dijkstra's algorithm with a priority queue, the preprocessing step based on shortest path distance still takes $O(m + n \log n)$ time per subproblem [8], where m is the number of edges and n is the number of nodes.

This is why the preprocessing step based on straight line distance is introduced. This is an optimality preserving preprocessing step that can be used before any of the other preprocessing steps, to speed up the subsequent preprocessing steps.

This preprocessing step works in exactly the same way as the preprocessing step based on shortest path distance, except that it simply uses the Euclidean distance between two nodes instead of the shortest path distance. This distance is always lower than or equal to the shortest path distance. Every node that will not be removed by the preprocessing step based on shortest path distance, will also not be removed by this step. Hence, this step is also optimality preserving.

The Euclidean distance between the end locations of a subproblem and every node in the network needs to be determined. Since calculating the Euclidean distance between two points takes constant time, this preprocessing step can be performed in $O(n)$ time per subproblem.

4.8.5 Chaining preprocessing steps

As was mentioned earlier, the order in which preprocessing steps are applied influences the outcome. In addition, it influences the time preprocessing takes. The place of a preprocessing step in the order needs to depend on the balance between time the preprocessing step takes and the reduction of the network it accomplishes.

It makes sense to apply the preprocessing step based on straight line distance first, because this is a fast preprocessing step that has the potential to reduce the size of the network greatly. It also makes sense to apply the preprocessing step based on shortest path distance last, because this is a relatively slow step that would run faster on a small network. It also accomplishes a greater reduction if preprocessing steps have been applied before this step, because other preprocessing steps may have left unreachable edges or nodes, which this step can then remove.

Hereafter, each preprocessing step will be referred to by its abbreviation: SP for shortest path distance, FF for availability, HF for hierarchy, and SL for straight line distance. Successive application of preprocessing steps will be denoted by an arrow \rightarrow . For example, $SL \rightarrow HF \rightarrow FF \rightarrow SP$ means that first SL preprocessing was applied, then HF, then FF and finally SP preprocessing.

4.9 Solving with local search

For larger problems with multiple subproblems it might not be possible to solve the problem (almost) optimally with the integer linear programming algorithm. For such cases, we use a local search based algorithm. This is not a big issue, because what is an optimal solution to such a problem now might not be the optimal solution later. When new orders are placed, the capacities of the cables in the network change, and therefore the optimal solution changes as well.

4.9.1 Mutation step

The most difficult part of the local search algorithm is finding a good mutation step. Because the problem is not a permutation problem, heuristics for network optimization such as Lin-Kernighan [9] will not be efficient. The problem could be represented as a bitstring problem, but the number of strings that lead to an infeasible solution is so large, that the simplest mutation steps will only generate infeasible neighbors.

Therefore we use a mutation step inspired by a method used in large neighborhood local search for vehicle routing problems [10]. The original large neighborhood local search uses a *destroy* step in which part of the solution is destroyed, and then a *repair* step which recreates this part of the solution. This results in a much larger neighborhood space, which prevents the algorithm from getting stuck in local optima too soon.

In our case, this means destroying part of a route in a current solution, and then repairing it by computing a new, different path for the destroyed part.

Destroy

The *destroy* step takes two points on a route in the solution, and removes all edges in between these points. The route to mutate is chosen randomly. However, if the two points are also chosen randomly, it will result in an unequal distribution of removed edges. The edges in the middle of the paths will be removed much more often than those at the beginning or end.

One way to fix this would be to change the mutation step such that always the entire route is being removed, or such that one of the points is always the start or end of a route. However, this means that small mutations will not occur very often, while small mutations retain a lot of the information about a solution. It also means that the repair step will take a longer time because the destroyed part of the route will be longer on average.

For these reasons, one of the points is chosen randomly, splitting the route into two parts. Then, the probability that the second point is chosen from the shorter part is made higher than the probability that the second point is chosen from the longer part. This will make sure every edge has a more or less equal probability of being removed. It will also make the probability of a small part of the route being removed higher, so we can have more small incremental mutations.

The steps for destroying part of a route can be found in Algorithm 8.

Algorithm 8 Destroys part of the route. Using this algorithm gives each edge a more or less equal probability of being removed, and has a higher probability of destroying short parts.

```

1: function DESTROY(Solution = (Routes, Unsolved), G = (V, E))
2:    $r \leftarrow \text{RANDOMCHOICE}(R)$  ▷ Pick random route from solution
3:    $e_1 \leftarrow \text{RANDOMCHOICE}(Route_r)$  ▷ Pick random edge from route
4:   if  $|Route_r| = 1$  then
5:      $e_2 \leftarrow e_1$ 
6:   else if  $\text{RANDOM}(0,1) < \text{index\_on\_Route}_r(e_1)/(|Route_r| - 1)$  then
7:     ▷ Pick random edge from second part
8:      $e_2 \leftarrow \text{RANDOMCHOICE}(Route_r[e_1\dots])$ 
9:   else
10:    ▷ Pick random edge from first part
11:     $e_2 \leftarrow \text{RANDOMCHOICE}(Route_r[...e_1])$ 
12:   end if
13:    $RoutePart \leftarrow Route_r[e_1\dots e_2]$  ▷ Destroy part of the route
14:    $Route_r \leftarrow Route_r \setminus RoutePart$ 
15: end function

```

Repair

A new path must be found for the destroyed part of the route. Since pathfinding makes for a relatively slow repair step, and the number of simple paths between two nodes can be exponentially large, we do not want to completely randomly search the neighborhood. We want to bias the search towards good neighbors.

The route is reconstructed in the same way as as happens with the simple heuristic (Section 4.4). Before each mutation, the lengths of the edges in the graph are adjusted. The lengths of virtual cables are multiplied by C_{virt} , and the lengths of the edges that form single point of failures with other edges in the solution are multiplied by C_{spof} . Then a shortest path is created with these adjusted lengths. This will bias the mutation towards better solutions. Using this repair step, not many of the resulting solutions will be infeasible, because if a path exists, it will be found, unless the cable has reached its capacity, or only paths that are too long are possible.

Randomness

Having fully random routes will not work well because there are too many options, but having fully biased routes will not work well either, because then the algorithm will repeatedly arrive at the same solutions. A balance needs to be found. This is why we introduce a *randomness* factor. The following equation is used for edge lengths within the *repair* step of the mutation:

$$e_{repair_length} = x^{randomness/maxrand} * e_{adjusted_length}^{(1-randomness)/maxrand} \quad (4.19)$$

Where x is a random number between 0 and 1, $maxrand = \max\{randomness, 1 - randomness\}$, and $e_{adjusted_length}$ is the length of an edge $e \in E$ adjusted for single point of failures and virtual cables, as described earlier. $randomness$ is the randomness factor between 0 and 1. From the formula it follows that if $randomness = 0$, then the length used in the algorithm is simply the adjusted length of the edge, and if $randomness = 1$, then the length used in the algorithm is fully random.

The mutation step is then incorporated into a simulated annealing [11, 12] algorithm.

4.9.2 Simulated annealing

Parameters

- Acceptance probability

Using probabiltiy acceptance function as explained by Ingber[13]:

$$p = \begin{cases} 1 & \text{if the new solution is better than the current solution} \\ \frac{1}{1+\exp(\frac{\Delta}{T})} & \text{otherwise} \end{cases}$$

Where Δ is the difference in score (energy) between the new and the current solution.

- T_0

The acceptance probability depends on the value of T and the difference between the score of the new and current solution Δ . If T is a small number, while Δ is a high number, the acceptance probability will be very small. The magntiude of the value of Δ can vary a lot between mutations, because a small change can bring several new single point of failures into the solution and increase the score immensely. In the beginning, we want the acceptance probability to be high, so the initial value of T , T_0 , should be a number that is high enough to result in high initial acceptance probabilities (around 0.5), yet not so high that the algorithm takes too long to reach lower probabilities.

- Cooling schedule: α and Q

Every Q mutations, T is multiplied by a number α to decrease it, to gradually decrease the acceptance probability.

Larger problem sets, i.e. sets with more subproblems, take longer to solve manually and should be allowed a longer time to compute automatically than smaller problem sets. In addition, a solution with a larger neighborhood should explore more neighbors

before converging to arrive at a solution of comparable quality. A problem set with more subproblems also has a larger neighborhood.

Therefore, the cooling schedule is defined in such a way that for most instances, it does not take longer than one minute per subproblem² to complete. Note that we do still solve all subproblems at once, not one by one. α is a constant and Q is a constant multiplied by the number of subproblems.

- Stopping criterion

The algorithm terminates when the temperature T is so low that the acceptance probability is close to 0. At this point, the algorithm acts like a hill climbing algorithm.

4.10 Combining solvers

Now we have four different ways of solving a set of problems. The first is the simple heuristic, which simplifies the problem into a polynomial time problem. This makes the simple heuristic run very fast, but also have very poor worst-case results.

The second way is the ILP solver. If this solves the set of problems, it is guaranteed to solve it to optimality. This makes the ILP solver very suitable for relatively small problems, which can be solved fairly quickly. Unfortunately, if the set of problems is large, it could take a very long time and it may not be able to solve it at all.

We also have a more complex heuristic, which simply considers each problem of the set separately. If the problems are not too dependent on each other, this will most likely lead to excellent results. A disadvantage is that this heuristic can potentially take a long time to complete. Another disadvantage is that this heuristic also does not have a theoretical bound to the quality of the solution, so the worst-case results may be very poor.

The local search solver will always find solutions, and it will always find them within a reasonable time. However, the results may be far from optimal, and there is no way of verifying this without knowing the optimal solution.

4.10.1 Providing the local search solver with initial solutions

Obviously, the local search solver can benefit greatly from a good initial solution. Since we have three other ways of finding a good solution, the solutions from these solvers can be used as the initial solution to the local search solver.

Even the ILP solver gives intermediate results which can be used. For example, we could try solving the problem to optimality with the ILP solver first. If that fails or takes too long, we can switch to the local search solver. The best known intermediate result from the ILP solver can then be used as the initial solution to the local search solver.

²Experimentally tested on a set of instances, see Chapter 5

4.10.2 Providing the ILP solver with initial solutions

For the ILP solver there is also much to be gained by providing it with some information about the solution. If the ILP solver knows a certain good solution in the beginning, this can be a good indicator of which single point of failures are relevant to the optimal solution. Then the list of important single point of failures can be initialized with these single point of failures.

Using the solution from the complex heuristic will work even better. The complex heuristic has already gone through the process of finding out which single point of failures are important to which problem, and the important single point of failures to the entire problem set will probably have a very high overlap with this set. So if we simply remember which single point of failures were important during the calculation of the complex heuristic, this can be used as input to the ILP solver to considerably speed up the algorithm, while still solving it to optimality.

4.10.3 Combining all solvers

Since the complex heuristic gives better solutions than the simple heuristic, it makes sense to try the complex heuristic first to try to find an initial solution. However, because the complex heuristic still solves an NP-hard problem, it can fail or take a long time. If this happens with one of the problems, that problem can be solved quickly with the simple heuristic. Then the rest of the problems in the problem set can still be solved with the complex heuristic.

If all problems were quickly solved using just the complex heuristic, we can try to solve the problem to optimality with the ILP solver. If this fails or takes too long, we take the best known solution up until that point and use it as input to the local search solver.

If one or more of the problems could not be solved quickly using the complex heuristic, and was solved using the simple heuristic instead, it makes no sense to use the ILP solver, because it will most likely take at least as long. In these cases, we continue with the local search solver immediately, using the combined result of the complex and simple heuristics as the initial solution.

Using these rules, we will find a solution that is guaranteed to be optimal if it can be found quickly, and if it cannot be found quickly, we will still get a solution from the local search algorithm. A flow chart of the rules can be found in Figure 4.16.

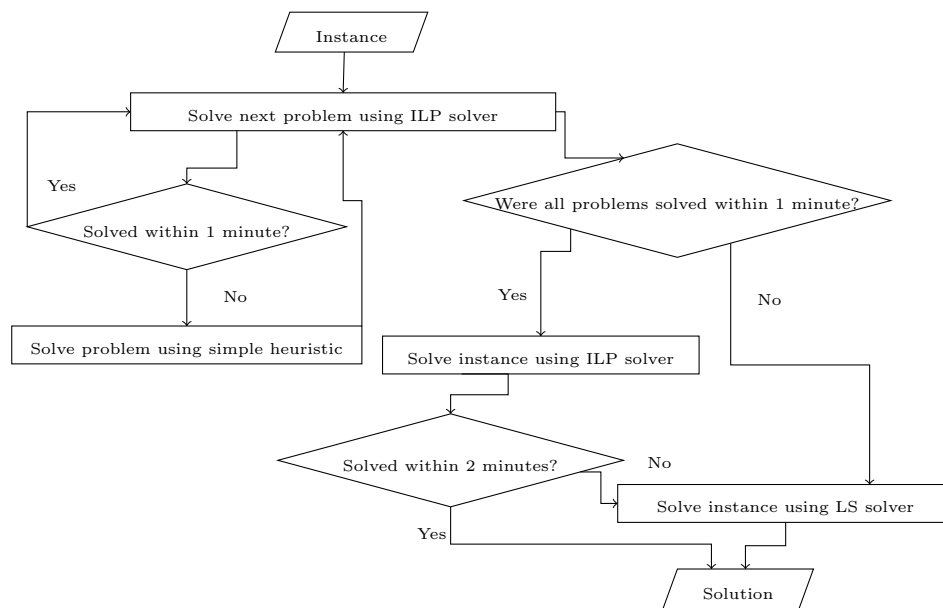


Figure 4.16: Flow chart of the combined solver. The upper left part represents the combined heuristic.

Chapter 5

Experiments

To find out and show the practical performance of the algorithms that were introduced, a number of experiments were conducted.

All experiments were conducted on an Intel Core i5 520UM processor with 4GB RAM. Approximately 3.5GB RAM was available to the experiments. The MILP solver that was used is Gurobi 6.0.4 [14]. The algorithms were written in C# 5.0, and compiled with Mono 4.0.3.

All times that are mentioned for the algorithms are excluding the time preprocessing takes. The time preprocessing takes will be discussed separately (Section 5.2 and Table 5.4). The results of all experiments will be discussed in Chapter 6.

5.1 Instances

All experiments were conducted on the same set of instances. Some of these were real orders, and some were artificially created to have interesting properties. The set of instances that the experiments were conducted on was chosen to be diverse, with each instance having its own unique properties. See Table 5.1 for the number of problems, the number of subproblems, and the optimal values for these instances after full preprocessing. See Appendix B for details on the instances, and Appendix C for more more detailed results of some of the experiments. The names of the instances have a pattern to indicate some of their properties. The first character indicates related instances. Instances with the same first character are related (details on how they are related can be found in Appendix B). After the first character, the number of problems in the instance follows, and then the number of subproblems in the instance. Instances with an "-INN" suffix only contain main hubs as destinations.

Anonymized versions of the entire network after adding virtual cables and preprocessing for all instances are available at <http://www.students.science.uu.nl/~3698246/thesis/instances.zip>.

Name	# Problems	# Subproblems	Opt
S1P1SP	1	1	88930.41
T1P1SP	1	1	10183.82
L1P2SP	1	2	956668.00
D1P2SP	1	2	27142790.82
H1P2SP-INN	1	2	15966
K1P2SP-INN	1	2	21984
F2P4SP	2	4	74198.87
B1P2SP	1	2	414805914.93
C1P2SP	1	2	497221875.23
D38P38SP	38	38	60003493745.49
D38P38SP-INN	38	38	30000862993.70
A14P14SP	14	14	20001205040.53
A14P14SP20	14	14	30000938037.37
A14P28SP	14	28	22444258457.01
J14P28SP8-INN	14	28	64214584040.96
D38P76SP	38	76	70890333523.64
D38P76SP-INN	38	76	41416301981.57
I10P20SP	10	20	7378557.13 - 7382948.01
G4P14SP-INN	4	14	416203.60
G4P14SP	4	14	2928273.08 - 4160358.57
G3P8SP	3	8	2469169.05
G1P4SP	1	4	1801837.70

Table 5.1: All the instances that will be used for the experiments. Optima are after full preprocessing (SL \rightarrow HF \rightarrow FF \rightarrow SP).

5.2 Preprocessing

A number of experiments were conducted to examine the effectiveness of the preprocessing steps that were introduced in Section 4.8. For all shortest path preprocessing, Dijkstra’s algorithm was used as the single source shortest path algorithm. We make a distinction between *optimality preserving preprocessing* (SL→FF→SP) and *full preprocessing* (SL→HF→FF→SP).

Optimality preserving preprocessing

Table 5.2 shows how the magnitude of the network is reduced by optimality preserving preprocessing. In almost all cases, using optimality preserving preprocessing reduces the network drastically. The only exceptions are D38P38SP, D38P38SP-INN, D38P76SP, and D38P76SP-INN.

Preprocessing based on hierarchy

When the preprocessing step based on hierarchy is added, the optimal values start to change. See Table 5.3. In most cases, the additional decrease in the size of the network is very large, while the optimal value increases marginally or not at all. The most striking exception to this is C1P2SP. The optimal value for this problem more than doubles. For the INN problem sets, the increase is also larger.

What is interesting to see is the enormous decrease in size for the INN problem sets. Compared to the problem sets they were derived from, many more cables are being removed, while, when using only optimality preserving preprocessing, the difference between the network sizes for the original problem sets and the INN derivatives were negligible.

Another interesting case is D38P38SP. In this case, preprocessing based on hierarchy resulted in one of the subproblems being infeasible, as becomes clear from Table C.1 in Appendix C.

