Karlsruhe Institute of Technology

Experimental study on the Wind Farm Substation Cable Installation Problem

Master thesis of

Christian Schmitz

Department of Computer Science
Institute for Theoretical Computer Science

Reviewers:	Prof. Dr. Dorothea Wagner
	Prof. Dr. Peter Sanders
Advisors:	Dr. Tamara Mchedlidze
	Dr. Martin Nöllenburg
	Dr. Hung-I Yu

Research Period: January 1, 2014 – June 30, 2014

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Karlsruhe, *Christian Schmitz*
in June 2014

Abstract

The planning of wind farms exhibits several interesting problems. One of them is the minimum cost cable installation in wind farms. In this master thesis we regard a subproblem of it, namely the Substation Cable Installation Problem (SCIP). The SCIP deals with the minimum cable installation connecting wind turbines to a substation.

It is therefore defined as follows: Given a set of coordinates for a substation and for a set of turbines, the turbines' rating and a set of cable types, find the cost minimal cable installation such that all turbines are connected to the substation and the capacity constraints imposed by the installation are met.

For this problem we identify several approaches from the literature for the SCIP as well as for related problems. We have adapted and implemented a part of these identified approaches. The implementation contains nine algorithms and heuristics which are presented in the main part of this thesis. Among them there is also a new integer linear program formulation and a new heuristic. Furthermore, we evaluate and compare the implemented algorithms and heuristics.

Zusammenfassung

Die Planung von Windparks wirft einige interessante Probleme auf. Eines davon ist die kostenminimale Kabelinstallation in Windparks. In dieser Masterarbeit wird ein Teilproblem dazu genauer betrachtet, nämlich das Substation Cable Installation Problem (SCIP). Das SCIP sucht die kostenminimale Kabelinstallation in Windparks, bei dem Windenergieanlagen (WEA) an einem Umspannwerk angeschlossen werden sollen.

Es ist daher folgendermaßen definiert: Gegeben eine Menge von Koordinaten für ein Umspannwerk und für eine Menge von Turbinen, die Leistung der Turbinen und eine Menge von Kabeltypen, finde die kostenminimale Kabelinstallation, sodass alle Turbinen mit dem Umspannwerk verbunden sind und die Kapazitätsbeschränkungen der Kabelinstallation eingehalten werden.

Für dieses Problem werden in dieser Arbeit Lösungsansätze aus der Literatur zum SCIP und zu benachbarten Problemstellungen identifiziert. Ein Teil der identifizierten Ansätze sind im Rahmen dieser Arbeit angepasst und implementiert worden. Die Implementierung umfasst neun Algorithmen und Heuristiken, die im Hauptteil dieser Arbeit genauer beschrieben werden. Darunter befinden sich auch ein neues ganzzahliges Programm und eine neue Heuristik. Die implementierten Algorithmen und Heuristiken werden anschließend in Experimenten evaluiert und miteinander verglichen.

Acknowledgment

I worked on parts of this master thesis during my stay at the National Chung Hsing University (NCHU) and finished it back at the Karlsruhe Institute of Technology (KIT). The possibility to research on my thesis topic at the NCHU was given to me by Prof. Dr. Dorothea Wagner and Prof. Dr. Der-Tsai Lee. Therefore, I would like to thank Prof. Dr. Dorothea Wagner for providing me with the opportunity to write my thesis at her chair and to combine it with a research stay abroad. Furthermore, I would like to thank Prof. Dr. Der-Tsai Lee for hosting me at the NCHU and for letting me work on my thesis at his chair.

While working on this master thesis I got very good assistance from my advisors Dr. Tamara Mchedlidze, Dr. Martin Nöllenburg (both KIT) and Dr. Hung-I Yu (NCHU). I am thankful for the interesting topic they gave me, their helpful input and the many discussions for the thesis. Especially, I would like to thank Dr. Hung-I Yu for the long discussion on the integer linear program (ILP) for the Substation Cable Installation Problem (SCIP).

Karlsruhe,
in June 2014

Christian Schmitz

Contents

1. Introduction

In the last years the climate change and the nuclear disaster in Fukushima have lead to a rethinking about our energy generation. Consequently, renewable energy has gained more and more attention. Today many countries try to shift from fossil energy sources to renewable ones. For example the European Union aims to meet 20% of the EU's energy needs with renewable energy [7, p. 3].

Due to this the importance of renewable energy as a source for power generation increases. One of the renewable energy sources which are designated to replace current energy sources is wind energy. Germany for example aims to install additional 8 GW onshore and 10 GW offshore wind power [3, p. 2-3].

Wind power installation is also interesting from a computer science and operations research perspective because we can derive optimization problems from it which are hard to solve and therefore, provide an application area for algorithmic solution approaches. One of these problems for instance deals with the optimal placement of wind turbines and has gotten increasing attention in the literature lately [34, p. 28]. However, in this thesis we deal with another wind energy problem which optimizes the cable installation in wind farms, namely the Wind Farm Cable Installation Problem (WFCIP). To be more specific, we consider subproblem of the WFCIP called SCIP. Research focused on this problem lately as well.

Before we get into detail about the main problem of this thesis let us first regard the wind energy setting more in detail.

In order to use wind as an energy source we need wind turbines. The maximum power a wind turbine can generate is given by its rating. Nowadays turbine ratings at 2 or 3 MW are common. 5 MW machines have also been installed offshore already and turbines with even larger generation capacities are under development. [24, p. 18]

Generally, wind turbines are positioned together forming *wind farms*, also called *wind parks* [24, p. 143]. A wind farm can comprise from two to more than a hundred turbines [24, p. 143].

In order to make use of their generated power the wind turbines need to be connected to the power grid at some grid connection point. The grid connection point can be positioned at different systems of the power grid, namely the transmission system, the distribution system and feeder system [26, p. 433]. All three systems operate at a different current level. The transmission system has a high-voltage current between 110 kV and 765 kV, the distribution system a medium-voltage between 10 kV and 69 kV and the feeder system a low-voltage between 230 V and 690 V [26, p. 433].

Depending on the total turbine rating the wind farm has to be connected to different systems. The feeder system can only be used for a turbine with a rating up to 100 kW. Wind farms with a total rating of more than 10 - 15 MW need to be connected to a high-voltage system, i.e. the transmission system. Otherwise the turbines can be connected to the distribution system. Hau [19, p. 742]

However, a turbine normally generates power at a 690 V [24, p. 185]. Hence, the current of the turbines has to be transformed in order to meet the current at the grid connection point.

The transformation to the high-voltage level is not done at the substation directly, though. The generated power is rather collected at substations where it is transformed to the needed current level. From the substations the wind farm is then connected to the next grid connection point over a high-voltage export cable. [2, p. 1-2]

Following DUTTA [9] we call the system connecting the wind turbines with the substation the *collector system*. The collector system operates at a medium-voltage level between 10 and 30 kV in continental Europe [4, p. 570].[1] Therefore, the cables used to connect the wind turbines to the substation are medium-voltage cables.