Straight line distance preprocessing

Using preprocessing based on straight line distance significantly reduces the time that optimality preserving preprocessing takes. See Table 5.4. In all cases, preprocessing takes only a few seconds for small instances to a few minutes for large instances. Preprocessing without the straight line distance step even fail on J14P28SP8-INN and D38P76SP due to insufficient memory.

Instance	# nodes	# cables	#spofs
S1P1SP	7330 (-94%)	15202 (-94%)	0
T1P1SP	15485 (-87%)	35291 (-85%)	0
L1P2SP	2630 (-98%)	5191 (-98%)	43387 (-98%)
D1P2SP	1941 (-98%)	3870 (-98%)	32054 (-99%)
H1P2SP-INN	18057 (-85%)	40317 (-83%)	442338 (-84%)
K1P2SP-INN	17643 (-85%)	39486 (-84%)	439817 (-84%)
F2P4SP	19846 (-84%)	44809 (-81%)	448158 (-84%)
B1P2SP	3469 (-97%)	7123 (-97%)	62224 (-98%)
C1P2SP	5558 (-95%)	11786 (-95%)	126350 (-95%)
D38P38SP	75224 (-38%)	164087 (-31%)	0
D38P38SP-INN	77973 (-36%)	167966 (-30%)	0
A14P14SP	8783 (-93%)	19165 (-92%)	0
A14P14SP20	3631 (-97%)	8115 (-97%)	0
A14P28SP	7895 (-93%)	17198 (-93%)	143760 (-95%)
J14P28SP8-INN	6191 (-95%)	13069 (-95%)	143760 (-95%)
D38P76SP	75224 (-38%)	164087 (-31%)	2022018 (-26%)
D38P76SP-INN	77973 (-36%)	167966 (-30%)	2079455 (-23%)
I10P20SP	16415 (-86%)	37566 (-84%)	528152 (-81%)
G4P14SP-INN	23702 (-80%)	52794 (-78%)	617626 (-77%)
G4P14SP	23076 (-81%)	51647 (-78%)	603738 (-78%)
G3P8SP	20579 (-83%)	45591 (-81%)	553507 (-80%)
G1P4SP	19244 (-84%)	43012 (-82%)	473371 (-83%)

Table 5.2: Magnitude of the network after optimality preserving preprocessing (SL→FF→SP). Percentages denote the change relative to the original network. The original network contains 121009 nodes, 239414 cables and 2717018 single point of failures.

Instance	# cables (before)	# cables (after)	Opt (before)	Opt (after)
S1P1SP	15202	1271 (-83%)	88073.08	88930.41 (+1.0%)
T1P1SP	35291	7810 (-78%)	10183.82	10183.82 (+0.0%)
L1P2SP	5191	1240 (-76%)	956665.52	956668.00 (+0.0%)
D1P2SP	3870	684 (-82%)	27142790.82	27142790.82 (+0.0%)
H1P2SP-INN	40317	395 (-99%)	14677.49	15966 (+8.8%)
K1P2SP-INN	39486	395 (-99%)	20629.87	21984 (+6.6%)
F2P4SP	44809	4512 (-90%)	74198.87	74198.87 (+0.0%)
B1P2SP	7123	924 (-87%)	414802954.66	414805914.93 (+0.0%)
C1P2SP	11786	1266 (-89%)	245837076.98	497221875.23 (+102.3%)
D38P38SP	164087	18197 (-88%)	50003418471.13*	60003493745.49 (+20.0%)
D38P38SP-INN	167966	2170 (-99%)	30000828444.99*	30000862993.70 (+0.0%)
A14P14SP	19165	2112 (-89%)	20001203841.18	20001205040.53 (+0.0%)
A14P14SP20	8115	1835 (-77%)	20001292047.84	30000938037.37 (+50.0%)
A14P28SP	17198	2004 (-88%)	22442215988.56*	22444258457.01 (+0.0%)
J14P28SP8-INN	13069	155 (-99%)	61810458097.76*	64214584040.96 (+3.9%)

Table 5.3: Experiments for the preprocessing step based on hierarchy. *Opt (before)* denotes the score of the optimal solution before preprocessing, and *Opt (after)* denotes the score of the optimal solution after full preprocessing. *# cables (before)* denotes the number of cables using optimality preserving (SL→FF→SP) preprocessing. *# cables (after)* denotes the number of cables using full preprocessing (SL→HF→FF→SP). The size of the network before preprocessing is 121009 nodes, 239414 cables and 2717018 single point of failures. Percentages after *# cables (after)* show the decrease after full preprocessing including hierarchy relative to optimality preserving preprocessing. Percentages after *Opt (after)* denote the increase in the optimal solution after full preprocessing relative to optimality preserving preprocessing. Values marked with a * indicate the best known solution, because the optimum before preprocessing is unknown. Instances for which both optima are unknown are not mentioned. For the full magnitude of the networks before and after preprocessing, refer to Table C.2 and C.1.

Instance	Optimality preserving		Full
	Time (-SL, ms)	Time (+SL, ms)	Time (+SL, ms)
S1P1SP	7207	3739	3319
T1P1SP	7924	5852	4574
L1P2SP	32325	8080	6388
D1P2SP	31942	7731	6132
H1P2SP-INN	42146	20492	10085
K1P2SP-INN	40087	19732	9450
F2P4SP	85656	43731	21713
B1P2SP	32340	7215	6218
C1P2SP	38792	13372	8265
D38P38SP	351688	146491	124970
D38P38SP-INN	305786	154047	126037
A14P14SP	138556	63362	49485
A14P14SP20	104065	56331	48744
A14P28SP	2394051	223462	129412
J14P28SP8-INN	X	227735	125456
D38P76SP	X	509360	339889
D38P76SP-INN	331260	560781	345211
I10P20SP	505123	240483	123455
G4P14SP-INN	304931	158212	74987
G4P14SP	300668	159170	78153
G3P8SP	169873	87666	41778
G1P4SP	76550	46512	22363

Table 5.4: Comparison between the time optimality preserving preprocessing takes without preprocessing based on straight line distance (FF→SP, column *Time (-SL, ms)*) and with preprocessing based on straight line distance (SL→FF→SP, column *Time (+SL, ms)*), and the time full preprocessing (SL→HF→FF→SP) takes.

Instance	Opt	Simple heuristic	Time (ms)
S1P1SP	88930.41	88930.41 (100%)	34
T1P1SP	10183.82	10183.82 (100%)	70
L1P2SP	956668.00	6183699.55 (646%)	171
D1P2SP	27142790.82	31873890.54 (117%)	42
H1P2SP-INN	15966	15966 (100%)	14
K1P2SP-INN	21984	21984 (100%)	14
F2P4SP	74198.87	79905.21 (108%)	318
B1P2SP	414805914.93	431446196.69 (104%)	116
C1P2SP	497221875.23	10000100000 (2011%)	36
D38P38SP	60003493745.49	60003493745.49 (100%)	834
D38P38SP-INN	30000862993.70	30000862993.70 (100%)	636
A14P14SP	20001205040.53	20001205040.53 (100%)	145
A14P14SP20	30000938037.37	30001280146.73 (100%)	125
A14P28SP	22444258457.01	22478598888.49 (100%)	373
J14P28SP8-INN	64214584040.96	75606897908.85 (118%)	120
D38P76SP	70890333523.64	71210464042.16 (100%)	2570
D38P76SP-INN	41416301981.57	41766955682.60 (101%)	1873
I10P20SP	7382948.01*	33514585.46 (454%)	144
G4P14SP-INN	416203.60	1894847.97 (455%)	232
G4P14SP	4160358.57*	7897258.69 (190%)	1152
G3P8SP	2469169.05	4807061.34 (195%)	639
G1P4SP	1801837.70	4136269.09 (230%)	305

Table 5.5: Simple heuristic experiments. Values are after complete preprocessing. Percentages show the ratio to the optimal value. Times are averaged over 20 runs. Values marked with a * indicate best known values, because the optimum is unknown.

5.3 Heuristics

5.3.1 Simple heuristic

A comparison between the results of the simple heuristic using the adjusted Dijkstra's algorithm as mentioned in Section 4.4 and the optimal values can be found in Table 5.5. In seven instances, the simple heuristic is enough to find the optimal solution to a problem set. In two cases (C1P2SP and J14P28SP8-INN), the simple heuristic could not solve a problem while the problem is solvable. For all instances which contain dark fiber ring problems (G4P14SP-INN, G4P14SP, G3P8SP and G1P4SP), the results are poor.

Instance	Opt	Complex heuristic	Time (ms)
F2P4SP	74198.87	74198.87 (100%)	20529
D38P38SP	60003493745.49	60003493745.49 (100%)	4928
D38P38SP-INN	30000862993.70	30000862993.70 (100%)	1414
A14P14SP	20001205040.53	20001205040.53 (100%)	1148
A14P14SP20	30000938037.37	30001280146.73 (100%)	853
A14P28SP	22444258457.01	22444258457.01 (100%)	33461
J14P28SP8-INN	64214584040.96	76395324143.48 (119%)	581
D38P76SP	70890333523.64	70890333523.64 (100%)	2330355
D38P76SP-INN	41416301981.57	41416301981.57 (100%)	15053
I10P20SP	7382948.01*	470700237.64 (6367%)	805486
G4P14SP-INN	416203.60	416203.60 (100%)	26766
G4P14SP	4189617.33*	X	X
G3P8SP	2469169.05	2469169.05 (100%)	12884483

Table 5.6: Complex heuristic experiments. Values are after complete preprocessing. Percentages show the ratio to the optimal value. Instances with only one problem have been left out, because using the complex heuristic on these problem sets will be the same as using the ILP solver directly, and is therefore useless. Times are averaged over ten runs. Values marked with a * indicate best known values, because the optimum is unknown.

5.3.2 Complex heuristic

For most instances, the complex heuristic solves the set to optimality. Most of these instances contain problems that are fairly spread out geographically, which means that the capacity changes of one problem do not affect the other problems much. A14P28SP has problems that are all within the same area, but the optimal solution to these problems uses mostly local cables which have a very high capacity.

For problem sets where capacity does play a role, the results of the complex heuristic are much worse. For J14P28SP8-INN, one of the problems becomes infeasible due to the changes in availability caused by previous problems. For I10P20SP, all problems can be solved, but the distance for which they are not geographically separated becomes much larger. The same is true for A14P14SP20.

Some instances only contain single dark fibers. In these cases, the result of the complex heuristic is always equal to the result of the simple heuristic. Therefore, these instances have been left out of the complex heuristic experiments.

5.3.3 Combining heuristics

Since the complex heuristic can take far too much time to complete, the complex heuristic and simple heuristic were combined. If a problem could not be solved to optimality in

Instance	Opt	Combined heuristic	All solved	Time (ms)
F2P4SP	74198.87	74198.87 (100%)	Yes	
D38P38SP	60003493745.49	60003493745.49 (100%)	Yes	
D38P38SP-INN	30000862993.70	30000862993.70 (100%)	Yes	
A14P14SP	20001205040.53	20001205040.53 (100%)	Yes	
A14P14SP20	30000938037.37	30001280146.73 (100%)	Yes	
A14P28SP	22444258457.01	22444258457.01 (100%)	Yes	
J14P28SP8-INN	64214584040.96	76395324143.48 (119%)	Yes	
D38P76SP	70890333523.64	70906973805.41 (100%)	No	154754
D38P76SP-INN	41416301981.57	41416301981.57 (100%)	Yes	
I10P20SP	7382948.01*	474524067.02 (6427%)	No	436944
G4P14SP-INN	416203.60	416203.60 (100%)	Yes	
G4P14SP	4160358.57*	7899761.19 (190%)	No	495129
G3P8SP	2469169.05	4809563.84 (195%)	No	246854

Table 5.7: Combined heuristic experiments. Maximum time for the complex heuristic is set to one minute for every problem. After that, the simple heuristic is used. *All solved* indicates whether all problems could be solved within the time limit for the complex heuristic. Times are averaged over five runs. Values marked with a * indicate best known values, because the optimum is unknown.

one minute, the simple heuristic was used for that problem. The results for this combined heuristic can be found in Table 5.7.

5.4 ILP

The time, the number of edges that needed to be constrained to be integer, and the number of single point of failures that needed to be included to find the optimal solution using the ILP solver were tested. See Table 5.8.

Almost all instances which only contain single dark fiber problems could be solved without constraining any variable to be integer. The only exception to this is A14P14SP20, which is an instance requiring twenty fibers instead of the usual two fibers.

5.4.1 Important single point of failures

The effectiveness of incrementally adding single point of failures as explained in Section 4.6 was tested. The purpose of this is making the calculation of the LP relaxation faster. The downside is that every time after adding a few single point of failures, the ILP solver needs to start over. The ILP solver (Algorithm 5) was initiated with all single point of failures (after full preprocessing) marked as important, with zero single point of failures

Instance	# Int. edges	# Imp. spofs	Time (ms)
S1P1SP	0	0	196
T1P1SP	0	0	1123
L1P2SP	82	241	71416
D1P2SP	23	57	3467
H1P2SP-INN	13	30	2214
K1P2SP-INN	11	22	1651
F2P4SP	27	81	34115
B1P2SP	236	1727	2230586
C1P1SP	165	302	210590
D38P38SP	0	0	10413
D38P38SP-INN	0	0	1597
A14P14SP	0	0	1012
A14P14SP20	5	0	1417
A14P28SP	196	420	479823
J14P28SP8-INN	29	37	5252
D38P76SP	607	2647	44401382
D38P76SP-INN	181	421	348800
G4P14SP-INN	51	66	53579
G3P8SP	598	898	13345233
G1P4SP	442	854	6048068

Table 5.8: ILP solver experiments. Experiments were conducted after full preprocessing. *# Int. edges* is the number of edges that were explicitly constrained to be integer to arrive at the optimal solution, and *# Imp. spofs* is the number of single point of failures that needed to be added to the model to arrive at the optimal solution. Note that edges and single point of failures are directed.

Instance	Time (without, ms)	Time (with, ms)	Time (after CH, ms)
L1P2SP	66749	71416	
D1P2SP	4699	3467	
H1P2SP-INN	3491	2214	
K1P2SP-INN	3429	1651	
F2P4SP	X	34115	10217 (-70%)
B1P2SP	253060	2230586	
C1P1SP	547355	210590	
A14P28SP	X	479823	87350 (-82%)
J14P28SP8-INN	6009	5252	14408 (+174%)
D38P76SP	X	4401382	2412727 (-55%)
D38P76SP-INN	X	348800	36923 (-89%)
I10P20SP	X	X	X
G4P14SP-INN	143498	53579	10047 (-81%)
G4P14SP	X	X	X
G3P8SP	X	13345233	352442 (-97%)
G1P4SP	X	6048068	

Table 5.9: Comparison between the time to reach the optimal solution using the ILP solver, with (*Time (with, ms)*) and without (*Time (without, ms)*) using the important single point of failure heuristic, and after using the complex heuristic (*Time (after CH, ms)*) to determine the initial important single point of failures. Instances with only one problem were not included in the complex heuristic tests, because the complex heuristic only works for instances with two or more problems. Percentages after *Time (after CH, ms)* show the relative change to *Time (with, ms)*.

marked as important, and with only the single point of failures that were used during the complex heuristic marked as important. The results can be found in Table 5.9.

5.5 Local Search

A number of local search experiments were performed to find the ideal parameter settings and to find out how effective this method is. All experiments were performed using simulated annealing with the mutation step using Dijkstra’s algorithm as a shortest path algorithm, a T_0 of 1000000000, and a T_{stop} of 0.001. This results in a fixed number of mutations of 538687 per subproblem for the algorithm to complete when using a cooling schedule of $Q = 10 * \#sp$ and $\alpha = 0.95$, where $\#sp$ is the number of subproblems.

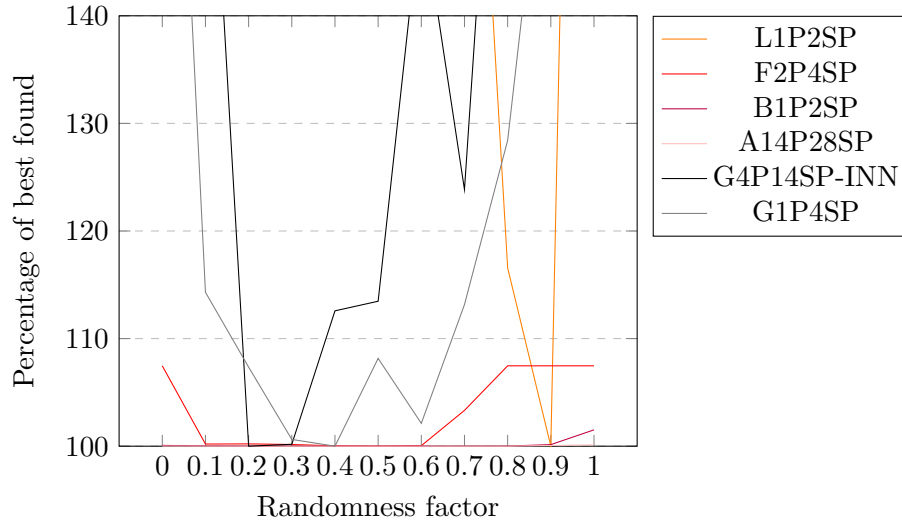


Figure 5.1: Comparison of the randomness factor to the best found score after the simulation annealing algorithm has terminated with a cooling schedule with $\alpha = 0.95$ and $Q = 10 * \#sp$, where $\#sp$ is the number of subproblems. The score is shown in percentages of the best found solution. All experiments have been performed at least 30 times with different random seeds. Details are in Table C.3.

5.5.1 Randomness factor

A randomness factor (see Section 4.9.1) that is too low will result in too many solutions being the same after mutation. A randomness factor that is too high will result in a very low bias towards good solutions, and may therefore need a longer running time to search the neighborhood. To verify this and to find the right balance, several experiments were conducted with different randomness factors. The results can be found in Figure 5.1. More details about every local search experiment that was performed can be found in Table C.3.

The experiments were all conducted using a cooling schedule that was chosen to finish within a minute per subproblem on all the tested instances: $Q = 10 * \#sp$ and $\alpha = 0.95$, where $\#sp$ is the number of subproblems. The simple heuristic was used as the initial solution. C1P2SP was not tested, because the simple heuristic cannot find a feasible solution to it, and finding a feasible solution is NP-complete (see Section 3.1.2). Instances for which the simple heuristic already yielded the optimal solution were not tested either.

The ideal randomness factor differs greatly between solutions, but it is around the area of 0.3 that most instances give decent results. Five out of the six instances that were tested are within 1% of the best found solution at a randomness factor of 0.3. The exception to this is L1P2SP, which gives very poor results on any randomness factor other than 0.9.

Instance	$Q = 10 * \#sp$	$Q = 20 * \#sp$	$Q = 40 * \#sp$
L1P2SP	3932618.75	3726711.91	3830759.17
D1P2SP	27142790.82	27142790.82	27142790.82
F2P4SP	74469.12	74418.73	74386.14
B1P2SP	414880421.92	414807376.22	414805915.23
A14P14SP20	30001280146.73	30001280146.73	30001280146.73
A14P28SP	22447288141.87	22447286702.13	22447282633.79
J14P28SP8-INN	75506508663.76	75506508663.76	75506508663.76
D38P76SP	70903011146.13	70902780373.09	70901514575.18
D38P76SP-INN	41639374523.60	41551910828.57	41555204408.31
I10P20SP	7532797.06	7482340.70	7388896.91
G4P14SP-INN	423478.21	423722.83	423361.56
G4P14SP	4207407.81	4185432.56	4185694.01
G3P8SP	2490502.96	2500119.23	2475050.56
G1P4SP	1832073.08	1811367.25	1823961.92

Table 5.10: Experiments with a higher Q . All experiments were performed using a randomness factor of 0.3 and $\alpha = 0.95$.

5.5.2 Higher Q

For some of the instances, the result of the local search solver was very dependent on the random seed. This indicates that the solver would be able to arrive at much better solutions if it were given more time.

Since the value of $Q = 10 * \#sp$ was chosen to limit the running time of the algorithm to a minute per subproblem for all the instances, this also means that for many instances, the local search solver takes only a fraction of this time to complete. It is possible that these instances would benefit from a longer running time.

Another test was performed where the value of Q , and therefore the expected running time, was doubled to $Q = 20 * \#sp$ and quadrupled to $Q = 40 * \#sp$. The results can be found in Table 5.10.

A higher Q yields better results, but the difference is not very large. In some cases, a higher Q leads to a worse solution, but looking at the confidence intervals in Table C.3, the difference is not significant.

5.6 Combined solver

5.6.1 Comparison between all solvers

A full comparison of the various solvers that have been introduced and the solutions they have yielded can be found in Table 5.11.

The combined solver creates an initial solution with the combined heuristic, switching from the complex to the simple heuristic if a problem could not be solved within one minute. If the complex heuristic finds a solution to all problems within a minute, the important single point of failures (see Section 5.4.1) found by and used by the complex heuristic are used as the initial important single point of failures for the ILP solver. If not all problems could be solved within one minute, the heuristic is used as the initial solution to the simulated annealing solver. If the ILP solver takes longer than two minutes to complete, the best known solution at that point is used as the initial solution to the simulated annealing solver. The simulated annealing solver runs with a cooling schedule of $Q = 10 * \#sp$, $\alpha = 0.95$ and a randomness factor of 0.3.

Instance	Opt	Simple heuristic		Complex heuristic		Combined heuristic	
		Score	Time (ms)	Score	Time (ms)	Score	Time (ms)
S1P1SP	88930.41	88930.41	34	88930.41	(196)	88930.41	(196)
T1P1SP	10183.82	10183.82	70	10183.82	(1123)	10183.82	(1123)
L1P2SP	956668.00	6183699.55	171	956668.00	(71416)	6183699.55	(60000)
D1P2SP	27142790.82	31873890.54	42	27142790.82	(3467)	27142790.82	(3467)
H1P2SP-INN	15966	15966	14	15966	(2214)	15966	(2214)
K1P2SP-INN	21984	21984	14	21984	(1651)	21984	(1651)
F2P4SP	74198.87	79905.21	318	74198.87	20529	74198.87	20529
B1P2SP	414805914.93	431446196.69	116	414805914.93	(2230586)	431446196.69	(60000)
C1P2SP	497221875.23	10000100000	36	497221875.23	(210590)	10000100000	(60000)
D38P38SP	60003493745.49	60003493745.49	834	60003493745.49	4928	60003493745.49	4928
D38P38SP-INN	30000862993.70	30000862993.70	636	30000862993.70	1414	30000862993.70	1414
A14P14SP	20001205040.53	20001205040.53	145	20001205040.53	1148	20001205040.53	1148
A14P14SP20	30000938037.37	30001280146.73	125	30001280146.73	853	30001280146.73	853
A14P28SP	22444258457.01	22478598888.49	373	22444258457.01	33461	22444258457.01	33461
J14P28SP-INN	64214584040.96	75606897908.85	120	76395324143.48	581	76395324143.48	581
D38P76SP	70890333523.64	71210464042.16	2570	70890333523.64	2330355	70906973805.41	154754
D38P76SP-INN	41416301981.57	41766955682.60	1873	41416301981.57	15053	41416301981.57	15053
I10P20SP	7382948.01*	33514585.46	144	470700237.64	805486	474524067.02	436944
G4P14SP-INN	416203.60	1894847.97	232	416203.60	26766	416203.60	26766
G4P14SP	4189617.33*	7897258.69	1152	X	X	7899761.19	495129
G3P8SP	2469169.05	4807061.34	639	2469169.05	12884483	4809563.84	495129
G1P4SP	1801837.70	4136269.09	305	1801837.70	(6048068)	4136269.09	(60000)
Instance	Opt	ILP solver		Simulated annealing		Combined solver	
		Score	Time (ms)	Score	Time (ms)	Score	Time (ms)
S1P1SP	88930.41	88930.41	196	88930.41		88030.41	1477
T1P1SP	10183.82	10183.82	1123	10183.82		10183.82	805
L1P2SP	956668.00	956668.00	71416	3932618.75	63165	956668.00	62813
D1P2SP	27142790.82	27142790.82	3467	27142790.82	8552	27142790.82	2479
H1P2SP-INN	15966	15966	2214	15966		15966	1561
K1P2SP-INN	21984	21984	1651	21984		21984	1234
F2P4SP	74198.87	74198.87	34115	74469.12	317007	74198.87	23351
B1P2SP	414805914.93	414805914.93	2230586	414880421.92	39029	414805997.23	163731
C1P2SP	497221875.23	497221875.23	210590	10000100000		503340110.13	200871
D38P38SP	60003493745.49	60003493745.49	10413	60003493745.49		60003493745.49	15565
D38P38SP-INN	30000862993.70	30000862993.70	1597	30000862993.70		30000862993.70	2706
A14P14SP	20001205040.53	20001205040.53	1012	20001205040.53		20001205040.53	1955
A14P14SP20	30000938037.37	30000938037.37	1417	30001280146.73	100978	30000938037.37	1207
A14P28SP	22444258457.01	22444258457.01	479823	22447288141.87	421693	22444258457.01	118365
J14P28SP-INN	64214584040.96	64214584040.96	5252	75506508663.76	37162	64214584040.96	10856
D38P76SP	70890333523.64	70890333523.64	4401382	70903011146.13	3707408	70899974199.22	3811078
D38P76SP-INN	41416301981.57	41416301981.57	348800	41639374523.60	568927	41416301981.57	53601
I10P20SP	7382948.01*	X	X	7532797.06	2765601	17479264.67	2958891
G4P14SP-INN	416203.60	416203.60	53579	423478.21	81892	416203.60	35899
G4P14SP	4189617.33*	X	X	4207407.81	2045292	4204414.27	2278156
G3P8SP	2469169.05	2469169.05	13345223	2490502.96	840822	2498448.66	1049235
G1P4SP	1801837.70	1801837.70	6048068	1832073.08	477298	1830930.38	679811

Table 5.11: Comparison between all the methods for solving the problem. Times are excluding preprocessing. A randomness factor of 0.3 and a cooling schedule of $\alpha = 0.95$ and $Q = 10 * \#sp$ was used for all simulated annealing and combined solver experiments. Grey values and times between parentheses indicate that the values to these experiments are known to be equal to other experiments, and are therefore included for easy comparison. For example, the solution to the complex heuristic is always equal to that of the ILP solver for instances with only one problem. Times for simulated annealing experiments were not measured if the simple heuristic yielded the optimal result. Optimal values are printed in boldface. Values marked with a * indicate best known values, because the optimum is unknown.

Instance	Found by consultants	Found by combined solver
S1P1SP	88930.41	88930.41 (-0%)
T1P1SP	15177.64	10183.82 (-33%)
L1P2SP	7567644.48	956668.00 (-87%)
F2P4SP	85346101.01	74198.87 (-100%)
K1P2SP-INN	26859	21984 (-18%)

Table 5.12: Comparison between the solutions that were found manually by the consultants of KPN, and the results found by the algorithm.

The complex heuristic is always at least as good as the simple heuristic. There is one exception: I10P20SP. This is interesting because the simple heuristic runs much faster.

While the simulated annealing solver is unable to solve C1P2SP because the simple heuristic cannot provide an initial solution, the combined solver finds the optimal solution. The same is true for one of the problems in J14P28SP-INN.

The ILP solver needs 37 minutes to find the optimal solution to B1P2SP. The simulated annealing and combined solvers find solutions that are optimal or near optimal in only a few minutes.

For the very difficult instances (I10P20SP and G4P14SP), the simulated annealing and combined solvers find answers where the ILP solver does not find any feasible solution. Unfortunately, the solutions for the other difficult problems (G3P8SP and G1P4SP) show that the results from the simulated annealing and combined solvers for these instances are far removed from the optimal solutions.

5.6.2 Comparison to consultants

Some of the instances were derived from old orders, and the solution that was used to ultimately realize the connection was known. The score as defined in Section 3.2.3 was calculated for these solutions so they could be compared with the solutions the algorithm finds. See Table 5.12.

The algorithm found much better solutions than the consultants of KPN found for these instances. For the dual dark fibers, routes with better geographical separation were found in all cases. Even for one of the single dark fibers, the algorithm could find a shorter path than the consultants could. What is also interesting is that for S1P1SP, the consultants found the same route as the algorithm, even though a better route would have been possible if preprocessing based on hierarchy would not have been applied.

It should be noted that in some cases, a solution that is found by one of the algorithms cannot be used for a real connection because of cost considerations or faulty data. This will often result in small bypasses, thus increasing the score. The solutions found by the consultants already include these bypasses, while those found by the algorithms do not.

Chapter 6

Discussion

6.1 Preprocessing

The preprocessing steps that were introduced greatly reduce the size of the network. Even when using only the preprocessing steps that retain optimality, the magnitude of the network is reduced to a fraction of the complete network. This allows the algorithms to be executed much faster and with a much smaller memory footprint. Most instances could not even be solved without preprocessing, or would take much more time.

The benefits that the preprocessing steps bring to the ILP solver and the related heuristics are much greater than they are for the simulated annealing solver and the simple heuristic. One reason for this is that the reduction of variables for the ILP solver is much greater than the reduction of the size of the network as it is being used by the simulated annealing solver, because the ILP solver uses variables for every edge for every subproblem. Another reason is that while the ILP solver needs to consider many variables, the simulated annealing solver is biased towards good solutions, and therefore considers less options. The disadvantage is of course that the simulated annealing solver does not solve the instance to optimality.

For some instances (D38P38SP, D38P38SP-INN, D38P76SP, D38P76SP-INN), the optimality preserving preprocessing steps do not reduce the network much. This is because the problems in these sets are spread out across the entire country, so a very large proportion of the network is relevant to these sets. For these instances, the network that is being used for the simulated annealing solver is still fairly large. However, because the ILP model contains a flow variable for every cable for every subproblem, the number of variables for the ILP solver is still greatly reduced. The relatively large size that remains after the reduction is also not as much of an issue for problems that are spread out geographically, because the results from the heuristic methods for these instances were very good, and in general, the heuristic methods are much faster than the other methods.

6.1.1 Preprocessing step based on hierarchy

When the preprocessing step based on hierarchy is added, the network can be reduced even more, while in most cases, the optimal solution only increases marginally. Especially in the case of problems where all endpoints are main hubs (instances with the *-INN* suffix), the reduction is vast and the algorithms run very fast. In addition, the routes that are found become simpler and more logical, and many large instances that were previously unsolvable can be solved. There is a large reduction because a consequence of the hierarchy preprocessing step is that all cables which are of lower hierarchy than the starting point can never be reached and will always be removed. Note that A14P14SP and A14P28SP (Table 5.3), which consist of connections between one address and one main hub, do not seem to benefit from this. This is probably because the address and main hub are fairly close to each other. Many of the local cables near the main hub cannot be removed because they are still relevant to the address.

The optimal values for instances in which all endpoints are main hubs increase more after preprocessing based on hierarchy than the values for other instances. One reason for this is simply the greater reduction in size of the network. Another reason is that the preprocessing step based on hierarchy removes all cables that are not INN cables. If two local networks are interconnected, it is possible to go from one main hub to another over the local network. If this alternative is shorter or allows for more geographical separation, this will be the optimal solution, and a solution which is not allowed to use this alternative will have a worse score. Luckily, it is highly unlikely that such an alternative would be used in a real situation if there exists a simpler and more logical solution with full geographical separation.

In the case of D38P38SP, one of the problems becomes infeasible due to preprocessing based on hierarchy. That means that there is no solution to this problem that does not ascend and descend in hierarchy multiple times (see Figure 4.13). Fortunately, this does not happen often, because the structure of the network is designed for finding routes with perfect hierarchy ascension and descension. The most likely cause for this to happen is when a cable has reached its capacity. Another good example of this is C1P2SP. The score of this instance after preprocessing is more than twice the score of the instance before preprocessing. Still, the optimal solution before preprocessing is so poorly geographically separated, that it is improbable that this order would be fulfilled at all. Clients requesting geographical separation will probably not accept a solution with too many single point of failures.

There is no lower bound guarantee to how great the reduction is, and there is no upper bound to the increase in optimal solution. In the worst case, all problems in an instance are infeasible when applying preprocessing based on hierarchy. Nevertheless, we have found that in practice, this rarely happens, and the preprocessing steps are only beneficial. Compared to the time the algorithms are allowed to run, and the saving of time that the preprocessing steps bring to these algorithms, full preprocessing is very fast. Even for the

largest instances, preprocessing never took more than a few minutes.

6.1.2 Preprocessing based on straight line distance

Using preprocessing based on straight line distance greatly reduces the time optimality preserving preprocessing takes, because it runs very fast ($O(n)$), and reduces the size of the network greatly, allowing the following preprocessing steps to run much faster. For full preprocessing, the time saving will not be great, because the preprocessing step based on hierarchy also runs in $O(n)$ time, and also reduces the network greatly. This shows again that the order of the preprocessing steps is of great importance; not only to the outcome of the steps, but also to the time the steps take to complete. More experiments would be necessary to determine the optimal order of preprocessing steps, and the effectivity of every individual step.

6.2 Heuristics

As we have seen in Section 3.1.1, solving multiple dark fiber problems at once is NP-hard, even if all problems are single dark fiber orders. Therefore, both heuristic methods solve the problems one by one instead of all at once. The simple heuristic takes this further, and solves every subproblem one by one, resulting in a polynomial time algorithm. The assumption is that solving the (sub)problems one by one instead of at once gives a decent solution as well.

This assumption appears to hold true in cases where capacity is not a big issue. This is the case when, for example, the problems are all in different areas, or when the number of fibers required is small. In instances with many problems within the same area or a large required number of fibers, both heuristic methods perform poorly.

6.2.1 Simple heuristic

Solving multiple dark fiber orders at once is NP-hard, but solving one dark fiber order with two or more geographically separated connections is also NP-hard (see Section 3.1.2). The simple heuristic simplifies this aspect too, by solving each connection one by one, while trying to avoid single point of failures with previously created connections. This guarantees that the simple heuristic runs in polynomial time. The experiments show the simple heuristic to run very fast.

For single dark fiber problems, the simple heuristic is guaranteed to solve the problem to optimality. For multiple single dark fiber problem sets where capacity is not a major issue, the simple heuristic gives excellent results in a very short time, often solving the instance to optimality.

That a score is close to optimality gives an indication of its quality, but it does not necessarily mean that it is a good solution. In some cases, the score of the simple heuristic

is near optimality, but simply because the same number of problems were solved; not solving a problem gives a great penalty. It could still mean that some of the problems that were solved are not well geographically separated. This seems to be the case for A14P14SP20, A14P14SP20, D38P76SP and D38P76SP-INN. Apparently these instances contain problems which can all be solved easily by the simple heuristic, but not always without large single point of failures.