The transformers at a substation and the available cables put some restriction on the connection of turbines to the substation. The turbines are connected to the substation via feeders (or circuits). The number of wind turbines on one feeder is restricted by the ampacity of the cables [5, p. 1]. Furthermore, the overall number of turbines which can be connected to the substation is limited as well. The actual number depends on the maximum power rating of the substation transformers [2, p. 3].

The objective of the WFCIP now is to optimize the cable installation of the collector system. This means that we look for a cost minimal installation of cables connecting all turbines to a substation such that the cable and substation constraints are met. The SCIP is the subproblem which optimizes the cable installation for one substation.

In this thesis we will look into this problem setting and focus on the SCIP. We identify, implement and experimentally evaluate approaches to solve this problem.

This thesis is structured as follows: In the next chapter we will define the WFCIP and the SCIP more formally in Section 2.1 and Section 2.2. We present the results of our literature search for solution approaches used for related problems of the SCIP in Section 2.3. In Chapter 3 we describe the approaches we use to solve the SCIP. The performance of those algorithms is then evaluated in Chapter 4. The thesis is summarized and concluded in Chapter 5.

[1] This is still a higher voltage than at the wind turbines. The turbines, though, have transformers for raising their current to the one of the collector system [4, p. 570].

2. Basics

2.1. The Wind Farm Cable Installation Problem

Berzan et al. [2] are one of the first who deal with the cable network design in wind farms from an algorithmic perspective. In their work they define the cable installation problem formally. We will derive our definition for the Wind Farm Cable Installation Problem (WFCIP) from their definition. Our definition, however, also includes a more restrictive requirement on the unsplittability of flow as presented by Hertz et al. [20]. Furthermore, we follow Hertz et al. [20] in allowing the installation of several cables between two sites and do not restrict it to one cable type only.

In the wind farm setting there are $|V|$ sites given as points which need to be connected using cable types given by the set K. The set of nodes consists of $|T|$ turbines, $|R|$ substations and $|S|$ intermediate points at which flow can be combined.

A turbine t^ν is a source with a certain weight b_ν which has to be sent to a substation r. The weight of the turbine corresponds to the power produced by it, i.e. its power rating. In the following we will assume that every turbine has the same weight. This way we can deal with capacities and flows in terms of the number of turbines and not in amperes or watts. We assume that the power produced at the turbines reaches the substation without any losses.

At the substations all the generated power is collected. Each substation r has a certain capacity of turbines μ_r which can be connected to it. This again relies on the rating of the substation and hence can be expressed in number of turbines as well.

From now on we define the set of turbines as $T = \{t^1, ..., t^{|T|}\}$ and the set of sinks as $R = \{r^1, ..., r^{|R|}\}$. The definition of set of sites, i.e. set of points, then looks the following: $V = \{u^1, ..., u^{|V|}\} = T \cup R$.

We assume that the points in V are given in the Euclidean plane and therefore, do not regard any terrain information as Berzan et al. [2] do. Hence, the distance between two sites $u^\nu, u^\iota \in V$ is given by $dist(u^\nu, u^\iota) = \sqrt{(u_1^\nu - u_1^\iota)^2 + (u_2^\nu - u_2^\iota)^2}$.

Between any two sites u^ν and u^ι we can install copies of $|K|$ cable types given by $K = \{k^1, ..., k^{|K|}\}$. Each cable type $k^i \in K$ has a cost c_i and a capacity μ_i. The installed cables therefore restrict the maximum flow between two sites. The cost of installing a cable type k^i between sites u^ν and u^ι is then $dist(u^\nu, u^\iota) \cdot c_i$.

Power can be merged at turbines and intermediate nodes. This means that at a node $u^\nu \in T \cup S$ the input over different cables can be output over less cables as long as the capacity constraints are met. This way flow from different turbines can be combined in one cable. Furthermore, the node weight must be output at its node over one cable only and therefore cannot be split and assigned to different output cables. Once two flows are merged in one cable they can never be split again. We call this the unsplittable or merge-only flow requirement. Hence, the flow from a turbine t^ν along a path P_ν to a sink r passes through one cable type only along any connection $(u^\nu, u^\iota) \in P_\nu$. The flow and the capacity in this conjunction of cables along path P_ν is therefore always non-decreasing.

Given these input description and installation requirements we can now define the WFCIP. Before we do so, however, note the following tow definitions first. Let $a^i_{\nu,\iota}$ be the integer variable indicating how many copies of cable type k^i are installed between nodes u^ν and u^ι. Furthermore, let $\mathbf{1}_A(s^\nu) : S \to \{0,1\}$ indicate whether there is a cable entering or leaving an intermediate node s^ν. The definition of the WFCIP is then given by:

Problem 1. *We are given a set of cables $K = \{k^1, ..., k^{|K|}\}$ including their costs $c : K \to \mathbb{Q}_{\geq 0}$ and capacities $\mu : K \to \mathbb{N}$. The input furthermore contains a set of points $V = \{u^1, ..., u^{|V|}\}$ in the Euclidean plane including turbines $T = \{t^1, ..., t^{|T|}\}$ and substations $R = \{r^1, ..., r^{|R|}\}$. Additionally there are turbines' weight $b : T \to \mathbb{N}$ and the substations' capacities $r_\mu : R \to \mathbb{N}$. The objective of the Wind Farm Cable Installation Problem (WFCIP) then is to find the cable installation $\{a^i_{\nu,\iota}\}_{1 \leq i \leq |K|, u^\nu, u^\iota \in V}$ of minimum cost $\sum_{u^\nu, u^\iota \in V} \sum_{i=1}^{|K|} dist(u^\nu, u^\iota) \cdot c(k^i) \cdot a^i_e$ which connects every turbine with exactly one substation over one distinct path. This cable installation thereby needs to ensure for every turbine that the installed cables along the path P_ν used by turbine t^ν to send its weight b_ν to the substation support the corresponding flow such that the flow is not split. Furthermore, the installation must connect only that amount of turbines to a substation which is supported by the substation's capacity μ_r.*

From this definition we see that the WFCIP is a multi-source multi-sink problem which consists of a flow problem and an allocation problem. In this thesis, though, we deal with a subproblem which neglects the allocation of turbines to substations and only regards one substation. The problem is described in the following.

2.2. Substation Cable Installation Problem

In this thesis we deal with a subproblem of the WFCIP in which we only regard the connection of a set of turbines T to one substation r. Accordingly, we call this subproblem the Substation Cable Installation Problem (SCIP).

Again we are given a set of points $V = \{u^1, ..., u^{|V|}\}$ in the Euclidean plane which can be connected via a given set of cable types $K = \{k^1, ..., k^{|K|}\}$. This time, though, the set of nodes only comprises the set of turbines T and one substation, i.e. $V = T \cup \{r\}$.

Furthermore, we assume that the number of turbines (and therefore their total power generation) is in compliance with the capacity restriction of the substation. Apart from that all requirements and assumptions made for the WFCIP hold as well for the SCIP.

By regarding only one substation we transform the WFCIP into a *single-sink* problem. The allocation is already fixed to this substation due to the problem input.