In total, seven out of the 22 instances that were tested could be solved to optimality by the simple heuristic. Five of these are sets which only contain single dark fiber problems. Two of these sets which only contain single dark fiber problems (S1P1SP and T1P1SP) only contain one problem, and therefore only one subproblem, so the simple heuristic is guaranteed to give the optimal solution. If capacity is not an issue, the simple heuristic will also find the optimal solution to single dark fiber problems. This is the case for the other three problem sets. D38P38SP and D38P38SP-INN are spread out over the entire country, so if one route uses a cable, it does not affect the available capacity for another route. A14P14SP does not require any capacity at all, because it was only used for analysis, and not for a real order.

For two instances, the simple heuristic could not solve problems, even though these problems are not infeasible. For C1P2SP, this is most likely because solving the first subproblem optimally changes the capacities in such a way that the second subproblem becomes infeasible. In the case of J14P28SP8-INN, it is most likely because of the high number of fibers in the problem set, as we will see when looking at the complex heuristic experiments.

For the instances which contain dark fiber rings (G4P14SP-INN, G4P14SP, G3P8SP, and G1P4SP), the results from the simple heuristic are not good. This is probably because due to the high number of subproblems in one problem, the number of single point of failure relations is very high. It is harder to find a connection free of single point of failures when there are already four other connections which need to be avoided, than it is to find a connection which needs to avoid single point of failures with only one other connection.

The poor results for I10P20SP can be explained in a similar way. The cables for this instance are very close to one another, which results in a very high number of single point of failures (see Table C.1). This makes it more difficult to find solutions without single point of failures. If one connection is already solved, it may become impossible to find another connection which does not form a single point of failure with the existing connection. Another factor is that because all problems are in the same area, capacities start playing a role. The capacities of the cables which do not form single point of failures with the existing connection may have been reached due to other problems in the instance.

6.2.2 Complex heuristic

The complex heuristic gives much better results. In nine out of the twelve instances with two or more problems that were tested, the complex heuristic found the optimal solution.

The main disadvantage is that it only simplifies the NP-hardness of solving multiple dark fibers at once. It still solves the geographical separation constraint as an ILP problem. Therefore, the execution times of the complex heuristic are poor compared to the simple heuristic. Compared to the ILP solver however, the complex heuristic finds excellent solutions for many instances in a much shorter time.

Unfortunately, the results for instances in which capacity is very important, are poor. This is most visible in I10P20SP, which is an instance with ten dual dark fiber orders in one dense area. The result found by the complex heuristic contains very poor geographical separation compared to the best known solution. What is surprising is that the result is even worse than the result from the simple heuristic. This is most likely a coincidence. Solving one problem to optimality forced another problem to have worse geographical separation. Since the problems in I10P20SP are all within the same area, capacities play a larger role, while the complex heuristic works excellently for instances in which capacity does not play a large role. This also shows that solving problems simultaneously instead of one by one can give great improvements, if the problems are within the same area.

Another example is J14P28SP8-INN. This is an instance with fourteen dual dark fiber orders of eight fibers instead of the usual two, making capacity more relevant. In this case, the complex heuristic failed to solve a problem that is not impossible to solve.

A way to improve the quality of the solutions that the complex heuristic finds could be to look at different orders of solving the problems. Since the solution of one problem sometimes prevents a good solution for another problem, a change in order of solving could sometimes give good improvements. The difficulty would be to predict which order will give the best results.

One way to possibly speed up the complex heuristic would be to use the simulated annealing solver instead of the ILP solver to solve the individual problems. This would probably make the complex heuristic much faster, while still giving good solutions. It will also be faster than running the simulated annealing solver on the entire instance at once, because the network for individual problems can be much smaller than the entire network, especially when the instance has orders spread out over the country (see Table C.1). The disadvantage is that a complex heuristic using simulated annealing will not provide important single point of failures, and therefore cannot be used to reduce the time the ILP solver takes.

Instances with only single dark fiber problems are guaranteed to give equal results for simple and complex heuristics. If we compare the execution times from Table 5.5 with those from Table 5.6, it becomes clear that the simple heuristic runs much faster.

6.2.3 Combined heuristic

Out of the four instances that could not be solved within the time limit of one minute per subproblem by the complex heuristic, the combined heuristic could find a solution to the largest instance (D38P76SP) that is close to the optimum within a few minutes. The

complex heuristic would take several hours to find a good solution to this instance.

For two of the instances (G4P14SP and G3P8SP), all problems (except one single dark fiber) failed to be solved within the time limit of the combined heuristic. Therefore, these instances have a score that is equal to that of the simple heuristic.

What is most interesting is that for I10P20SP, the result from the combined heuristic is worse than both that of the simple heuristic and that of the complex heuristic. This is again explainable by the large role that capacities play in this instance. This can coincidentally make solutions to problems turn out worse.

6.3 ILP

For most instances, the ILP solver solves the problem to optimality in a reasonable time. Out of the 22 instances that were tested, twelve could be solved within a minute. Four problems (B1P2SP, D38P76SP, G3P8SP, G1P4SP) took an exceptionally long time that is not acceptable given their size. Looking at the number of important single point of failures that needed to be added to the model before the optimal solution could be reached, it becomes clear that instances with more relevant single point of failures take much longer to solve.

For most of the instances which only contain single dark fiber problems, it was not necessary to constrain any variable to be integer. Therefore, these instances can be solved in polynomial time by simply solving the LP relaxation. Even though these instances had multiple optimal solutions, there was always at least one of them in which all variables had integer values.

The only exception to this is A14P14SP20. In this instance, capacities play a large role because of the high number of fibers in this instance. However, the instance was still relatively easy to solve with only five integer variables. In addition, a dark fiber order requiring twenty fibers is very rare.

6.3.1 Important single point of failures

Incrementally adding single point of failures as they are violated can significantly decrease the execution time of the ILP solver. Even when starting with zero single point of failures, six out of the eight instances that were unsolvable when simply adding all single point of failures to the model at once, become solvable. For most other instances, the execution time of the ILP solver reduces. In the case of C1P1SP, the reduction is more than half.

When adding the important single point of failures that were found during the calculation of the complex heuristic as the initial important single point of failures, the decrease in execution time of the ILP solver becomes very large. In one case (G3P8SP), the reduction is as high as 97%, while all the instances are still solved to optimality.

The only exception to this is J14P28SP8-INN. This means that the important single point of failures that were found during the complex heuristic and the important single

point of failures to the complete instance have little overlap. This could be due to the fact that one problem could not be solved by the complex heuristic. If most important single point of failures occur in this problem, the complex heuristic will not have provided them.

A disadvantage is that to find these important single point of failures, the complex heuristic has to be run, which can take a long time. In addition, the intermediate results from the ILP solver are generally poor. This is because during intermediate results, not all important single point of failures may have been added. These single point of failures will then not be considered. Another disadvantage is that the complex heuristic only works for instances with more than one problem. B1P2SP for example, where the execution time without using important single point of failures is much shorter than the execution time with important single point of failures, it would be interesting to see if providing a good initial set of important single point of failures would solve the problem faster. Unfortunately, in this case, the complex heuristic will not be able to provide them.

It would be interesting to try to find a faster heuristic to find important single point of failures, and one which would also work on instances with a single problem. An example might be to generate several solutions using a fast heuristic method and see which single point of failure constraints are violated in these solutions. If a fast heuristic to find important single point of failures would exist, the total time to find optimal solutions could be decreased drastically.

Another idea may be to give a higher priority to single point of failures with a greater length. Since these single point of failures have a larger influence on the score of the solution, the intermediate results will be better.

6.4 Local search

The simulated annealing methods provide good solutions. For some of the instances that are very difficult to solve using the ILP solver, the simulated annealing solver comes with very good solutions, almost all within the execution time goal of approximately one minute per subproblem. The simulated annealing solver also finds much better intermediate results. Because the mutation step uses a higher weight for edges which are part of single point of failures, even when using a high randomness factor, the search is quickly guided to a solution with a low number of single point of failures and therefore a good score.

The mutation step, based on the destroy and repair steps from large neighborhood local search, allows every possible path to be found, while still being able to bias the search towards good solutions. Therefore, the mutation step that was introduced could also prove to be useful for solving other NP-hard path finding problems, such as the longest path problem. Unfortunately this mutation step is relatively slow. It is difficult to find faster mutation steps. Using the A* heuristic [15] will not work well because the straight line distance does not relate to the penalty for a single point of failure heuristic. Using other methods like bidirectional Dijkstra's [16] algorithm could work, but the effect will most

likely not be as good as on normal pathfinding problems, because of the randomness we introduce.

The simulated annealing method described in this thesis uses the same complete network (after preprocessing) for every subproblem. The ILP solver uses different networks for every subproblem, generating only the variables for relevant cables to a certain subproblem. This idea could be used for the simulated annealing solver as well. If the mutation step would work with multiple smaller networks, it would take more memory, but it could also speed up the mutation step. It can also reduce the number of infeasible solutions, though there are already very few.

That the optimal solution is not guaranteed to be found is not a major problem, because as we have seen when discussing the heuristic methods, the capacity of the network can influence the optimal solution greatly. Therefore, what is an optimal solution today might not be an optimal solution when the capacities in the network have changed. Also, after preprocessing based on hierarchy, the optimal solutions change. The optimal solution after preprocessing based on hierarchy is not the real optimal solution. What could be an issue however, is that when using only the simulated annealing solver to find a solution, there is no way to know how far the solution is removed from the optimum. This could result in orders being rejected even though it might be possible to solve them. An example is L1P2SP, where the simulated annealing solver has great difficulty coming close to the optimal solution, while the ILP solver solves it to optimality in a little over seventy seconds.

The number of mutation steps that result in infeasible solutions is very small. However, even though there is a fair amount of randomness, there is still a significant number of solutions that remain equal after mutation. Methods to resolve this could have a great impact on the results on the simulated annealing algorithm.

6.4.1 Randomness factor

Using fully random routes does not give good results, because there are too many neighbors to consider within the short timespan the simulated annealing algorithm is allowed to run in. However, fully biasing the solution towards good solutions does not work well either. This is because simulated annealing relies on occasionally accepting worse solutions, to escape from local optima.

One example of when this becomes problematic is when, in one dual dark fiber, solving the first subproblem to optimality makes the second subproblem infeasible. The first subproblem would have to be changed to a worse solution to be able to solve the second problem, and arrive at the global optimum. Another disadvantage of biasing towards good solutions is that many mutations will result in a path that has already been found before.

The randomness factor describes the balance between bias towards good solutions and randomness, where a factor of 0 means a fully biased mutation, and a factor of 1 means a fully random mutation. The best randomness factor seems to be around 0.3, although the optimal factor can differ greatly between instances. For L1P2SP for example, only

a randomness factor of 0.9 gave good results. Unfortunately we do not know in advance which randomness factor is going to be the best.

One solution to this could be to use an adaptive randomness factor. Since the number of mutation steps that result in equal solutions depends heavily on the randomness factor, it could be used as an indicator to whether the randomness factor is too high or too low. We could start with a low randomness factor, and if there are too many equal solutions, the randomness factor could be increased for future mutations.

It should be noted that mutations steps with a higher randomness factor were generally slower. This can be explained by the fact that mutation steps with lower randomness factors generally find shorter paths. In the network of KPN, shorter paths generally also consist of a lower number of cables. Due to the changed lengths as used in our mutation step, the shortest path, i.e. the path with the lowest adjusted weight, is more often a path consisting of a higher number of cables. Therefore, Dijkstra's algorithm needs more steps to complete. Due to this, a solver that uses a lower randomness factor would be able to run for a longer time without violating the time limit, and therefore give better solutions.

6.4.2 Initial solution

Simulated annealing starts from an initial feasible solution. Finding an initial solution where all problems are solved to an instance with more than two dark fibers or one or more dual dark fibers is NP-hard. For this thesis, the simple heuristic was used as a starting solution. However, the simple heuristic does not guarantee that all problems are solved. For two instances (C1P2SP and J14P28SP-INN) this was the case. Therefore, the simulated annealing solver could not solve the problems that remained unsolved by the simple heuristic and the solutions were far removed from optimality. For the combined solver, this is not an issue anymore for these instances, because the ILP solver reaches a feasible solution before the time limit.

For these instances to be solved using a pure simulated annealing strategy, the mutation step and score functions would have to be adjusted in a way that would temporarily remove the solution to the subproblem that conflicts with the subproblem that cannot be solved. One could think of changing the mutation step in a way that there is a probability that all edges belonging to a certain subproblem are removed from the solution, and there is a probability that such a subproblem would be solved again.

6.4.3 Higher Q

Using a higher Q generally increases the quality of the solution that is found. Since KPN uses basic consumer hardware to run the algorithms described in this paper, a relatively low Q had to be chosen. When better hardware becomes available, the value of Q could easily be increased while still keeping the execution time under one minute per subproblem.

When using a higher Q , the algorithm uses the same value for T for a larger number of

mutations. Since the mutation step includes randomness, this means that a larger part of the neighborhood is allowed to be explored using a certain value for T . The necessity to bias the mutations towards good solutions decreases then. Therefore, it could be beneficial to use a higher randomness factor. Currently, the number of mutations that result in equal solutions is quite large. If a higher randomness factor would be used, it would also solve this problem. On the other hand, using a higher randomness factor could slow down the mutation step.

Using a higher Q also makes the results of the simulated annealing algorithm more reliable. With the experiments with a lower Q , very good solutions for L1P2SP were found, but not often enough to conclude that the algorithm works well for that instance. The outcome was very dependent on the random seed. This is also visible in the confidence intervals (see Table C.3). When using a higher Q however, the confidence intervals become much smaller, and the solutions become much better.

Q was chosen to be a constant times the number of subproblems mainly because of the goal we have for the execution time. When the goal is to find the best solution, regardless of strict time constraints, there may be better options. A solution with a larger neighborhood should explore more neighbors before converging. The size of the neighborhood is not necessarily proportional to the number of subproblems. One could think of taking into account the degree of the relevant nodes or the size of the network after preprocessing. These will influence the number of nodes.

One idea that was tested was to let Q depend on the number of edges to approximate the size of the neighborhood. A route is destroyed by randomly choosing two edges from it. Therefore, the number of possible destructions on a route r is the number of pairs of edges in a route, including all edges with themselves: $|Route_r| * (|Route_r| - 1) / 2 + |Route_r|$. There are $|R|$ routes, so the number of possible destructions is $\sum_{r \in R} (|Route_r| * (|Route_r| - 1) / 2 + |Route_r|)$. The Q parameter of the simulated annealing algorithm was defined to be this number times a constant. While the results were good, it turned out that solving some of the larger instances would take many hours, which was not acceptable in practice.

For some problems, the mutation step takes much longer than for other problems. Since the mutation step is just the simple heuristic with added randomness, Table 5.5 gives a good indication to the time the mutation step will take for each instance. If the mutation step is fast, the simulated annealing algorithm will be able to terminate long before the time goal, because the algorithm makes 538687 mutations per subproblem before terminating, regardless of the instance. Therefore, if the mutation step is fast, Q could be increased while still keeping execution time low. If there were a good way to approximate the time that a mutation step will take, this could make a good improvement to the algorithm.

A high Q significantly improves the results. However, a low Q of 10 times the number of subproblems already generates very good solutions for most instances. For a higher Q to really have a significant impact, the increase in Q has to be fairly high. This increases the time it takes for the algorithm to complete as well. Instead, it could be more useful to keep Q low and try several different randomness factors instead, as this has a higher

impact on the solution.

6.5 Combined solver

We have seen that providing good initial single point of failures can greatly improve the execution time of the ILP solver. Providing a good initial solution can increase the quality of the results of the simulated annealing algorithm too. The combined solver tries to combine all the solvers that were introduced in this thesis to get the best possible solution within a given time.

The combined solver finds very good solutions, almost always reaching the optimum. It took a long time to solve C1P2SP with the ILP solver, and it was impossible to solve this instance using the simple heuristic, but the combined solver finds a solution that is near the optimum in less than three minutes. D38P76SP-INN, A14P28SP, F2P4SP and L1P2SP are solved to optimality, and guaranteed to be optimal, much sooner than the ILP solver due to the complex heuristic providing important single point of failures.

For the five instances that could be compared to the manual results, the combined solver never found a worse solution. The solutions found by the combined solver were almost always much better than the ones found manually. Even if you take into account the fact that not all automatically generated solutions can be used to complete real orders, the improvement is so great that it is undoubtedly helpful.

A reason why the algorithm might perform so much better than the consultants, is because the consultants usually strictly adhere to the hierarchies and structures that are put on the network. An example is using the ring structures in the network to avoid single point of failures. A ring is supposed to be free of single point of failures. This works well for one ring, but when a solution contains multiple rings, the rings could form single point of failures with each other. Then finding a solution without single point of failures quickly becomes very complicated. Even one ring can cause problems, since the rings are often not fully geographically separated in practice.

The algorithms introduced in this thesis do not consider such structures. They simply try to find the paths with maximum geographical separation. This results in much better solutions, and these solutions are valid. For the dark fiber product, adhering to the structures in the network is not necessary.

That consultants often adhere to network structures also becomes clear if we look at S1P1SP. For this simple single dark fiber order, the consultants found the exact same solution as the combined solver. This is interesting, because as we can see in Table 5.3, this is not the best possible solution. Without preprocessing, the algorithms would have found better solutions. Therefore, in this case, it is due to the preprocessing step based on hierarchy that the more logical solution was found.

Even though preprocessing based on hierarchy in some cases makes it impossible to find the real optimal solution, the solutions that were found were never worse than the results

that were found manually for the tested instances.