The aim of the SCIP is to find a cost-minimal cable installation which connects every turbine with the sink such that all the power generated at the turbine is send to the sink over a path with unsplittable flow.

The subproblem can then be defined as follows:

Problem 2. *We are given a substation r and a set of turbines $T = \{t^1, ..., t^{|T|}\}$ as input points V. The input furthermore contains the turbine weights $b : T \to \mathbb{N}$ and a set of cables $K = \{k^1, ..., k^{|K|}\}$ including their costs $c : K \to \mathbb{Q}_{\geq 0}$ and capacities $\mu : K \to \mathbb{N}$. The objective of the Substation Cable Installation Problem (SCIP) is then to find the cable installation $\{a^i_{\nu,\iota}\}_{1 \leq i \leq |K|, u^\nu, u^\iota \in V}$ of minimum cost $\sum_{u^\nu, u^\iota \in V} \sum_{i=1}^{|K|} dist(u^\nu, u^\iota) \cdot c(k^i) \cdot a^i_e$ which connects all turbines $t^\nu \in T$ with the substation r over a distinct path such that the installed cables ensure an unsplittable flow of each turbine weight to the substation.*

For the ease of notation let us define $c_i = c(k^i)$, $\mu_i = \mu(k^i)$ and $b_\nu = b(u^\nu)$. Furthermore, please note again that we assume that every turbine weight is the same. We call this property *uniform turbine weight* which is in contrast to the non-uniform weight with arbitrary positive values for b_ν. Since the turbine weight is uniform we set $b_\nu = 1$ and scale the cable capacities accordingly if needed.

2.2.1. Previous work

In the literature we can already find some articles dealing with the cable installation in a wind farm for one substation.

Fagerfjäll [10] for example formulates a mixed-integer linear program (MILP) for maximising the power production as well as a MILP for minimizing the infrastructure costs. For the latter model cost occurring by foundations of a turbine, roads, cables and the location of the transformer station (substation) are considered [10, p. 10-13]. Cables are along roads [10, p. 11] and therefore the cable installation depends on the cable costs and on the road costs. The two MILPs can be combined but as Fagerfjäll [10] states it is common to solve the optimal infrastructure problem once the location of the wind turbines are fixed [10, p. 11].

Dutta [9] optimizes the collector system design with regard to the trenching length. Therefore, she uses in a first algorithm a minimum spanning tree algorithm which is improved in a second version by allowing Steiner points. Since in a practical setting it might be necessary that the number of turbines is limited on a feeder, Dutta [9] uses k-means clustering for restricting the maximum number of turbines [9, p. 86]. The MST is then calculated for each cluster. For the resulting tree of the algorithms the cable installation can be calculated by selecting a terminal node and choosing the best cable for transporting the power of the terminal node along its incident edge. The terminal node and its incident edge is then deleted from the tree and the power transported to the adjacent node is added to this one. Thereby, power losses on the cable are considered.

Instead of optimizing the trenching length, the works of Hertz et al. [20] and Berzan et al. [2] deal with the optimal cable installation such that the installation costs are minimized. Thereby, Hertz et al. [20] uses a MILP formulation and cutting planes methods to calculate the optimal design for the single sink case. The model's input consists of a graph $G = (V, E)$ with E representing the set of transmission lines available for connecting the turbines to the substation and V being a set of nodes representing the turbines, the substation and transmission line endpoints. Their approach assumes that two cable types are available, one for underground and one for above-ground connections of nodes in V. At each edge a certain number of parallel cables of one type can be used. For the cable cost they regard a cost structure in which an additional cable copy along one edge does not cost more than the installation of the previous copy of that cable. The costs are non-uniform and the flow is modeled as unsplittable. Hertz et al. [20] state that their model can be modified such that multiple cables for each edge of the graph is regarded. Though, their most efficient cutting planes assume a consecutive use of the same type of cable which is mostly the case only in their unmodified model.

Berzan et al. [2] also define the SCIP. As for the WFCIP they do provide an algorithm for the single-cable type version of the problem but not for the multi-cable one. For their model they consider costs which take into account the type of cable and the terrain used by the installation. Along an edge one cable type can be installed. Due to cable capacity and the tree constraint for their solution the flow is unsplittable. In the single cable setting Berzan et al. [2] solve the SCIP by applying an algorithm for the Capacitated Minimum Spanning Tree (CMST) Problem.

2.3. Literature review

Since the approaches to the SCIP found are only a few we have looked for related problems to identify algorithms to apply on the SCIP as well.

A well researched related problem is the Single-Sink Buy-at-Bulk (SSBB) problem which is the single sink variant of the Buy-at-Bulk (BB) problem. The SSBB was introduced by Salman et al. [31] and has gotten further consideration over the following years leading to several approximation algorithms for the BB and its variants. Following Grandoni et al. [16, p. 190] we define the SSBB as follows:[1]

Problem 3. *As an input for the problem we are given an undirected Graph $G = (V, E)$, with edge weights $w : E \to \mathbb{Q}_{\geq 0}$, a subset of nodes $T \subset V$ called sources and their integer-valued demand $b : T \to \mathbb{N}$, a sink $r \in V$ and a set of cables K, with capacities $\mu_1 \leq ... \leq \mu_{|K|}$ and costs $c_1 \leq ... \leq c_{|K|}$, where $\frac{c_i}{\mu_i}$ is decreasing in i (economies of scale). The objective of the Single-Sink Buy-at-Bulk (SSBB) is to find a cable installation $\{a^i_{\nu,\iota}\}_{1 \leq i \leq |K|, e=\{u^\nu, u^\iota\} \in E}$ of minimum cost $\sum_{e \in E} \sum_{i=1}^{|K|} w(e) \cdot c_i \cdot a^i_e$ such that from every source $t^\nu \in T$ b_ν units of flow can be routed to r without surpassing the capacity installed on an edge $e \in E$.*

From this definition we see that the SSBB is quite similar to the SCIP. The only differences are that the SSBB demands economy of scale for the cable type input and the problem also allows Steiner points which we rule out in our definition of the SCIP by setting $V = T \cup \{r\}$. Furthermore, the SSBB is defined on general graphs and not only on points in the Euclidean plane. This special case is dealt with in the BB literature nevertheless by Salman et al. [31] and Czumaj et al. [8].

In addition, the SSBB has been regarded in different variations. Apart from the multi-sink setting, the literature also deals with the cases where the cable costs are uniform or non-uniform, i.e. where the costs along an edge are the same for every edge or can be different. A further distinction on the problem is whether it takes into account splittable or unsplittable flow. In the context of BB the definition of unsplittability differs from ours, though. Here the solution is not split if it is a tree. Hence, a splitting along cables is not ruled out.

However, one can easily see that for every problem instance of the SCIP there is at least one optimal solution which is also a tree solution. This is due to the Euclidean plane setting. If two outgoing paths from a node u^ν to the sink exist in a feasible solution of the SCIP then the weight of u^ν can only flow to the sink over one of these paths due to the unsplittability constraint. The other path, however, must have a predecessor node u^ι of u^ν which we can directly connect to the successor u^ω of u^ν such that the costs for this path are at least the same due to the triangle inequality in the Euclidean plane.[2]

The SSBB has been also studied under the *Single-Sink Network Loading Problem (SSNLP)* [23, p. 91]. The Network Loading Problem (NLP) itself was first termed by Magnanti et al. [25]. In this problem a capacitated network is modeled for which no variable flow cost exist and the capacity for carrying the flow is determined by the amount of cables installed [25, p. 143]. As in the BB problems the aim is to minimize the overall cable costs such that the source-sink demands are met [25, p. 143].[3] As for BB, several variants exist

[1]Grandoni et al. [16] use this definition for what they call the Cable Single-Sink Buy-at-Bulk (CAB-SSBB) problem. The CABSSBB, however, is equivalent to the original definition of Salman et al. [31] for the SSBB.

[2]In fact the costs are always reduced as long as all three nodes u^ι, u^ν and u^ω are not on one line.

[3]Please note that we adapted the terminology used in the NLP literature to the one in the BB context. In the NLP literature facilities are installed along edges instead of cables. If the flow splits it is bifurcated.

in the literature for the NLP differing in the handling of splittability, number of sinks and number of cables . However, we will focus on approaches for solving the single-sink variant with multiple cables. The single-sink variant of the NLP whose node set contains only sources and one sink is also referred to as Local Access Network Design Problem (LANDP) [23, p. 89].

For the single cable variant of the SCIP Berzan et al. [2] use a CMST approach to solve it. Gamvros et al. [11] [12] introduced the Multi-level Capacitated Minimum Spanning Tree (MLCMST). In contrast to the CMST in which we have only one facility with a fixed capacity the CMST has a set of facilities to choose of for finding the minimum cost tree to transport the traffic from the sources to a sink [11, p. 100]. In accordance to Gamvros et al. [12, p. 348], we define the MLCMST as follows:

Problem 4. *We are given a graph $G = (V, A)$ with a set of nodes $V = \{u^1, ..., u^{|V|}\}$ and a set of edges E. One node $r \in V$ is defined as the sink and the others as source nodes $T = \{t^1, ..., t^{|T|}\}$. Let $b_\nu : E \to \mathbb{N}$ be the node weight of t^ν which has to be transported to the sink. In addition, we are given a set of cables $K = \{k^1, ..., k^{|K|}\}$, with capacities $\mu_1 < ... < \mu_{|K|}$ and costs $c_{i,\nu\iota}$ for every edge edge $(u^\nu, u^\iota) \in E$ and cable type k^i, which can be installed between u^ν and u^ι. The objective of the Multi-level Capacitated Minimum Spanning Tree (MLCMST) is then to find the cost minimal tree on G which supports the traffic from the sources to sink.*

Gamvros et al. [12, p. 349] restrict the number of cables which can be installed along one edge to 1. However, using dynamic programming we can calculate a mapping which assigns the cost minimal cable type combination to capacity values. This mapping, hence, allows the installation of multiple cables and cable types by only specifying the capacity to be installed. Furthermore, they assume that the cost function obeys economies of scale as well. According to Gamvros et al. [12, p. 349], the MLCMST is the version of the LANDP in which the solution is restricted to be a tree.

Among the literature for the problems SSBB, SSNLP and MLCMST we have searched for further approaches which can be applied to the SCIP. In the following we present the found literature and state our choice of algorithms for the later implementation at the end of this chapter.

2.3.1. Findings

Buy-at-Bulk literature

For solving the SSBB Problem several approximation algorithms have been presented in the computer science literature. Salman et al. [31] [32] introduce this problem and also present an $O(\min\{\log \frac{B}{\mu_1}, log\frac{l_{\max}}{l_{\min}}\})$ approximation algorithm for the SSBB with points in the Euclidean plane, where B is the total demand, μ_1 the minimum capacity of all cables, $l_{\max}$ the biggest and $l_{\min}$ the shortest distance between two points. For their algorithm they use a uniform cost function. Their algorithm first divides the plane successively into squares. The topmost square is centered at the sink covering all other nodes. Each square is split into 4 subsquares again leading to a center in the middle of the square where all subsquares intersect. This is done recursively until either the lowest level has side length of at most $\frac{l_{\max}}{(D/\mu_1)}$ units or each square contains one source node at most. In the next phase of the algorithm the flow is routed from the sources through the centers to the sink and the cables installed accordingly. However, if a square containing a source has the sink at one of his corners the source is not connected to next level's center but to the sink directly.

Meyerson et al. [28] present a randomized approach which approximates the SSBB problem within an expected factor of $O(\log |T|)$ on a general graph. They consider non-uniform cost. The flow is modeled indivisible leading to a tree solution. They solve the SSBB by transforming the problem into one of the Cost-Distance Problem and running their algorithm for this problem. Thereby, they replace every edge e by $|K|$ parallel edges e_i and assign each of these parallel edges with fixed cost $w_e \cdot c_i$ and incremental costs $w_e \cdot \frac{c_i}{\mu_i}$. Their Cost-Distance algorithm iteratively finds a matching on a subset of nodes V_i. This subset is reduced in every iteration since the matched nodes are randomly aggregated such that both are represented by only one. The edges being part of a matching during the execution of the algorithm form the later edges of the output connected graph minimizing the cost. This algorithm is derandomized by Chekuri et al. [6].

In their work Garg et al. [13] provide an approximation of the SSBB with a ratio of $O(|K|)$. For their approach they take an undirected graph as input. The flow is unsplittable. The cable cost are uniform. For solving the SSBB Garg et al. [13] transform the problem instance into one of a related problem, namely the Deep-Discount problem. The Deep-Discount Problem approximates the SSBB by a factor of 2 [13, p. 173]. In order to solve it they first prune the set of available cables. Then they solve a relaxation of an integer linear program (ILP) formulation of the Deep-Discount Problem optimally. Using this solution they use a rounding algorithm to get an integral solution. This algorithm incrementally builds a tree on a subset of nodes by using Steiner tree and Light Approximate Shortest-Path Tree[4] algorithms installing lower capacities each round.

Talwar [35] present a modified version of the work of Garg et al. [13] and prove a constant integrality gap $O(1) = 216$ for their own version. Their modifications contain a different cable pruning. Furthermore, Talwar [35] leave out one constraint of the ILP used by Garg et al. [13]. In the rounding algorithm they change the subset selection for the tree building at each step.

The first constant approximation algorithm for the SSBB has been provided by Guha et al. [17] with a ratio of $O(1) = 292$. Their algorithm is randomized coping with the case of a uniform, cable cost function and splittable flow. It works on a pruned cable set K' and iteratively installs cable types $k^i \in K'$. Let V_i be the set of nodes with positive demand. In each iteration cable type k^i is installed. The corresponding cable installation is determined by calculating a Steiner forest on V_i and routing the flow to one random node in each tree. On the node set V several shortest path trees are calculated and rooted at a subset of V. The aggregated flows at the Steiner Forest nodes are then sent to the root of their shortest path tree and then rerouted to another randomly chosen node in this tree. In each iteration cable type i supports the flow shifts. The set V_i decreases with every iteration and the flow is sent to the root in the end.