6.6 Reflection

6.6.1 Perspective on the interpretation of the experiments

Some of the experiments took many hours to complete. The experiments in which the outcomes are not deterministic (i.e. the simulated annealing experiments) were all conducted multiple times to ensure reliable outcomes, but some of the ILP solver experiments that took several hours were conducted less often. Therefore, some of the execution times that are mentioned may deviate somewhat from the average running time if it were conducted more often.

Even though the solutions that are found by the algorithms in this thesis are valid given the data, it may not always be possible to turn these solutions into real connections in practice. The data that was used for the network was not always reliable (see Appendix A), so it may turn out that a valid solution found by the algorithm is not possible in real life. Also, the algorithm for creating virtual cables introduced in this thesis assumes that it is possible to dig everywhere. In reality, there may be a river or a railroad blocking these places. This could very well influence the decision about which SAN cable to connect to, and therefore even which local network to connect to. Even though most solutions will not have to be changed very much to solve these issues, it should be taken into account when comparing the algorithm to the results that were manually found.

6.6.2 Network single point of failures

As discussed in Section 2.2, cables belonging to the same ring should ideally never form a single point of failure. In reality however, they often do. Many cables are left over from older network structures or acquired from other companies, so these were not originally designed to be free from single point of failures. The algorithm for single point of failure analysis in Section 4.1 of this thesis can be very helpful in identifying where these problems occur, and which of them should be prioritized. Then the definitions of the rings could be adjusted, or if necessary, cables could be relocated to make the rings fully separated.

If this would happen, it would become much easier to find geographically separated paths. First of all because the number of single point of failures in the network would be reduced, but also because simply following the ring structure would create paths that are free from any single point of failures. Of course, in some rare cases, this might not be possible due to overlapping rings, but it will at least make the problem much easier to solve.

6.6.3 Score definition

The definition of the score that was used is based on the goal of maximizing geographical separation. In reality however, not all single point of failures are equally bad. Also, geographical separation and (digging) length are not the only concerns when finding solutions.

A single point of failure occurring inside a building on property owned by KPN is not as bad as a single point of failure occurring on a public street, because the probability that these connections will fail is smaller. Similarly, there are areas in the country where construction work is more likely to happen than in other areas. This should be simple to include in the algorithm. Once there is data about where construction work is most likely to happen, i.e. where the urban areas are, the lengths of the single point of failures in these areas could be increased in the model.

What was also not taken into account during this research is that clients will not always accept every solution. The assumption that every client would rather want a solution that is very poorly geographically separated than no solution at all is almost certainly wrong. In many cases, a client will not accept this. Therefore, if we calculate an instance of multiple problems, we could prefer a solution in which most problems are unsolved, but the problems that are solved are fully geographically separated over a solution in which most problems are solved, but contain single point of failures. Unfortunately, this is very difficult to take into account. Simply changing $C_{unsolved}$ is enough to do this, but every client has different requirements, so finding the right $C_{unsolved}$ is impossible without knowing whether a client will accept or reject a solution. And knowing if a client will accept or reject a solution is impossible because clients will always request a solution with as much geographical separation as possible.

The reason we allow preprocessing based on hierarchy to make the solutions that are found worse, is because these solutions are often more logical. Therefore, in practice, they are not much worse than the optimal solutions. If these solutions are not much worse, it would be better if more logical solutions also had better scores. This would close the gap between the solutions found by after optimality preserving preprocessing and those after full preprocessing even further.

Turning a solution into a real connection can cost a large amount of money. Welding points have to be made, the virtual cables have to be dug, and connections have to be changed. But there is also a profit from the client who pays for the connection. What is the best solution for the client is not always the most profitable solution for KPN. The definition of the score and the weights of the cables could be changed to be based on the real costs and profits of connections. In that way, the objective of the problem could be changed to find the solution that would yield the highest profit instead of finding the solution with best geographical separation.

6.6.4 Other methods

The method described in this thesis to model the geographical separation constraint is based on graph pathfinding and requires heavy preprocessing as described in Section 4.1 and Appendix A. Another way to look at it is to use geometric algorithms, and try to find paths which avoid single point of failures directly from the drawings. The geometric algorithm to find single point of failures as described in Section 4.1 can be helpful in creating such an algorithm.

Dark fiber rings are now solved in the same way as the other dark fiber products. The advantage is that the same algorithm can be used. The disadvantage is that the order in which the locations of the ring are connected needs to be decided in advance. It would be interesting to look at options for automatizing the process of finding this order as well. This would add another layer of complexity to the algorithm, since there is an exponential number of options.

Chapter 7

Conclusion

7.1 Overview

Manually finding paths for dark fiber connections through a large network is a time-consuming process that yields results that are far from optimal. In this thesis, the options for automating this process have been examined.

Unfortunately, both the problem of finding multiple dark fiber connections in a network with limited capacity, and the problem of finding geographically separated dark fiber problems are NP-complete.

To try to solve the problem, the network can be represented as a graph. The number of addresses in the country is very large. In addition, there can be many cables near the address to connect the address to. Therefore, representing all addresses as nodes in the graph would make the graph far too large to efficiently use for computation. A better approach is to only add nodes for addresses temporarily in order to solve a specific problem.

To model geographical separation in a graph, geographical information of the cable has to be analyzed and turned into constraints on the graph. The algorithm described in Section 4.1 analyzes vector drawings of cables and outputs pairs of cables that are not geographically separated (*single point of failures*), along with a length for which they are not geographically separated.

The problem can be formulated as an integer linear program (ILP) based on multi-commodity flow problems. To model geographical separation, the single point of failure pairs can be used in an ILP model by introducing a variable and constraint for single point of failure violations, and assigning a penalty to that variable. This ensures a soft constraint, while still being able to fully correctly model the single point of failure constraint.

Solving problems using this method requires a lot of memory and takes far too much time, because of the large size of the network and the large number of single point of failure pairs.

To reduce the number of single point of failures in the model, the single point of failure

constraints can be incrementally added (marking these as *important*) until there are no violated constraints left which are not marked as important. This is guaranteed to still give the optimal solution.

Preprocessing can be used to reduce the size of the network, based on information about the problems. Several such preprocessing steps have been introduced. Using preprocessing based on straight line distance, the time and memory preprocessing takes can be greatly reduced.

Using only preprocessing steps that do not change the optimal solutions and the important single point of failures method, the size of the network can be greatly reduced and several of the instances that were tested could be solved to optimality. For a few other instances, it was possible to find feasible solutions, but not the optimal solutions. However, most instances could still not be solved.

Since the network contains hierarchies, this information can be exploited to introduce a preprocessing step based on hierarchy. This step is based on the idea that good solutions will usually not ascend, then descend and then ascend in hierarchy again. This preprocessing step is not guaranteed to preserve optimality, but we have found that in many cases it does preserve optimality, or increase the objective value only marginally. The preprocessing step based on hierarchy accomplishes very large reductions in the size of the network, especially when the endpoints of the problem are at a high hierarchy.

Using these steps, the ILP solver is able to find excellent solutions to almost all tested instances, even for some of the larger instances that only occasionally need to be calculated. However, not all instances could be solved, and some instances took several hours to solve, which is not acceptable.

To be able to solve very hard instances in a reasonable time, a simulated annealing algorithm can be used. A good mutation step to use for simulated annealing is destroying part of a connection and then reconstructing it. To choose which part of a connection is used, an adjusted random distribution can be used to make sure every edge has an equal chance of being removed.

Because the connection can often be reconstructed in many ways, the neighborhood can become very large. It would take a far too long time to discover the entire neighborhood. A solution to this is to bias the search towards good solutions by reconstructing the connection using a shortest path algorithm on a graph with adjusted weights to avoid single point of failures.

Fully biasing the search towards good solutions does not work well, because the number of mutations that result in equal solutions will be very large. Fully randomly searching the neighborhood does not work well either, because the neighborhood is so large. A *randomness factor* can be used to adjust the balance between bias and randomness.

What is a good randomness factor is very dependent on the problem. For most problems, a randomness factor of 0.3 seems to be a decent choice. If a fixed randomness factor is not required, or if time is not important, the randomness factor could be adjusted based on the number of mutations that result in equal results, or the computation can be performed

multiple times with different randomness factors.

The simulated annealing algorithm is able to find very good solutions to most instances in a reasonable time. When the cooling schedule depends on the number of subproblems, and a cooling schedule of Q equals 10 times the number of subproblems and an α value of 0.95 is chosen, the schedule will finish within a minute per subproblem for almost every instance.

Since finding a feasible solution to only one dual dark problem is already NP-hard, it is impossible to create an initial solution in which dual dark fiber problems are solved in polynomial time, unless $P = NP$. The combined solver found a feasible solution for every tested instance, but there is no guarantee that this will be the case for every possible instance.

Based on the way consultants currently manually solve problems, a simple heuristic method was introduced. The simple heuristic calculates paths avoiding single point of failures with paths that were calculated earlier. The simple heuristic runs in polynomial time and is guaranteed to solve instances with one single dark fiber problem to optimality. It is also guaranteed to solve sets of single dark fibers requiring zero fibers (used for analyzing purposes) and sets of single dark fibers with fully disjoint networks after preprocessing. It serves as a good initial solution to simulated annealing algorithms. In other cases, the simple heuristic often performs poorly, and there is no guarantee that the simple heuristic can find feasible solutions to dual dark fiber problems or instances with two or more dark fiber problems.

Using the methods to solve the problem as an ILP problem, a complex heuristic was introduced. This complex heuristic involves solving all problems one by one to optimality, updating the capacities after every solved problem. This heuristic works very well on problems in which capacities do not play a major role, for example when the problems are spread out geographically in such a way that they do not interfere.

Since the complex heuristic can take a very long time, a combined heuristic was introduced. The combined heuristic tries to solve each problem in an instance to optimality using the complex heuristic. If it fails to solve a problem within 60 seconds, the simple heuristic is used for that problem instead.

An advantage of the complex heuristic is that it also uses important single point of failures. The single point of failures that were marked as important during the calculation of the complex heuristic can be marked as important when using the ILP solver as well. This results in a very large speed up of the ILP solver. If there were a faster method to find which single point of failures are most likely to be important, this could drastically decrease the time to solve instances to optimality.

Since the complex heuristic can provide a very good estimation of the number of which single point of failures are important to the ILP solver, a combined solver was introduced. The combined solver uses the combined heuristic to determine if an instance is easy or difficult by seeing if the time limit for the combined heuristic was ever violated. If it was, the instance is considered difficult, and solved with simulated annealing using the result

from the combined heuristic as the initial solution. If it was not, the important single point of failures that were found by the combined heuristic are used as the initial important single point of failures for the ILP solver. If the ILP solver still fails to solve the instance within two minutes, the simulated annealing solver is used instead using the best known solution at that moment as the initial solution.

7.2 Best methods to solve problems

For single dark fibers, a simple shortest path algorithm, such as used for the simple heuristic, will often perform very well after full preprocessing. When there is only one problem, a shortest path algorithm will solve the problem to optimality in a very short time. When there are multiple problems spread out over a large geographical area, shortest path algorithms are also very likely to find optimal solutions. When many single dark fibers are in the same area, capacities become an issue. In these cases shortest path algorithms perform poorly.

In such instances where it is required to solve multiple single dark fiber problems in the same area, and the capacities of the cables in that area are limited, an ILP solver will often still solve such a problem to optimality. In many such cases, the LP relaxation already has an integer optimal solution. In case the ILP solver does not solve the problem within a reasonable time, local search methods can be used.

One dual dark fiber problem can best be solved by using an ILP solver. This will often give the optimal solution within a reasonable time. In some rare cases, the ILP solver will take a longer time. For these cases, local search can be used. When an instance contains multiple dual dark fiber problems spread out over a large area, the best is to use an ILP solver with a good heuristic to find which single point of failures are important. The complex heuristic can be used for this purpose. If the dual dark fibers are all within the same area, the problem becomes very difficult, and it is most likely best to use local search methods.

Network dark fiber problems with two connections, like dual dark fiber problems, both contain a quadratic number of single point of failure constraints. Dark fiber rings, however, contain more connections and can contain many more single point of failure constraints. Therefore, these are very difficult to solve. An ILP solver can solve some instances to optimality, but is likely to take many hours. It is better to use local search methods for these types of problems. An exception is when all nodes of the ring are of a high hierarchy. In such cases, the preprocessing step based on hierarchy will reduce the network to such an extent that the instance can be solved to optimality using an ILP solver.

Currently, the best way to be guaranteed to solve any instance to optimality is to use optimality preserving preprocessing, then using the complex heuristic to find the important single point of failures, and using these as input to the ILP solver. Unfortunately, this will be slow and may fail for large or difficult instances due to the large memory requirement.

7.3 Comparison to the current situation

Finding shorter paths for dark fibers by automating the process saves capacity in the network, and therefore leaves more capacity for other services with a higher margin of profit than the dark fiber product.

Since the simple heuristic is based on the way consultants currently manually find routes, this indicates that this method is far from optimal. There is much to be gained in both single point of failure reduction and capacity saving by looking at both routes at the same time, instead of first finding a shortest path, and then finding a second path avoiding single point of failures.

The optimal solution for a set of dark fiber orders is sometimes much better than the solution found by the complex heuristic, especially with a large number of orders, and orders within the same geographical area. This means that looking at multiple orders at once instead of finding solutions for orders one by one can in some cases improve solutions drastically. It is far too difficult for humans to consider this when finding routes manually, but using the algorithms described in this thesis, it is possible to take this into account and find much better solutions.

The algorithms that were introduced in this thesis are able to find excellent solutions to the problem of finding multiple dark fiber connections with geographical separation. The algorithms even work for large instances that only occasionally need to be calculated. The structures in the network make it possible for humans to quickly find connections in the network, but the quality of these connections is often not as good as is possible. Automating the process of finding connections can give great improvements to the efficiency in the use of the network, and can generate routes with better geographical separation.

Appendix A

Data interpretation

During the research, only limited data about the network was available. The logical data and the geographical data of the inter node cables had to come from separate sources which were difficult to combine. In addition, there was no logical data about the local cables available. The only data about local cables that was available, were digital vector graphic drawings that were redrawn by hand from drawings on paper. Therefore, all information about local cables was geographical. There was no data on where cables started or which places in the network it visited. The data itself also contained various inconsistencies, for example in names of cables or nodes, or in units of lengths. This appendix discusses how the data was converted into a logical, consistent graph which could be used to efficiently apply the algorithms to. The solutions generated by the algorithms discussed in this thesis rarely turned out to be impossible in practice, so the assumptions made to generate the graph must not be far from reality.

A.1 Inter node network

A.1.1 Endpoints

For the inter node cables, there was logical data available in the form of a database containing all the sections of cables, including their properties. A cable section is a part of a cable. A cable consists of several cable sections, i.e. a cable can be split in two at a certain point. A cable runs between two main hubs, but a cable section could run between two welding points, or a welding point and a main hub.

Unfortunately, even though the cables were reported per section, the properties attached to these sections were the properties of the entire cable, not just the section. There was no data available on welding points. For example, if a cable used to go from point A to point B, but is then split in the middle to also go to point C, there would be two entries: A-B, and A-C, both with the same cable name and cable number, but with different lengths

and different capacities. The sum of these two capacities would then result in the original cable capacity.

To get as close to reality as possible with the available data, the assumption was made that the cable name and both endpoints constitute one unique cable. Since the capacities for cables that were split were also split in the data, this will not result in capacities that are too high.

A.1.2 Geographical information

Geographical information was available from a different source than the logical data. Unlike the logical information, the geographical information was reported per cable instead of per cable section. The geographical information was matched with the logical information on cable number. This means that, considering the example mentioned earlier, both A-B and A-C will have the same geographical information, i.e. they will both follow the exact same path. This is not correct, but since both entries have the same cable name and number, and since there is only one drawing for this cable name and number, it was the best option with the available data.

For some cables, there was no geographical data available at all. For these cables, it was assumed that it forms a single point of failure for every other cable that runs between the same endpoints. The length of the single point of failure is the minimum of the lengths of the two cables.

A.1.3 Lengths

The lengths of the cables as reported were not always correct. For example, many of the cables had a reported length which was shorter than the straight line distance between its endpoints. This is of course impossible. Many of these cables do not belong to the main infrastructure, and were rarely used, but an algorithm which looks for short routes will prefer these cables, and therefore prefer routes which are impossible. Since these cables were rarely used, the choice was made to leave all these cables out of consideration. This makes routes between some locations impossible in the algorithm, even though they might be possible in reality.

A.1.4 Hierarchy information

In some rare cases, INN cables were incorrectly marked as PAN cables. Due to this, the preprocessing step based on hierarchy that was introduced in this paper in some cases removed INN cables when they should not be removed, because they were marked as PAN cables. Since INN cables are often the only way to navigate between main hubs, and in some areas there are not many alternative INN cables nearby, this can result in infeasible problems. Unfortunately there is no simple solution to this, because given the digital data

that was available for this research, there is no other reliable way to distinguish between INN cables and PAN cables.

A.2 Local network

For the local network (PAN/SAN), the only information that was available about the cables were vector graphic drawings and the corresponding cable names. These drawings were first drawn by hand on paper, and then manually redrawn from paper into vector graphics (a drawing of one cable consists of a set of connected line segments). This makes the data less reliable, and the lack of logical information (length, endpoints, capacity, . . .) makes it difficult to efficiently run graph algorithms on.

Another issue was that the drawings are per cable, not per section. So for example, if a PAN cable was cut to connect a SAN cable to it, there was no information about this on the drawing of the PAN cable. Similarly, there was no information on the drawing of the SAN cable about which PAN cable it was connected to. There was also no information about which cables are SAN cables and which cables are PAN cables.