Gupta et al. [18] present an algorithm which is based on the one of Guha et al. [17] but with better approximation ratio, namely $O(1) = 72.8$. Their approach keeps the idea of iteratively installing cable types with higher capacity and aggregating flow at a subset of nodes in each iteration using Steiner tree and shortest path techniques. However, their algorithm always makes sure that at the beginning of each iteration the aggregated flow at a node is 0 or μ_i and in each iteration cable type k^i and k^{i+1} is installed. Furthermore they assume that the cables' capacities μ_i and costs c_i as well as the total source weight B are to a power of 2. The core of their approach is an aggregation algorithm which allows the shift of flows along a tree such that each node's aggregated weight is either 0 or a given value U and the capacity to support the shift does not succeed U. The main idea is then to

[4]A Light Approximate Shortest-Path tree ensures that the total tree length and the length from any node to the sink are within given constants (see definition in[13, p. 179]).

randomly select a subset of the nodes with positive demand μ_i in each iteration and collect the demand from the nodes which are not part of this subset at the nodes of the subset. Building a Steiner tree on the subset, using the aggregation algorithm and a subsequent random redistribution the flow is aggregated at $\frac{B}{\mu_{i+1}}$ nodes such that the new node weight is μ_{i+1} and the cables installed for this aggregation are of type k^{i+1}. The number of nodes in the selected subset decreases with each iteration and in the end all node weight is sent to the sink.

In subsequent work Jothi and Raghavachari [21], Grandoni and Italiano [15] and Grandoni et al. [16] improved the ratios to $O(1) = 65.49$, $O(1) = 24.29$ and $O(1) = 20.41$ by slight changes to the assumptions on the cables and total demand as well as to the random selections during the algorithms.

The algorithm by Gupta et al. [18] and its improvements are designed for the splittable flow setting. Jothi and Raghavachari [21] provide a version for the work of Gupta et al. [18] which transfers the algorithm to the unsplittable flow setting with approximation ratio $O(1) = 145.6$ while Grandoni et al. [16] point out a more general result which allows the transformation of a splittable solution to an unsplittable one worsening the ratio by a factor of 2.

Furthermore, Zuylen [38] present the derandomized versions of the works of Gupta et al. [18] and Grandoni and Italiano [15]. The ratios are a little bit worse, namely $O(1) = 80$ and $O(1) = 27.72$.

Czumaj et al. [8] deal with the SSBB in the Euclidean plane setting. They consider the flow as splittable and deal with uniform cable costs. The algorithm first reduces the problem instance. Therefore, it connects all source with higher weight than the maximum cable capacity $\mu_{|K|}$ directly to the sink over $\lfloor \frac{b_\nu}{\mu_{|K|}} \rfloor$ copies of cable type $k|K|$ using the shortest path. Furthermore, a grid over the plane is created and for every square containing nodes with a total amount of at least $\mu_{|K|}$ a Steiner tree is generated which is also directly connected to the sink. Hence, after this reduction every node and every square has a weight less than $\mu_{|K|}$ left. On this reduced instance a quadtree is calculated which contains one node in each leaf forming a division of the plane into squares. This allows a bottom-up dynamic programming procedure solving the problem of finding a multigraph connecting all sources in a square optimally along defined portals at the squares' borders.

Single-Sink Network Loading Problem

Berger et al. [1] present a tabu search approach for the single-sink multiple-cable unsplittable NLP. In their tabu search they use a long term adaptive memory approach for finding a start point after the restart of the search and the k-shortest path algorithm for creating the neighbourhood structure. They tested their algorithm on some randomly generated graphs with 50, 100 and 200 vertices (50% of them being demand vertices) and 9 cable types. In experiments they compared their approach with the results of a 1-opt and 2-opt neighbourhood search heuristic and showed that they improve those heuristics' results on average by a small percentage.

Gendron et al. [14] improve the work of Berger et al. [1] by providing three other diversification strategies for the determination of the start point. In experiments they show that two of the three diversification strategies perform better than the one in Berger et al. [1].

Salman et al. [33] solve the LANDP exactly. They consider three MILP formulations and a branch-and-bound algorithm. The basic idea of that branch-and-bound algorithm is to utilize the lower envelope of the cost function, which is derived from the cables, to calculate the bounds at a branch-and-bound tree node.

Ljubić et al. [23] present a branch-and-cut algorithm with Bender cuts for the LANDP and compare them with three MILP formulations. For the Bender cuts they introduce several separation approaches. In experiments they showed that their approaches mostly outperform the approach of Salman et al. [33].

Multi-level Capacitated Minimum Spanning Tree

Gamvros et al. [11] introduce the MLCMST problem and present a genetic algorithm and a MILP formulation against which they compare the experimental results of their genetic algorithm. In a consecutive work Gamvros et al. [12] revise their genetic algorithm and introduce a savings based heuristic, two local search heuristics and three MILP. Starting from a star solution the savings based heuristic iteratively searches for cable upgrades which lead to a cost reduction after rerouting additional flows over the potentially new cable. The two local search heuristics use the savings based heuristic in their initial start point and their evaluation of the improvement. The genetic algorithm makes use of the previous heuristic types as well. They are used in the initialization, selection and mutation step. In their experiments they compare the relaxation of the MILP formulations with each other on small instances. The heuristics are tested on small (up to 30 source nodes) and bigger instances (up to 150 source nodes). Their results are compared to the solutions resp. lower bounds found by one of MILP and its relaxation. The experiments showed that the genetic algorithm produces the best approximations to the benchmark instances but also need the most amount of time. The quality of the savings heuristic is the worst but therefore the computation time is significantly lower than any other of the tested approaches.

Martins et al. [27] present an ILP formulation for the MLCMST which they use for their main contribution, a GRASP approach. This "multi-start meta heuristic" [27, p. 138] creates subsets of $V \setminus \{r\}$ and solves the MLCMST exactly on the subset including the sink with the ILP. In a next step the feasible solution found is improved in a local search phase which also includes the usage of the ILP on smaller MLCMST instances. In experiments they test their approach with the instances given by Gamvros et al. [12] and compare the found upper bounds. Most of the time the GRASP approach by Martins et al. [27] finds better upper bounds than the approaches by Gamvros et al. [12].

Uchoa et al. [37] add cutting planes to the ILP formulation which Martins et al. [27] use. They replace the ILP solving steps with the corresponding Branch-and-Cut algorithm in the according parts of the GRASP approach of Martins et al. [27]. Furthermore, they extend the local search phase with a routine which prevents the call of the Branch-and-Cut algorithm on already regarded subproblems. In experiments on the same problem instances they can further improve the upper bounds especially for the hardest instances.

The algorithms of Gamvros et al. [12] assume unit demand, i.e. a node weight of one. Pappas et al. [29] extend the savings heuristic of Gamvros et al. [12] to non-unit demand and derive two more heuristics from it which regard more possibilities of node upgrades. In experiments they also included the heuristics in a Branch-and-Cut algorithm. The new heuristic generally perform better than the adapted one from Gamvros et al. [12] regarding the solution value but worsen the computation time. The inclusion in the Branch-and-Cut algorithm further improves the solution values.