Most cables follow the paths of streets. When the paper drawings were redrawn to vector graphics, the streets these cables run alongside were used to determine the location of the cable. Therefore, if a cable is drawn alongside a street on the vector graphics, we can safely assume that this data is fairly accurate. However, the main hubs and endpoints are often inside buildings, and not on the street. Since the drawings mostly run alongside streets, it is not always visible from the drawing that the cable enters the building, but if a cable is so close to a main hub, we need to assume that it is connected.

To estimate the logical data, the following assumptions were made:

- If the name of a cable indicates that it is a SAN cable, it is a SAN cable. Otherwise, it is a PAN cable.
- The length of a local cable (section), is the length of the drawing, i.e. the sum of the lengths of all line segments it consists of, scaled up to proportion using the scale of the map it was drawn on.
- If a local cable (PAN or SAN) runs within fifty meters of a main hub, it is connected to this main hub.
- If a local cable (PAN or SAN) runs within five meters of the endpoint of a local cable with the same name, it is connected to this cable.
- If a PAN cable runs within five meters of the endpoint of a SAN cable, this PAN cable is connected to this SAN cable.

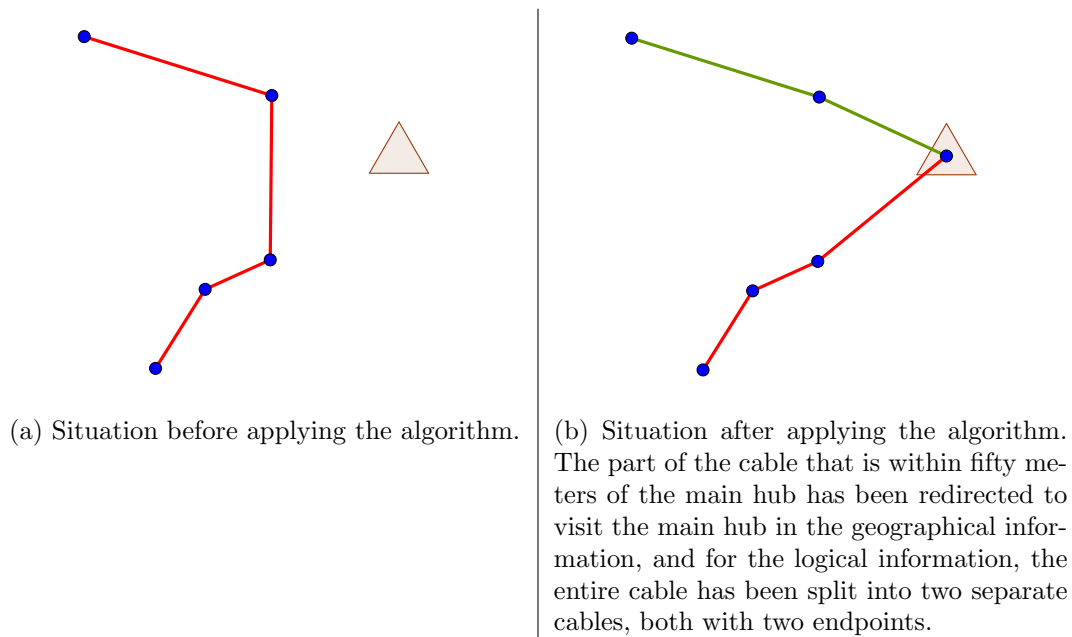


Figure A.1: Suppose the triangle represents a main hub and the string of line segments represents the geographical information of a local (PAN/SAN) cable running within fifty meters of this main hub. This figure shows the situation before and after applying the algorithm.

Because the drawings often included multiple sections of a cable including splits, and point-to-point cables were needed for our algorithms, the drawings had to be cut into sections, and then reconnected, taking the assumptions mentioned above into account.

A.2.1 Algorithm to interpret drawings

The algorithm to interpret the drawings of the local cables and estimate the logical data consists of two steps.

First, the algorithm walks along every line segment in every drawing. If this line segment is within five meters of the endpoint of a local cable with the same name of the cable it belongs to, this line segment is split into two line segments, both connected to this endpoint. This endpoint is then called a *local intersection point* for this cable. If the line segment belongs to a PAN cable and it is within five meters of the endpoint of a SAN cable, this line segment is split into two line segments, both connected to this endpoint. This endpoint is then called a *SAN intersection point*. If the line segment is within fifty meters of a main hub, this line segment is split into two line segments, both connected to this main hub.

Then, the algorithm walks along all cables that resulted from the previous step. If the cable is a SAN cable and ends on a SAN intersection point, or it is a PAN cable and ends on one of the SAN intersection points that this cable was connected to in the previous step, a node is created for this SAN intersection point (if one does not exist at this coordinate), and this cable is connected to this node.

If the cable ends on a local intersection point that this cable was connected to in the previous step, a node is created for this local intersection point (if one does not exist at this coordinate), and this cable is connected to this node.

If the cable ends on a main hub, it is connected to this main hub.

The lists for main hubs and SAN endpoints are stored as KD-trees [17] for fast access. See Algorithm 9 for a simplified version of the algorithm in pseudocode.

Algorithm 9 Simplified version of the data interpretation algorithm

```

1: function INTERPRETDRAWINGS(Drawings,  $G = (V, E)$ )
2:   endpoints  $\leftarrow \emptyset$  ▷ Dictionary
3:   sanEndpoints  $\leftarrow \emptyset$  ▷ KD-Tree
4:   for all Drawing  $\in$  Drawings do
5:     endpoints[Drawing.Name]  $\leftarrow$  endpoints[Drawing.Name]  $\cup$  Drawing.Endpoints
6:     if ISAN(Drawing) then
7:       sanEndpoints  $\leftarrow$  sanEndpoints  $\cup$  Drawing.Endpoints
8:     end if
9:   end for ▷ Step one
10:  sanIntersectionPoints  $\leftarrow \emptyset$ 
11:  for all Drawing  $\in$  Drawings do ▷ Step one
12:    for all  $(c_1, c_2) \in$  Drawing do ▷ Every line segment
13:      for all endpoint  $\in$  endpoints[Drawing.Name] do
14:        if DISTANCE( $(c_1, c_2), endpoint$ )  $<$  5meters then ▷ Shortest distance line segment
15:          cut Drawing into two new drawings: from  $c_1$  to endpoint, and from endpoint
16:          to  $c_2$ 
17:        end if
18:      end for
19:      if !ISAN(Drawing) then
20:        for all endpoint  $\in$  sanEndpoints do
21:          if DISTANCE( $(c_1, c_2), endpoint$ )  $<$  5meters then ▷ Shortest distance line segment
22:            cut Drawing into two new drawings: from  $c_1$  to endpoint, and from
23:            endpoint to  $c_2$ 
24:          end if
25:        end for
26:        for all mainHub  $\in$  mainHubs do
27:          if DISTANCE( $(c_1, c_2), mainHub$ )  $<$  50meters then ▷ Shortest distance line segment
28:            cut Drawing into two new drawings: from  $c_1$  to mainHub, and from mainHub
29:            to  $c_2$ 
30:          end if
31:        end for
32:      end for ▷ Step two
33:      E  $\leftarrow$  E  $\cup$  Drawing ▷ Updated set of drawings
34:      V  $\leftarrow$  V  $\cup$  Drawing.endpoints ▷ Add new cable to edges
35:    end for ▷ Add (new) nodes
36: end function

```

Appendix B

Details on the instances

This appendix discusses some properties of the instances that were used during the experiments (Chapter 5). The optimal value for every instance can be found in Table 5.1. Information about the number of virtual cables that need to be created for an instance and the sizes of the networks after preprocessing can be found in Table C.1.

S1P1SP

A real, simple single dark fiber order between two addresses in a suburban area. The result that was originally found by the consultants of KPN had a score of 88930.41, while the optimal value, before any preprocessing, is 88073.08.

T1P1SP

A real, simple single dark fiber order between two addresses in a large city. The result that was originally found by the consultants of KPN had a score of 15177.64, while the optimal value, before any preprocessing, is 10183.82.

L1P2SP

A real, dual dark fiber order between two addresses in small towns located in a rural area.

D1P1SP

A real, dual dark fiber order between two addresses in small towns located in a rural area.

H1P2SP-INN

A network dark fiber from one main hub to two other main hubs in a dense area.

K1P2SP-INN

A real, dual dark fiber between two main hubs in a dense area.

F2P4SP

A real order of two dual dark fibers in a large city. The result that was originally found by the consultants of KPN had a score of 85346101.01.

B1P2SP

A real, dual dark fiber order between two addresses. One of the addresses is in a small town, the other is in a relatively small city.

C1P2SP

A real, dual dark fiber order between two addresses. One of the addresses is in a small town, the other is in a small city.

D38P38SP

A real order of 38 single dark fibers. The addresses are spread all over the country.

D38P38SP-INN

The same order as D38P38SP, but only between the main hubs of the geographical areas the addresses of D38P38SP lie in.

A14P14SP

A set of fourteen single dark fiber problems, all from different locations in a rural area to the same main hub in a small town. This was not a dark fiber order, but was used to find the shortest distance from locations to the main hub for analytical purposes. Because of this, the number of fibers required for every subproblem is zero.

A14P14SP20

The same set as A14P14SP, but with the number of fibers for every subproblem set to twenty.

A14P28SP

This is A14P14SP turned into a set of dual dark fiber problems. The result is a set of dual dark fiber orders from fourteen locations in the same area and all to the same main hub. Dark fibers from a location to a main hub happen sometimes, for example if a client wants to be connected to the internet. The main hub is the gateway to the internet.

J14P28SP8-INN

A set of fourteen dual dark fiber problems between main hubs in the same (rural) area. What is interesting about this instance is that these are orders of eight fibers instead of the usual two fibers. This makes carefully watching the capacities of the cables more important.

D38P76SP

This is D38P38SP turned into a set of dual dark fiber problems.

D38P76SP-INN

This is D38P38SP-INN turned into a set of dual dark fiber problems.

I10P20SP

A set of ten dual dark fiber orders between two addresses in a large city.

G4P14SP

A real order consisting of:

- One non-closed dark fiber ring between six addresses, with one branch to a main hub.
- One non-closed dark fiber ring between four addresses.
- One closed dark fiber ring between four addresses.
- One single dark fiber between two addresses.

All addresses are within the same large urban area.

G4P14SP-INN

The same order as G4P14SP, but only between the main hubs of the geographical areas the addresses of G4P14SP lie in.

G3P8SP

The same order as G4P14SP, but without the non-closed dark fiber ring between six addresses.

G1P4SP

The closed dark fiber ring between four addresses, taken from G4P14SP.

Appendix C

Details on the experimental results

Table C.2: Number of nodes, cables and single point of failures per subproblem after the preprocessing steps that retain optimality (SL→FF→SP), and after the complete preprocessing chain (SL→HF→FF→SP). Percentages after the numbers for the chain that retains optimality show the relative difference between that chain and the full network, and percentages after the number for the complete chain show the relative difference between that chain and the chain that retains optimality. # *vc* represents the number of virtual cables.

Instance	# <i>vc</i>	SL→FF→SP			SL→HF→FF→SP		
		# nodes	# cables	#spofs	# nodes	# cables	#spofs
S1P1SP	161	7332 (-94%)	15202 (-94%)	0	605 (-92%)	1240 (-92%)	13108
T1P1SP	279	15487 (-87%)	35291 (-85%)	0	3122 (-80%)	7810 (-78%)	0
L1P2SP	49	2632 (-98%)	5191 (-98%)	43387 (-98%)	605 (-77%)	1240 (-76%)	13108 (-70%)
	49	2632 (-98%)	5191 (-98%)	43387 (-98%)	605 (-77%)	1240 (-76%)	13108 (-70%)
D1P2SP	45	1943 (-98%)	3870 (-98%)	32054 (-99%)	344 (-82%)	684 (-82%)	4146 (-87%)
	45	1943 (-98%)	3870 (-98%)	32054 (-99%)	344 (-82%)	684 (-82%)	4146 (-87%)
H1P2SP-INN	0	17645 (-85%)	39486 (-84%)	442338 (-84%)	113 (-99%)	395 (-99%)	2586 (-99%)
	0	18059 (-85%)	40317 (-83%)	442338 (-84%)	121 (-99%)	411 (-99%)	2586 (-99%)
K1P2SP-INN	0	17645 (-85%)	39486 (-84%)	439817 (-84%)	113 (-99%)	395 (-99%)	2358 (-99%)
	0	17645 (-85%)	39486 (-84%)	439817 (-84%)	113 (-99%)	395 (-99%)	2358 (-99%)
F2P4SP	845	19555 (-84%)	44140 (-82%)	478939 (-82%)	1370 (-93%)	3646 (-92%)	51378 (-89%)
	845	19555 (-84%)	44140 (-82%)	478939 (-82%)	1370 (-93%)	3646 (-92%)	51378 (-89%)
	845	19759 (-84%)	44629 (-81%)	487169 (-82%)	1071 (-95%)	3268 (-93%)	40422 (-92%)
	845	19759 (-84%)	44629 (-81%)	487169 (-82%)	1071 (-95%)	3268 (-93%)	40422 (-92%)
B1P2SP	68	3471 (-97%)	7123 (-97%)	62224 (-98%)	371 (-89%)	924 (-87%)	14665 (-76%)
	68	3471 (-97%)	7123 (-97%)	62224 (-98%)	371 (-89%)	924 (-87%)	14665 (-76%)
C1P2SP	126	5560 (-95%)	11786 (-95%)	126350 (-95%)	611 (-89%)	1266 (-89%)	6913 (-95%)
	126	5560 (-95%)	11786 (-95%)	126350 (-95%)	611 (-89%)	1266 (-89%)	6913 (-95%)
D38P38SP	140	4178 (-97%)	8627 (-96%)	0	470 (-89%)	1032 (-88%)	0
	338	3576 (-97%)	7567 (-97%)	0	371 (-90%)	924 (-88%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	167	1936 (-98%)	4176 (-98%)	0	368 (-81%)	768 (-82%)	0
	221	17091 (-86%)	37782 (-84%)	0	900 (-95%)	2298 (-94%)	0
	177	5566 (-95%)	12404 (-95%)	0	295 (-95%)	603 (-95%)	0
	77	4437 (-96%)	10662 (-96%)	0	283 (-94%)	710 (-93%)	0
	234	10528 (-91%)	23625 (-90%)	0	723 (-93%)	1931 (-92%)	0
	64	1603 (-99%)	3612 (-98%)	0	285 (-82%)	678 (-81%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	84	799 (-99%)	1613 (-99%)	0	246 (-69%)	541 (-66%)	0
	83	2106 (-98%)	4281 (-98%)	0	341 (-84%)	733 (-83%)	0
	263	6860 (-94%)	14518 (-94%)	0	580 (-92%)	1155 (-92%)	0
	263	6625 (-95%)	13842 (-94%)	0	318 (-95%)	681 (-95%)	0
	140	4178 (-97%)	8627 (-96%)	0	470 (-89%)	1032 (-88%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	47	729 (-99%)	1370 (-99%)	0	242 (-67%)	412 (-70%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0

Continued on next page

Appendix C. Details on the experimental results

Instance	# vc	SL→FF→SP			SL→HF→FF→SP		
		# nodes	# cables	#spofs	# nodes	# cables	#spofs
	195	399 (-100%)	972 (-100%)	0	200 (-50%)	454 (-53%)	0
	11	14411 (-88%)	33062 (-86%)	0	327 (-98%)	714 (-98%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	198	2976 (-98%)	6026 (-97%)	0	285 (-90%)	522 (-91%)	0
	234	9752 (-92%)	21868 (-91%)	0	319 (-97%)	792 (-96%)	0
	40	2188 (-98%)	4403 (-98%)	0	321 (-85%)	614 (-86%)	0
	243	17739 (-85%)	39152 (-84%)	0	742 (-96%)	1899 (-95%)	0
	79	1952 (-98%)	3918 (-98%)	0	344 (-82%)	684 (-83%)	0
	234	11452 (-91%)	26572 (-89%)	0	306 (-97%)	785 (-97%)	0
	112	1040 (-99%)	2121 (-99%)	0	184 (-82%)	354 (-83%)	0
	217	6663 (-94%)	14005 (-94%)	0	485 (-93%)	1151 (-92%)	0
	119	4463 (-96%)	9054 (-96%)	0	247 (-94%)	545 (-94%)	0
	210	15591 (-87%)	35264 (-85%)	0	289 (-98%)	806 (-98%)	0
	47	894 (-99%)	1708 (-99%)	0	255 (-71%)	492 (-71%)	0
	87	5542 (-95%)	13787 (-94%)	0	2 (-100%)	0 (-100%)	0
	47	894 (-99%)	1708 (-99%)	0	255 (-71%)	492 (-71%)	0
	333	1907 (-98%)	4230 (-98%)	0	297 (-84%)	629 (-85%)	0
	210	15591 (-87%)	35264 (-85%)	0	289 (-98%)	806 (-98%)	0
	243	2112 (-98%)	4402 (-98%)	0	198 (-91%)	450 (-90%)	0
	8	2410 (-98%)	4811 (-98%)	0	65 (-97%)	120 (-98%)	0
D38P38SP-INN	0	4117 (-97%)	8393 (-96%)	0	63 (-98%)	134 (-98%)	0
	0	3500 (-97%)	7123 (-97%)	0	63 (-98%)	96 (-99%)	0
	0	3625 (-97%)	7366 (-97%)	0	66 (-98%)	103 (-99%)	0
	0	1871 (-98%)	3903 (-98%)	0	37 (-98%)	63 (-98%)	0
	0	17026 (-86%)	37448 (-84%)	0	136 (-99%)	437 (-99%)	0
	0	5505 (-95%)	12121 (-95%)	0	50 (-99%)	163 (-99%)	0
	0	4413 (-96%)	10546 (-96%)	0	53 (-99%)	104 (-99%)	0
	0	10409 (-91%)	23180 (-90%)	0	94 (-99%)	227 (-99%)	0
	0	1576 (-99%)	3499 (-99%)	0	35 (-98%)	64 (-98%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	762 (-99%)	1472 (-99%)	0	18 (-98%)	25 (-98%)	0
	0	2083 (-98%)	4158 (-98%)	0	39 (-98%)	66 (-98%)	0
	0	6773 (-94%)	14207 (-94%)	0	82 (-99%)	158 (-99%)	0
	0	6518 (-95%)	13404 (-94%)	0	74 (-99%)	133 (-99%)	0
	0	4117 (-97%)	8393 (-96%)	0	63 (-98%)	134 (-98%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	712 (-99%)	1295 (-99%)	0	30 (-96%)	41 (-97%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	337 (-100%)	677 (-100%)	0	18 (-95%)	21 (-97%)	0
	0	14551 (-88%)	33339 (-86%)	0	94 (-99%)	312 (-99%)	0
	0	3770 (-97%)	7353 (-97%)	0	53 (-99%)	100 (-99%)	0
	0	2910 (-98%)	5708 (-98%)	0	40 (-99%)	65 (-99%)	0
	0	9653 (-92%)	21462 (-91%)	0	86 (-99%)	205 (-99%)	0
	0	2165 (-98%)	4325 (-98%)	0	39 (-98%)	57 (-99%)	0
	0	17662 (-85%)	38775 (-84%)	0	145 (-99%)	455 (-99%)	0
	0	1933 (-98%)	3815 (-98%)	0	40 (-98%)	87 (-98%)	0
	0	11385 (-91%)	26313 (-89%)	0	99 (-99%)	258 (-99%)	0
	0	997 (-99%)	1947 (-99%)	0	33 (-97%)	61 (-97%)	0
	0	6558 (-95%)	13614 (-94%)	0	86 (-99%)	152 (-99%)	0
	0	4449 (-96%)	8927 (-96%)	0	63 (-99%)	148 (-98%)	0
	0	15531 (-87%)	34951 (-85%)	0	102 (-99%)	361 (-99%)	0
	0	881 (-99%)	1648 (-99%)	0	35 (-96%)	47 (-97%)	0
	0	14299 (-88%)	33656 (-86%)	0	91 (-99%)	301 (-99%)	0
	0	881 (-99%)	1648 (-99%)	0	35 (-96%)	47 (-97%)	0
	0	1791 (-99%)	3706 (-98%)	0	31 (-98%)	56 (-98%)	0
	0	15531 (-87%)	34951 (-85%)	0	102 (-99%)	361 (-99%)	0
	0	2034 (-98%)	4032 (-98%)	0	39 (-98%)	58 (-99%)	0
	0	2407 (-98%)	4800 (-98%)	0	42 (-98%)	84 (-98%)	0
A14P14SP	541	8452 (-93%)	18396 (-92%)	0	265 (-97%)	488 (-97%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	541	8697 (-93%)	18999 (-92%)	0	341 (-96%)	667 (-96%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	541	8189 (-93%)	17886 (-93%)	0	410 (-95%)	856 (-95%)	0
	541	6182 (-95%)	13654 (-94%)	0	121 (-98%)	242 (-98%)	0
	541	7183 (-94%)	15700 (-93%)	0	139 (-98%)	289 (-98%)	0
	541	8547 (-93%)	18686 (-92%)	0	264 (-97%)	472 (-97%)	0
	541	7329 (-94%)	16123 (-93%)	0	143 (-98%)	314 (-98%)	0
	541	6626 (-95%)	14571 (-94%)	0	129 (-98%)	259 (-98%)	0
	541	6182 (-95%)	13654 (-94%)	0	121 (-98%)	242 (-98%)	0
	541	8255 (-93%)	17995 (-92%)	0	595 (-93%)	1334 (-93%)	0
	541	8638 (-93%)	18858 (-92%)	0	340 (-96%)	667 (-96%)	0
	541	8205 (-93%)	17902 (-93%)	0	563 (-93%)	1315 (-93%)	0
A14P14SP20	541	3461 (-97%)	7735 (-97%)	0	144 (-96%)	225 (-97%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	541	3609 (-97%)	8032 (-97%)	0	219 (-94%)	391 (-95%)	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0

Continued on next page

Appendix C. Details on the experimental results

Instance	# vc	SL→FF→SP			SL→HF→FF→SP		
		# nodes	# cables	#spofs	# nodes	# cables	#spofs
	541	3161 (-97%)	7165 (-97%)	0	300 (-91%)	611 (-91%)	0
	541	1953 (-98%)	4629 (-98%)	0	41 (-98%)	60 (-99%)	0
	541	2771 (-98%)	6328 (-97%)	0	50 (-98%)	79 (-99%)	0
	541	3525 (-97%)	7854 (-97%)	0	143 (-96%)	204 (-97%)	0
	541	2495 (-98%)	5645 (-98%)	0	41 (-98%)	80 (-99%)	0
	541	2108 (-98%)	4950 (-98%)	0	42 (-98%)	61 (-99%)	0
	541	1956 (-98%)	4633 (-98%)	0	41 (-98%)	60 (-99%)	0
	541	3311 (-97%)	7436 (-97%)	0	478 (-86%)	1077 (-86%)	0
	541	3584 (-97%)	7985 (-97%)	0	219 (-94%)	395 (-95%)	0
	541	3207 (-97%)	7264 (-97%)	0	447 (-86%)	1059 (-85%)	0
A14P28SP	541	7708 (-94%)	16858 (-93%)	178561 (-93%)	204 (-97%)	391 (-98%)	1749 (-99%)
	541	7708 (-94%)	16858 (-93%)	178561 (-93%)	204 (-97%)	391 (-98%)	1749 (-99%)
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	541	7811 (-94%)	17036 (-93%)	180525 (-93%)	278 (-96%)	561 (-97%)	3464 (-98%)
	541	7811 (-94%)	17036 (-93%)	180525 (-93%)	278 (-96%)	561 (-97%)	3464 (-98%)
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	541	7486 (-94%)	16401 (-93%)	173753 (-94%)	355 (-95%)	769 (-95%)	2850 (-98%)
	541	7486 (-94%)	16401 (-93%)	173753 (-94%)	355 (-95%)	769 (-95%)	2850 (-98%)
	541	5105 (-96%)	11551 (-95%)	129760 (-95%)	82 (-98%)	169 (-99%)	772 (-99%)
	541	5105 (-96%)	11551 (-95%)	129760 (-95%)	82 (-98%)	169 (-99%)	772 (-99%)
	541	6665 (-94%)	14575 (-94%)	156318 (-94%)	96 (-99%)	213 (-99%)	985 (-99%)
	541	6665 (-94%)	14575 (-94%)	156318 (-94%)	96 (-99%)	213 (-99%)	985 (-99%)
	541	7748 (-94%)	16930 (-93%)	179154 (-93%)	202 (-97%)	369 (-98%)	1594 (-99%)
	541	7748 (-94%)	16930 (-93%)	179154 (-93%)	202 (-97%)	369 (-98%)	1594 (-99%)
	541	6376 (-95%)	14062 (-94%)	153064 (-94%)	87 (-99%)	208 (-99%)	903 (-99%)
	541	6376 (-95%)	14062 (-94%)	153064 (-94%)	87 (-99%)	208 (-99%)	903 (-99%)
	541	5730 (-95%)	12840 (-95%)	144055 (-95%)	86 (-98%)	179 (-99%)	798 (-99%)
	541	5730 (-95%)	12840 (-95%)	144055 (-95%)	86 (-98%)	179 (-99%)	798 (-99%)
	541	5107 (-96%)	11553 (-95%)	129765 (-95%)	82 (-98%)	169 (-99%)	772 (-99%)
	541	5107 (-96%)	11553 (-95%)	129765 (-95%)	82 (-98%)	169 (-99%)	772 (-99%)
	541	7569 (-94%)	16609 (-93%)	176483 (-94%)	534 (-93%)	1237 (-93%)	6292 (-96%)
	541	7569 (-94%)	16609 (-93%)	176483 (-94%)	534 (-93%)	1237 (-93%)	6292 (-96%)
	541	7791 (-94%)	17002 (-93%)	179577 (-93%)	277 (-96%)	559 (-97%)	3372 (-98%)
	541	7791 (-94%)	17002 (-93%)	179577 (-93%)	277 (-96%)	559 (-97%)	3372 (-98%)
	541	7511 (-94%)	16456 (-93%)	174022 (-94%)	501 (-93%)	1216 (-93%)	5995 (-97%)
	541	7511 (-94%)	16456 (-93%)	174022 (-94%)	501 (-93%)	1216 (-93%)	5995 (-97%)
J14P28SP8-INN	0	3972 (-97%)	8436 (-96%)	93996 (-97%)	37 (-99%)	52 (-99%)	91 (-100%)
	0	3972 (-97%)	8436 (-96%)	93996 (-97%)	37 (-99%)	52 (-99%)	91 (-100%)
	0	3390 (-97%)	7157 (-97%)	81328 (-97%)	30 (-99%)	41 (-99%)	72 (-100%)
	0	3390 (-97%)	7157 (-97%)	81328 (-97%)	30 (-99%)	41 (-99%)	72 (-100%)
	0	3385 (-97%)	7096 (-97%)	80030 (-97%)	30 (-99%)	41 (-99%)	72 (-100%)
	0	3385 (-97%)	7096 (-97%)	80030 (-97%)	30 (-99%)	41 (-99%)	72 (-100%)
	0	4009 (-97%)	8523 (-96%)	94546 (-97%)	38 (-99%)	56 (-99%)	97 (-100%)
	0	4009 (-97%)	8523 (-96%)	94546 (-97%)	38 (-99%)	56 (-99%)	97 (-100%)
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	2 (-100%)	0 (-100%)	0	2 (-0%)	0	0
	0	4546 (-96%)	9669 (-96%)	107052 (-96%)	36 (-99%)	51 (-99%)	87 (-100%)
	0	4546 (-96%)	9669 (-96%)	107052 (-96%)	36 (-99%)	51 (-99%)	87 (-100%)
	0	3610 (-97%)	7560 (-97%)	83805 (-97%)	24 (-99%)	27 (-100%)	50 (-100%)
	0	3610 (-97%)	7560 (-97%)	83805 (-97%)	24 (-99%)	27 (-100%)	50 (-100%)
	0	5049 (-96%)	10463 (-96%)	107918 (-96%)	48 (-99%)	68 (-99%)	126 (-100%)
	0	5049 (-96%)	10463 (-96%)	107918 (-96%)	48 (-99%)	68 (-99%)	126 (-100%)
	0	4568 (-96%)	9472 (-96%)	100364 (-96%)	41 (-99%)	60 (-99%)	109 (-100%)
	0	4568 (-96%)	9472 (-96%)	100364 (-96%)	41 (-99%)	60 (-99%)	109 (-100%)
	0	5456 (-95%)	11550 (-95%)	120218 (-96%)	46 (-99%)	67 (-99%)	132 (-100%)
	0	5456 (-95%)	11550 (-95%)	120218 (-96%)	46 (-99%)	67 (-99%)	132 (-100%)
	0	2132 (-98%)	4566 (-98%)	52054 (-98%)	16 (-99%)	16 (-100%)	27 (-100%)
	0	2132 (-98%)	4566 (-98%)	52054 (-98%)	16 (-99%)	16 (-100%)	27 (-100%)
I10P20SP	797	16200 (-87%)	37172 (-84%)	525982 (-81%)	1670 (-90%)	4656 (-87%)	77713 (-85%)
	797	16200 (-87%)	37172 (-84%)	525982 (-81%)	1670 (-90%)	4656 (-87%)	77713 (-85%)
	797	14724 (-88%)	34045 (-86%)	491194 (-82%)	1928 (-87%)	4879 (-86%)	49817 (-90%)
	797	14724 (-88%)	34045 (-86%)	491194 (-82%)	1928 (-87%)	4879 (-86%)	49817 (-90%)
	797	16409 (-86%)	37553 (-84%)	528025 (-81%)	2155 (-87%)	5571 (-85%)	85719 (-84%)
	797	16409 (-86%)	37553 (-84%)	528025 (-81%)	2155 (-87%)	5571 (-85%)	85719 (-84%)
	797	14708 (-88%)	34032 (-86%)	492137 (-82%)	2130 (-86%)	4566 (-87%)	42943 (-91%)
	797	14708 (-88%)	34032 (-86%)	492137 (-82%)	2130 (-86%)	4566 (-87%)	42943 (-91%)
	797	15065 (-88%)	34718 (-85%)	497946 (-82%)	1744 (-88%)	3522 (-90%)	21594 (-96%)
	797	15065 (-88%)	34718 (-85%)	497946 (-82%)	1744 (-88%)	3522 (-90%)	21594 (-96%)

Continued on next page

Appendix C. Details on the experimental results

Instance	# vc	SL→FF→SP			SL→HF→FF→SP		
		# nodes	# cables	#spofs	# nodes	# cables	#spofs
	797	15349 (-87%)	35441 (-85%)	511114 (-81%)	1262 (-92%)	2632 (-93%)	18049 (-96%)
	797	15349 (-87%)	35441 (-85%)	511114 (-81%)	1262 (-92%)	2632 (-93%)	18049 (-96%)
	797	14881 (-88%)	34338 (-86%)	493167 (-82%)	1209 (-92%)	2685 (-92%)	19444 (-96%)
	797	14881 (-88%)	34338 (-86%)	493167 (-82%)	1209 (-92%)	2685 (-92%)	19444 (-96%)
	797	15561 (-87%)	35887 (-85%)	515081 (-81%)	1269 (-92%)	3240 (-91%)	36753 (-93%)
	797	15561 (-87%)	35887 (-85%)	515081 (-81%)	1269 (-92%)	3240 (-91%)	36753 (-93%)
	797	15374 (-87%)	35517 (-85%)	512674 (-81%)	955 (-94%)	2396 (-93%)	34598 (-93%)
	797	15374 (-87%)	35517 (-85%)	512674 (-81%)	955 (-94%)	2396 (-93%)	34598 (-93%)
	797	13593 (-89%)	32023 (-87%)	476439 (-82%)	662 (-95%)	1615 (-95%)	11690 (-98%)
797	13593 (-89%)	32023 (-87%)	476439 (-82%)	662 (-95%)	1615 (-95%)	11690 (-98%)	
G4P14SP-INN	0	15001 (-88%)	34036 (-86%)	505205 (-81%)	115 (-99%)	357 (-99%)	2491 (-100%)
	0	14663 (-88%)	33167 (-86%)	497723 (-82%)	106 (-99%)	337 (-99%)	2443 (-100%)
	0	14220 (-88%)	32318 (-87%)	476988 (-82%)	95 (-99%)	310 (-99%)	2352 (-100%)
	0	16127 (-87%)	36860 (-85%)	509282 (-81%)	119 (-99%)	376 (-99%)	2473 (-100%)
	0	15322 (-87%)	34915 (-85%)	516167 (-81%)	110 (-99%)	352 (-99%)	2472 (-100%)
	0	12608 (-90%)	28971 (-88%)	410775 (-85%)	83 (-99%)	271 (-99%)	2096 (-99%)
	0	14746 (-88%)	32991 (-86%)	471312 (-83%)	105 (-99%)	328 (-99%)	2619 (-99%)
	0	14786 (-88%)	33066 (-86%)	470883 (-83%)	105 (-99%)	328 (-99%)	2619 (-99%)
	0	14901 (-88%)	33804 (-86%)	454706 (-83%)	115 (-99%)	364 (-99%)	2347 (-99%)
	0	14815 (-88%)	33568 (-86%)	443369 (-84%)	114 (-99%)	352 (-99%)	2316 (-99%)
	0	14625 (-88%)	33568 (-86%)	477423 (-82%)	102 (-99%)	340 (-99%)	2576 (-99%)
	0	12487 (-90%)	29843 (-88%)	374033 (-86%)	77 (-99%)	260 (-99%)	1901 (-99%)
	0	13692 (-89%)	31752 (-87%)	465479 (-83%)	88 (-99%)	305 (-99%)	2483 (-99%)
	0	14472 (-88%)	33271 (-86%)	0	101 (-99%)	339 (-99%)	0
	G4P14SP	491	15634 (-87%)	35578 (-85%)	502723 (-81%)	1447 (-91%)	3941 (-89%)
491		14927 (-88%)	34075 (-86%)	498207 (-82%)	1735 (-88%)	4729 (-86%)	68991 (-86%)
383		13955 (-88%)	32132 (-87%)	470945 (-83%)	691 (-95%)	1859 (-94%)	41837 (-91%)
383		16106 (-87%)	37133 (-84%)	498636 (-82%)	518 (-97%)	1323 (-96%)	31110 (-94%)
383		15300 (-87%)	35057 (-85%)	504971 (-81%)	696 (-95%)	1478 (-96%)	11364 (-98%)
243		12550 (-90%)	29003 (-88%)	403921 (-85%)	459 (-96%)	1237 (-96%)	21494 (-95%)
383		14886 (-88%)	33614 (-86%)	470343 (-83%)	1109 (-93%)	2680 (-92%)	43836 (-91%)
383		14725 (-88%)	33321 (-86%)	469034 (-83%)	881 (-94%)	2283 (-93%)	51130 (-89%)
491		14952 (-88%)	34130 (-86%)	453078 (-83%)	995 (-93%)	2136 (-94%)	12501 (-97%)
491		15015 (-88%)	34244 (-86%)	437694 (-84%)	1225 (-92%)	2536 (-93%)	17591 (-96%)
351		14398 (-88%)	33423 (-86%)	473165 (-83%)	644 (-96%)	1466 (-96%)	10156 (-98%)
261		11306 (-91%)	27057 (-89%)	344916 (-87%)	340 (-97%)	817 (-97%)	5569 (-98%)
351		13553 (-89%)	31719 (-87%)	460991 (-83%)	920 (-93%)	1934 (-94%)	14200 (-97%)
351		13816 (-89%)	32186 (-87%)	0	641 (-95%)	1310 (-96%)	0
G3P8SP		198	14815 (-88%)	33310 (-86%)	469942 (-83%)	1109 (-93%)	2680 (-92%)
	198	14664 (-88%)	33035 (-86%)	468782 (-83%)	881 (-94%)	2283 (-93%)	51130 (-89%)
	306	14891 (-88%)	33844 (-86%)	452677 (-83%)	995 (-93%)	2136 (-94%)	12501 (-97%)
	306	14954 (-88%)	33958 (-86%)	437441 (-84%)	1225 (-92%)	2536 (-93%)	17591 (-96%)
	166	14337 (-88%)	33137 (-86%)	472912 (-83%)	644 (-96%)	1466 (-96%)	10156 (-98%)
	166	11255 (-91%)	26838 (-89%)	344114 (-87%)	340 (-97%)	817 (-97%)	5569 (-98%)
	166	13486 (-89%)	31422 (-87%)	460698 (-83%)	920 (-93%)	1934 (-94%)	14200 (-97%)
	166	13755 (-89%)	31900 (-87%)	0	641 (-95%)	1310 (-96%)	0
G1P4SP	151	14895 (-88%)	33695 (-86%)	437425 (-84%)	1225 (-92%)	2536 (-92%)	17591 (-96%)
	151	14332 (-88%)	33114 (-86%)	472896 (-83%)	644 (-96%)	1466 (-96%)	10156 (-98%)
	151	11250 (-91%)	26815 (-89%)	344098 (-87%)	340 (-97%)	817 (-97%)	5569 (-98%)
	151	13481 (-89%)	31399 (-87%)	460682 (-83%)	920 (-93%)	1934 (-94%)	14200 (-97%)

Table C.3: Simulated annealing experiments grouped by instance, including confidence intervals.