2.3.2. Results

In the found literature there are different ways to tackle the SCIP and related problems. For our experimental study we have chosen five different approaches which we adapt and implement for the SCIP.

For the calculation of an exact solution we use the MILP presented by Hertz et al. [20]. Since their formulation is designed for the two cable case we have to transfer it to the multi-cable case. As Hertz et al. [20, p. 96] state themselves, the MILP can be easily adapted to the later case.

The formulation of Hertz et al. [20] is quite complex, though. Hence, we also have a closer look at the ILP formulation used in Martins et al. [27] and Uchoa et al. [37].

In addition, we also want to regard some heuristics for the SCIP. Thereby, we have decided to neglect meta-heuristics. Therefore, our choice restricts to approaches from the SCIP, SSBB and the MLCMST literature since in the SSNLP we have found only meta-heuristic and exact approaches.

From the SCIP literature we apply the idea to use a Minimum Spanning Tree (MST) approach to solve the SCIP as is also done by Dutta [9].

Since, Grandoni et al. [16] provides the best approximation ratio and the algorithmic idea has been regarded by several authors in the SSBB literature we have decided on the implementation of this algorithmic approach. Hence, we do not implement an approach of the SSBB literature which is already set in the Euclidean plane. One of the reasons for this is that it is essential for these approaches to install Steiner points. Therefore, they cannot be used for the SCIP. For Grandoni et al. [16], however, it is easy to abolish the Steiner point creation.

Among the heuristics for the MLCMST we have decided to implement all heuristics presented by Pappas et al. [29].

The algorithms and their adaption are discussed in the following chapter.

3. Implementation

As discussed in Subsection 2.3.2 we identified five approaches which we adapt to the SCIP and implement. In addition, we also present one new ILP formulation and one heuristic which is inspired by the ones of Pappas et al. [29].

In the following we present the adaption of the MILP formulation of Hertz et al. [20] in Section 3.1. In Section 3.2 our ILP formulation of the SCIP is explained. The ILP formulation used by Uchoa et al. [37] is described in Section 3.3. The heuristics are illustrated in the following. We show the MST approach in Section 3.4, the approximation algorithm of Grandoni et al. [16] in Section 3.5 and the heuristics of Pappas et al. [29] in Section 3.6. Finally, we introduce our heuristic in Section 3.7.

3.1. An adapted exact mixed-integer linear program

In their work Hertz et al. [20] deal with a SCIP version which is restricted to two cable types and a uniform turbine weight of 1. They approach the problem by modeling a MILP for finding the optimal solution. From this MILP we derive a MILP formulation which is also applicable for more than two cable types. For a better differentiation we call the adapted MILP the Two-Path MILP throughout the rest of this thesis.

The MILP models the problem as a flow problem from one source 0 to a sink r. Given an input graph $G = (V, A)$ we therefore have to include the fictitious source 0. This node is then connected to all turbines $t^\nu \in T$, i.e. to the original sources. Following these changes we define $V' = V \cup \{0\}$, $A' = A \cup \{(0, t^\nu) | \forall t^\nu \in T\}$ and $G' = (V', A')$. Furthermore, Hertz et al. [20] distinguish two subsets of A'. The set A'_1 contains all arcs (u^ν, u^ι) in A' for which there are no opposite arcs (u^ι, u^ν) in A'. All arcs which have an opposite arc in A' are included in E as edges $\{u^\nu, u^\iota\}$. In addition, let P_2 be the set of node combinations $(u^\nu, u^\iota, u^\omega)$ which define a path of length 2 in G'.

In our setting graph G contains all points in V and the arc set A consists of all possible connections between two points and the connection of every turbine to the sink. Consequently, A'_1 contains arcs $(0, t^\nu)$ and (t^ν, r) only.

Along every arc all cable types k^i can be installed. Though, the number of copies for each cable type is restricted by $|L_i|$. We set this limit to $|L_i| = \lceil \frac{B}{\mu_i} \rceil$. This way we can support the highest possible flow in the graph with every cable type.

Hertz et al. [20] assume that the cost of a link installation decreases with every additional link installed. We drop this assumption and use the same cost for every copy of a cable type.

For their model Hertz et al. [20] define four types of variables. Variable $x^l_{i,\nu\iota}$ is a binary decision variable which is 1 if along arc (u^ν, u^ι) cable type k^i is installed on link l. The flow along arc (u^ν, u^ι) using link l of cable type k^i is represented by $x^l_{i,\nu\iota} \geq 0$. The other two variables are defined for all node combinations in P_2 and are utilized later on to assure unsplitability. Like $x^l_{i,\nu\iota}$ variable $y^{ll'}_{ij,\nu\iota\omega}$ is a binary variable. It indicates whether the flow over (u^ν, u^ι) using link l of cable type k^i leaves u^ι to u^ω over l' of cable type k^j. Analogously, $z^{ll'}_{ij,\nu\iota\omega} \geq 0$ defines the flow entering u^ι from u^ν over link l of cable type k^i and then leaving over arc (u^ι, u^ω) over link l' of cable type k^j.

In the following we will discuss the MILP in more detail.

Total cost minimization The aim of the MILP is to minimize the total cost. Therefore, we regard every cost which might occur in the graph. Costs $dist(u^\nu, u^\iota) \cdot c_i$ occur if we install a cable type k^i on a link l along an arc ν, ι, i.e. if $t^l_{i,\nu\iota} = 1$. Hence, our minimization objective is defined as follows:

$$\min \sum_{(u^\nu,u^\iota)\in A'} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} dist(u^\nu, u^\iota) \cdot c_i \cdot t^l_{i,\nu\iota}$$

Flow conservation constraint Since we want to send all flow from the source 0 to the sink r we have to make sure that every node in between outputs all its incoming flow. In set $P(u^\nu)$ we store all incoming and in set $S(u^\nu)$ all outgoing arcs. In order to calculate a node's input flow we have to sum over $x^l_{i,\iota\nu}$ for all incoming edges (u^ι, u^ν) and all their individual links $l \in L_i \; \forall k^i \in K$. Similarly, the outgoing flow is computed over $x^l_{i,\nu\iota}$ for all outgoing arcs and their links. The difference of the incoming flow to the outgoing flow must be 0 for all nodes except 0 and r. For the source 0 the difference must equal $-B$ since all energy from the turbines must be output. Accordingly, the difference should be B for the sink because all the energy is collected there. Putting these thoughts into a mathematical formulation we get the following constraints:

$$\sum_{(u^\iota,u^\nu)\in P(u^\nu)} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x^l_{i,\iota\nu} - \sum_{(u^\nu,u^\iota)\in S(u^\nu)} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x^l_{i,\nu\iota} = \begin{cases} |T| & u^\nu = r \\ -|T| & u^\nu = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall u^\nu \in V'$$