Instance	α	$Q/\#\text{sp}$	Randomness	#	Score avg.	Confidence 95% \pm
L1P2SP	.95	10	0	30	4171288.54	0
L1P2SP	.95	10	.1	30	4045263.19	0
L1P2SP	.95	10	.2	30	3947165.30	200641.23
L1P2SP	.95	10	.3	30	3932618.75	200300.05
L1P2SP	.95	10	.4	30	3869577.18	207219.40
L1P2SP	.95	10	.5	30	3846499.86	205296.90
L1P2SP	.95	10	.6	30	3464308.36	425948.80
L1P2SP	.95	10	.7	30	3030554.05	518138.41
L1P2SP	.95	10	.8	30	1907058.66	454323.44
L1P2SP	.95	10	.9	30	1635978.90	202352.81
L1P2SP	.95	10	1	30	4735472.71	683899.24
L1P2SP	.95	20	.3	30	3726711.91	332878.04

Continued on next page

Appendix C. Details on the experimental results

Instance	α	Q/#sp	Randomness	#	Score avg.	Confidence 95% \pm
L1P2SP	.95	40	.3	30	3830759.17	203887.17
D1P2SP	.95	10	0	30	27142790.82	0
D1P2SP	.95	10	.1	30	27142790.82	0
D1P2SP	.95	10	.2	30	27142790.82	0
D1P2SP	.95	10	.3	30	27142790.82	0
D1P2SP	.95	10	.4	30	27142790.82	0
D1P2SP	.95	10	.5	30	27142790.82	0
D1P2SP	.95	10	.6	30	27142790.82	0
D1P2SP	.95	10	.7	30	27142790.82	0
D1P2SP	.95	10	.8	30	27142790.82	0
D1P2SP	.95	10	.9	30	27142790.82	0
D1P2SP	.95	10	1	30	27142790.82	0
D1P2SP	.95	20	.3	30	27142790.82	0
D1P2SP	.95	40	.3	30	27142790.82	0
F2P4SP	.95	10	0	30	79905.21	0
F2P4SP	.95	10	.1	30	74500.45	21.27
F2P4SP	.95	10	.2	30	74510.91	0.10
F2P4SP	.95	10	.3	30	74469.12	46.28
F2P4SP	.95	10	.4	30	74374.74	63.37
F2P4SP	.95	10	.5	30	74352.23	109.47
F2P4SP	.95	10	.6	30	74384.85	131.67
F2P4SP	.95	10	.7	30	76825.83	1066.97
F2P4SP	.95	10	.8	30	79905.21	0
F2P4SP	.95	10	.9	30	79905.21	0
F2P4SP	.95	10	1	30	79905.21	0
F2P4SP	.95	20	.3	30	74418.73	54.47
F2P4SP	.95	40	.3	30	74386.14	58.07
B1P2SP	.95	10	0	30	415117175.92	61249.38
B1P2SP	.95	10	.1	30	414925607.59	150223.95
B1P2SP	.95	10	.2	30	414826837.51	28024.94
B1P2SP	.95	10	.3	30	414880421.92	146220.62
B1P2SP	.95	10	.4	30	414808844.26	4150.78
B1P2SP	.95	10	.5	30	414807938.01	4092.88
B1P2SP	.95	10	.6	30	414813238.23	7080.43
B1P2SP	.95	10	.7	30	414814863.83	6051.47
B1P2SP	.95	10	.8	30	414842987.68	21874.87
B1P2SP	.95	10	.9	30	415419111.54	336276.15
B1P2SP	.95	10	1	30	421098534.98	1752122.86
B1P2SP	.95	20	.3	30	414807376.22	2987.49
B1P2SP	.95	40	.3	30	414805915.23	0.43
A14P14SP20	.95	10	0	30	30001280146.73	0
A14P14SP20	.95	10	.1	30	30001280146.73	0
A14P14SP20	.95	10	.2	30	30001280146.73	0
A14P14SP20	.95	10	.3	30	30001280146.73	0
A14P14SP20	.95	10	.4	30	30001280146.73	0
A14P14SP20	.95	10	.5	30	30001280146.73	0
A14P14SP20	.95	10	.6	30	30001280146.73	0
A14P14SP20	.95	10	.7	30	30001280146.73	0
A14P14SP20	.95	10	.8	30	30001280146.73	0
A14P14SP20	.95	10	.9	30	30001280146.73	0
A14P14SP20	.95	10	1	30	30001280146.73	0
A14P14SP20	.95	20	.3	30	30001280146.73	0
A14P14SP20	.95	40	.3	30	30001280146.73	0
A14P28SP	.95	10	0	30	22449848709.27	0
A14P28SP	.95	10	.1	30	22448683373.50	425853.23
A14P28SP	.95	10	.2	30	22447293548.75	6136.54
A14P28SP	.95	10	.3	30	22447288141.87	3642.94

Continued on next page

Appendix C. Details on the experimental results

Instance	α	Q/#sp	Randomness	#	Score avg.	Confidence 95% \pm
A14P28SP	.95	10	.4	30	22447284472.21	669.14
A14P28SP	.95	10	.5	30	22447287038.08	4051.37
A14P28SP	.95	10	.6	30	22447288301.84	11687.18
A14P28SP	.95	10	.7	30	22447382211.06	35018.87
A14P28SP	.95	10	.8	30	22447548899.17	26269.89
A14P28SP	.95	10	.9	30	22461734296.72	2652649.19
A14P28SP	.95	10	1	30	22474712553.87	3000167.82
A14P28SP	.95	20	.3	30	22447286702.13	3257.64
A14P28SP	.95	40	.3	30	22447282633.79	2533.58
J14P28SP8-INN	.95	10	.3	30	75506508663.76	0
J14P28SP8-INN	.95	20	.3	30	75506508663.76	0
J14P28SP8-INN	.95	40	.3	30	75506508663.76	0
D38P76SP	.95	10	.3	30	70903011146.13	804215.73
D38P76SP	.95	20	.3	30	70902780373.09	909739.35
D38P76SP	.95	40	.3	30	70901514575.18	609735.58
D38P76SP-INN	.95	10	.3	30	41639374523.60	39092499.48
D38P76SP-INN	.95	20	.3	30	41551910828.57	48323985.37
D38P76SP-INN	.95	40	.3	30	41555204408.31	47307478.93
I10P20SP	.95	10	.3	30	7532797.06	59451.27
I10P20SP	.95	20	.3	30	7482340.70	52670.17
I10P20SP	.95	40	.3	30	7388896.91	2153.64
G4P14SP-INN	.95	10	0	30	1894847.97	0
G4P14SP-INN	.95	10	.1	30	716463.81	223769.83
G4P14SP-INN	.95	10	.2	30	422737.92	707.61
G4P14SP-INN	.95	10	.3	30	423478.21	1609.66
G4P14SP-INN	.95	10	.4	30	475915.45	100104.90
G4P14SP-INN	.95	10	.5	30	479684.58	99847.01
G4P14SP-INN	.95	10	.6	30	635568.14	188113.26
G4P14SP-INN	.95	10	.7	30	523550.11	102839.18
G4P14SP-INN	.95	10	.8	30	776193.53	201315.12
G4P14SP-INN	.95	10	.9	30	782221.98	222777.01
G4P14SP-INN	.95	10	1	30	727759.94	221668.06
G4P14SP-INN	.95	20	.3	30	423722.83	1969.36
G4P14SP-INN	.95	40	.3	30	423361.56	1778.67
G4P14SP	.95	10	.3	30	4207407.81	31930.45
G4P14SP	.95	20	.3	30	4185432.56	14113.10
G4P14SP	.95	40	.3	30	4185694.01	15522.24
G3P8SP	.95	10	.3	30	2490502.96	13166.96
G3P8SP	.95	20	.3	30	2500119.23	45315.62
G3P8SP	.95	40	.3	30	2475050.56	2272.84
G1P4SP	.95	10	0	30	3567107.21	0
G1P4SP	.95	10	.1	30	2081056.49	15981.58
G1P4SP	.95	10	.2	30	1954407.65	45888.64
G1P4SP	.95	10	.3	30	1832073.08	26278.80
G1P4SP	.95	10	.4	30	1820617.76	12370.42
G1P4SP	.95	10	.5	30	1969188.46	220213.14
G1P4SP	.95	10	.6	30	1859281.88	64310.17
G1P4SP	.95	10	.7	30	2060252.66	122735.35
G1P4SP	.95	10	.8	30	2338077.96	235561.64
G1P4SP	.95	10	.9	30	2977294.42	331389.52
G1P4SP	.95	10	1	30	3909119.72	119434.91
G1P4SP	.95	20	.3	30	1811367.25	6810.00
G1P4SP	.95	40	.3	30	1823961.92	27365.24

Appendix C. Details on the experimental results

Instance	# vc	SL→FF→SP			SL→HF→FF→SP		
		# nodes	# cables	#spofs	# nodes	# cables	#spofs
S1P1SP	161	7330	15202	0	556	1271	0
T1P1SP	279	15485	35291	0	3120	7810	0
L1P2SP	49	2630	5191	43387	603	1240	13108
D1P2SP	45	1941	3870	32054	342	648	4146
H1P2SP-INN	0	18057	40317	442338	119	411	2586
K1P2SP-INN	0	17643	39486	439817	111	395	2358
F2P4SP	845	19846	44809	448158	1490	4512	55066
B1P2SP	68	3469	7123	62224	369	924	14665
C1P2SP	126	5558	11786	126350	609	1266	6913
D38P38SP	1654	75224	164087	0	7995	18197	0
D38P38SP-INN	0	77973	167966	0	922	2170	0
A14P14SP	541	8783	19165	0	791	2112	0
A14P14SP20	541	3631	8115	0	670	1835	0
A14P28SP	541	7895	17198	181436	726	2004	9254
J14P28SP8-INN	0	6191	13069	143760	155	155	155
D38P76SP	1620	X	X	X	7995	18197	164047
D38P76SP-INN	0	X	X	X	922	2170	11395
I10P20SP	797	16415	37566	528152	5248	12864	148627
G4P14SP-INN	0	23702	52794	617626	190	569	3546
G4P14SP	491	23076	51647	603738	3755	9374	147881
G3P8SP	306	20579	45591	553507	2328	5231	68169
G1P4SP	151	19244	43012	473371	1532	3231	20206

Table C.1: Magnitude of the network after optimality preserving preprocessing, and after full preprocessing.

Instance	Constraints	Variables	# Int. edges	# Imp. spofs	Time (ms)
S1P1SP	1828	2478	0	0	196
T1P1SP	10931	15177	0	0	1123
L1P2SP	2930	5036	82	241	71416
D1P2SP	1484	2698	23	57	3467
H1P2SP-INN	711	1625	13	30	2214
K1P2SP-INN	663	1585	11	22	1651
F2P4SP	10776	26990	27	81	34115
B1P2SP	5118	5304	236	1727	2230586
C1P1SP	3090	5215	165	302	210590
D38P38SP	322054	50094	0	0	10413
D38P38SP-INN	37248	10374	0	0	1597
A14P14SP	13202	14145	0	0	1012
A14P14SP20	11231	8459	5	0	1417
A14P28SP	25562	29736	196	420	479823
J14P28SP8-INN	2148	2448	29	37	5252
D38P76SP	632603	200736	607	2647	44401382
D38P76SP-INN	74572	36708	181	421	348800
G4P14SP-INN	5894	10748	51	66	53579
G3P8SP	26783	37704	598	898	13345233
G1P4SP	1193	18333	442	854	6048068

Table C.4: Number of constraints, variables, integer edges, and important single point of failures to reach the optimum using the ILP solver, with no initial single point of failures.

Instance	#	(Avg.) score	Confidence 95% \pm	Time (ms)
S1P1SP		88030.41 (100%)		1477
T1P1SP		10183.82 (100%)		805
L1P2SP		956668.00 (100%)		62813
D1P2SP		27142790.82 (100%)		2479
H1P2SP-INN		15966 (100%)		1561
K1P2SP-INN		21984 (100%)		1234
F2P4SP		74198.87 (100%)		23351
B1P2SP	30	414805997.23 (100%)	167.43	163731
C1P2SP	30	503340110.13 (101%)	0	200871
D38P38SP		60003493745.49 (100%)		15565
D38P38SP-INN		30000862993.70 (100%)		2706
A14P14SP		20001205040.53 (100%)		1955
A14P14SP20		30000938037.37 (100%)		1207
A14P28SP		22444258457.01 (100%)		118365
J14P28SP-INN		64214584040.96 (100%)		10856
D38P76SP	30	70899974199.22 (100%)	1241849.27	3811078
D38P76SP-INN		41416301981.57 (100%)		48595
I10P20SP	30	17479264.67 (237%)	3580011.32	2958891
G4P14SP-INN		416203.60 (100%)		35899
G4P14SP	30	4204414.27 (101%)	30724.17	2278156
G3P8SP	30	2498448.66 (101%)	23914.93	1049235
G1P4SP	30	1830930.38 (102%)	28629.69	679811

Table C.5: Combined solver experiments. All experiments were performed using a randomness factor of 0.3. Instances for which no count and confidence interval is reported could be solved to optimality without simulated annealing.

Bibliography

- [1] S. Even, A. Itai, and A. Shamir, “On the complexity of time table and multi-commodity flow problems,” in *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, pp. 184–193, IEEE Computer Society, 1975.
- [2] R. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, pp. 85–103, Plenum Press, 1972.
- [3] M. I. Shamos and D. Hoey, “Geometric intersection problems,” in *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pp. 208–215, IEEE Computer Society, 1976.
- [4] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [5] J. W. Suurballe and R. E. Tarjan, “A quick method for finding shortest pairs of disjoint paths,” *Networks*, vol. 14, no. 2, pp. 325–336, 1984.
- [6] P. Kovács, “Minimum-cost flow algorithms: An experimental evaluation,” *Optimization Methods and Software*, vol. 30, no. 1, pp. 94–127, 2015.
- [7] U. Föbmeier and M. Kaufmann, “Algorithms and area bounds for nonplanar orthogonal drawings,” in *Graph Drawing*, vol. 1353 of *Lecture Notes in Computer Science*, pp. 134–145, Springer Berlin Heidelberg, 1997.
- [8] M. L. Fredman and R. E. Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms,” *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.
- [9] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.
- [10] P. Shaw, “Using constraint programming and local search methods to solve vehicle routing problems,” in *Principles and Practice of Constraint Programming — CP98*, vol. 1520 of *Lecture Notes in Computer Science*, pp. 417–431, Springer Berlin Heidelberg, 1998.

-
- [11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [12] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.
- [13] L. Ingber, "Adaptive simulated annealing (ASA): Lessons learned," *Control and Cybernetics*, vol. 25, no. 1, pp. 33–54, 1996.
- [14] *Gurobi Optimizer Reference Manual*, 2015. Gurobi Optimization, Inc., <http://www.gurobi.com>.
- [15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.
- [16] I. S. Pohl, "Bi-directional search," *Machine Intelligence*, vol. 6, pp. 127–140, 1971.
- [17] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.