Power output constraint Since all flow origins at 0 every turbine has to receive its rating as an input flow. Therefore, the following constraint sums up all flow over an arc $(0, t^\nu)$ and restricts this sum to be one, i.e. the assumed power rating of a turbine.

$$\sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x^l_{i,0\nu} = 1 \qquad \forall t^\nu \in T$$

Link capacity constraint With the link capacity constraints as formulated as below we achieve two things: First, we restrict each flow along one link l to the capacity μ_{k^i} of the corresponding cable type k^i. Second, we do not allow any flow over the link if the link is not installed, i.e. if $t^l_{i,\nu\iota} = 0$.

$$x^l_{i,\nu\iota} \leq \mu_i \cdot t^l_{i,\nu\iota} \qquad \forall (u^\nu, u^\iota) \in A', \forall l \in L_i, \forall k^i \in K$$

Proper usage of variables $t^l_{i,\nu\iota}$ As we know from the definition of E there are arcs in G' which have an opposite arc. Since the limitation of links holds for both directions we need to make sure that link l is installed and therefore used in only one direction. The following constraints therefore force $t^l_{i,\nu\iota}$ and $t^l_{i,\iota\nu}$ to be 0 if the other one is 1 for all $\{u^\nu, u^\iota\}$.

$$t^l_{i,\nu\iota} + t^l_{i,\iota\nu} \leq 1 \qquad \forall \{u^\nu, u^\iota\} \in E, \forall l \in L_i, \forall k^i \in K$$

Furthermore, we require that a link $l+1$ can only be utilized if l of the same cable type k^i is installed already. For this we distinguish two cases. The first one regards arcs $\nu, \iota \in A'_1$. Since we know that there are no opposite arcs in A'_1 variable $t^{l+1}_{i,\iota\nu}$ can only be one if $t^l_{i,\nu\iota}$ is one as well. This is formulated in the following constraints:

$$t^{l+1}_{i,\nu\iota} \leq t^l_{i,\nu\iota} \qquad \forall (u^\nu, u^\iota) \in A'_1, \forall l \in L_i, l < |L_i|, \forall k^i \in K$$

The second case deals with edges $\{u^\nu, u^\iota\} \in E$. Since a link can be used in both directions we now have to check whether $t^l_{i,\nu\iota}$ or $t^l_{i,\iota\nu}$ equal 1. Due to the first constraint in this group only one of these variables can be 1 at a time. If one of the variables either $t^{l+1}_{i,\iota\nu}$ or $t^{l+1}_{i,\iota\nu}$ can be one as well. This leads to the following formulation:

$$t^{l+1}_{i,\nu\iota} + t^{l+1}_{i,\iota\nu} \leq t^l_{i,\nu\iota} + t^l_{i,\iota\nu} \qquad \forall\{u^\nu, u^\iota\} \in E, \forall l \in L_i, l < |L_i|, \forall k^i \in K$$

Unsplittable flow With the following constraints we model the unsplittable flow property of the SCIP. For this purpose we use the path variables $y^{ll'}_{ij,\nu\iota\omega}$ and $z^{ll'}_{ij,\nu\iota\omega}$. Since $z^{ll'}_{ij,\nu\iota\omega}$ represents the flow going over link l of cable type k^i on arc (u^ν, u^ι) and then over l' of k^j on (u^ι, u^ω) this flow must equal the flow represented by $x^l_{i,\nu\iota}$ and $x^{l'}_{j,\iota\omega}$. The following two constraints relate the two variable types accordingly.

$$\sum_{(u^\nu,u^\iota)\in P(u^\iota)} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} z^{ll'}_{ij,\nu\iota\omega} = x^{l'}_{j,\iota\omega} \qquad \forall(u^\iota, u^\omega) \in A', u^\iota \neq 0, \forall l' \in L_{l'}, \forall k^j \in K$$

$$\sum_{(u^\iota,u^\omega)\in S(u^\iota)} \sum_{j=1}^{|K|} \sum_{l=1}^{|L_{k^j}|} z^{ll'}_{ij,\nu\iota\omega} = x^l_{i,\nu\iota} \qquad \forall(u^\nu, u^\iota) \in A', u^\iota \neq r, \forall l \in L_i, \forall k^i \in K$$

Remark that the first constraints are not applied to $u^\iota = 0$ because there is no arc from any node to 0 and hence, $P(0) = \emptyset$. Analogously, the second constraints only hold for nodes $u^\iota \neq r$.

The flow along a path P_2 is obligated to meet the capacity constraints as well. For now the introduced capacity constraints only ensure that each flow on one link meets the corresponding cable's capacity. For $z^{ll'}_{ij,\nu\iota\omega}$ we have to make sure that the flow does not exceed the capacity μ_{k^i} of cable type k^i nor μ_{k^j} of cable type k^j. Hence, we restrict the flow $z^{ll'}_{ij,\nu\iota\omega}$ to the minimum of μ_{k^i} and μ_{k^j} if $y^{ll'}_{ij,\nu\iota\omega} = 1$. If the flow from u^ν to u^ι over l of cable type k^i is not output at u^ν to u^ω over l' of k^j, i.e. $y^{ll'}_{ij,\nu\iota\omega} = 0$, the flow must equal 0 as well. The path capacity constraint is then given by

$$z^{ll'}_{ij,\nu\iota\omega} \leq \min(c_{k^i}, c_{k^j}) \cdot y^{ll'}_{ij,\nu\iota\omega} \qquad \forall(u^\nu, u^\iota, u^\omega) \in P_2, \forall l, \in L_i, \forall l' \in L_{k^j}, \forall k^i, k^j \in K.$$

With the flow conservation constraint we already model the fact that all input flow must be output for every node except the sink.[1] Consequently, $t^l_{i,\nu\iota} = 1$ implies that there must be a $y^{ll'}_{ij,\nu\iota\omega} = 1$ in order to allow an output of the corresponding flow from u^ν to u^ι over link l of cable type k^i. Otherwise, if $t^l_{i,\nu\iota} = 0$ there cannot be any flow over the corresponding link and consequently over any path including this link. Hence, the following constraint ensures that exactly one $y^{ll'}_{ij,\nu\iota\omega}$ equals 1 if $t^l_{i,\nu\iota}$ equals 1 and every $y^{ll'}_{ij,\nu\iota\omega}$ is 0 if $t^l_{i,\nu\iota}$ is 0.

$$\sum_{(u^\iota,u^\omega)\in S(u^\iota))} \sum_{j=1}^{|K|} \sum_{l'=1}^{|L_{k^j}|} y^{ll'}_{ij,\nu\iota\omega} = t^l_{i,\nu\iota} \qquad \forall(u^\nu, u^\iota) \in A', u^\iota \neq r, \forall l \in L_i, \forall k^i \in K$$

The full MILP is presented below.

[1] In case of the ficticous source the total turbine weight must be output.

$$\min \sum_{(u^\nu,u^\iota)\in A'} \sum_{i=1}^{|K|}\sum_{l=1}^{|L_i|} dist(u^\nu,u^\iota)\cdot c_i \cdot t^l_{i,\nu\iota} \tag{3.1}$$

$$\text{s.t.} \sum_{(u^\iota,u^\nu)\in P(u^\nu)} \sum_{i=1}^{|K|}\sum_{l=1}^{|L_i|} x^l_{i,\iota\nu} - \sum_{(u^\nu,u^\iota)\in S(u^\nu)} \sum_{i=1}^{|K|}\sum_{l=1}^{|L_i|} x^l_{i,\nu\iota} = \begin{cases} |T| & u^\nu = r \\ -|T| & u^\nu = 0 \\ 0 & \text{otherwise} \end{cases} \quad \forall u^\nu \in V' \tag{3.2}$$

$$\sum_{i=1}^{|K|}\sum_{l=1}^{|L_i|} x^l_{i,0\nu} = 1 \qquad \forall t^\nu \in T \tag{3.3}$$

$$x^l_{i,\nu\iota} \le \mu_i \cdot t^l_{i,\nu\iota} \qquad \forall (u^\nu,u^\iota)\in A', \forall l \in L_i, \forall k^i \in K \tag{3.4}$$

$$t^l_{i,\nu\iota} + t^l_{i,\iota\nu} \le 1 \qquad \forall \{u^\nu,u^\iota\}\in E, \forall l \in L_i, \forall k^i \in K \tag{3.5}$$

$$t^{l+1}_{i,\nu\iota} \le t^l_{i,\nu\iota} \qquad \forall (u^\nu,u^\iota)\in A'_1, \forall l \in L_i, l < |L_i|, \forall k^i \in K \tag{3.6}$$

$$t^{l+1}_{i,\nu\iota} + t^{l+1}_{i,\iota\nu} \le t^l_{i,\nu\iota} + t^l_{i,\iota\nu} \qquad \forall \{u^\nu,u^\iota\}\in E, \forall l \in L_i, l < |L_i|, \forall k^i \in K \tag{3.7}$$

$$\sum_{(u^\iota,u^\omega)\in S(u^\iota)} \sum_{j=1}^{|K|}\sum_{l=1}^{|L_{kj}|} z^{ll'}_{ij,\nu\iota\omega} = x^l_{i,\nu\iota} \qquad \forall (u^\nu,u^\iota)\in A', u^\iota \ne r, \forall l \in L_i, \forall k^i \in K \tag{3.8}$$

$$\sum_{(u^\nu,u^\iota)\in P(u^\iota)} \sum_{i=1}^{|K|}\sum_{l=1}^{|L_i|} z^{ll'}_{ij,\nu\iota\omega} = x^{l'}_{j,\iota\omega} \qquad \forall (u^\iota,u^\omega)\in A', u^\iota \ne 0, \forall l' \in L_{l'}, \forall k^j \in K \tag{3.9}$$

$$z^{ll'}_{ij,\nu\iota\omega} \le \min(c_{k^i}, c_{k^j}) \cdot y^{ll'}_{ij,\nu\iota\omega} \qquad \forall (u^\nu,u^\iota,u^\omega)\in P_2, \forall l, \in L_i, \forall l' \in L_{k^j}, \forall k^i, k^j \in K \tag{3.10}$$

$$\sum_{(u^\iota,u^\omega)\in S(u^\iota))} \sum_{j=1}^{|K|}\sum_{l'=1}^{|L_{kj}|} y^{ll'}_{ij,\nu\iota\omega} = t^l_{i,\nu\iota} \qquad \forall (u^\nu,u^\iota)\in A', u^\iota \ne r, \forall l \in L_i, \forall k^i \in K \tag{3.11}$$

$$t^l_{i,\nu\iota} \in \{0,1\} \qquad \forall (u^\nu,u^\iota)\in A', \forall l \in L_i, \forall k^i \in K \tag{3.12}$$

$$x^l_{i,\nu\iota} \ge 0 \qquad \forall (u^\nu,u^\iota)\in A', \forall l \in L_i, \forall k^i \in K \tag{3.13}$$

$$y^{ll'}_{ij,\nu\iota\omega} \in \{0,1\} \qquad \forall (u^\nu,u^\iota,u^\omega)\in P_2, \forall l, \in L_i, \forall l' \in L_{k^j}, \forall k^i, k^j \in K \tag{3.14}$$

$$z^{ll'}_{ij,\nu\iota\omega} \ge 0 \qquad \forall (u^\nu,u^\iota,u^\omega)\in P_2, \forall l, \in L_i, \forall l' \in L_{k^j}, \forall k^i, k^j \in K \tag{3.15}$$

3.2. A new exact integer linear program formulation

Apart from the adaption of the MILP of Hertz et al. [20] we have also formulated a new ILP to which we refer as the Path ILP in the remainder of the thesis. The basic idea is to regard the turbine-sink cable path for every turbine. For each path we ensure that the flow output at the turbine reaches the sink and that paths once combined in a cable are not split anymore.

The Path ILP is designed for working directly with the input described in Section 2.1 and Section 2.2. For the further description let E be the set of edges for all possible connections between two distinct nodes u^ν, i.e. $E = \{(u^\nu, u^\iota) | u^\nu \in V, u^\iota \in V \setminus \{u^\nu\}\}$.

For the formulation of the ILP let $x_{i,l}^{\nu,\iota\omega}$ be the binary variable which indicates if the power produced at turbine t^ν is sent along edge (u^ι, u^ω) over copy l of cable type k^i. Furthermore, let $a_{i,l}^{\iota\omega}$ represent the decision whether a cable copy l of cable type k^i is installed along edge (u^ι, u^ω). Finally, the binary variables $y_\gamma^{t^\nu,t^\iota}$ and $z_\gamma^{t^\nu,t^\iota}$ model the state whether there are two flows from t^ν and t^ι entering resp. leaving u^γ via different cable copies.

In addition, we define the function $f(u^\omega, k^i, l)$ as

$$f(u^\omega, k^i, l) = i \cdot |V|^2 + l \cdot |V| + \omega$$

and the constant ϵ as

$$\epsilon = \frac{1}{(|K| + 2) \cdot |V|^2}.$$

The value of function $f(u^\omega, k^i, l)$ is unique for every $u^\omega \in V, k^i \in K, l \in L_i$ and multiplied by ϵ the value is always less than 1.

In order to model the unsplitability constraint we introduce two macros $IN_\gamma^{t^\nu,t^\iota}$ and $OUT_\gamma^{t^\nu,t^\iota}$ which we define as

$$IN_\gamma^{t^\nu,t^\iota} = \epsilon \cdot \sum_{u^\omega \in V \setminus \{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l \in L_i} f(u^\omega, k^i, l) \cdot (x_{i,l}^{\nu,\omega\gamma} - x_{i,l}^{\iota,\omega\gamma})$$

and

$$OUT_\gamma^{t^\nu,t^\iota} = \begin{cases} \epsilon \cdot \sum_{u^\omega \in V \setminus \{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l \in L_i} f(u^\omega, k^i, l) \cdot (-x_{i,l}^{\iota,\gamma\omega}) & u^\iota = t^\nu \\ \epsilon \cdot \sum_{u^\omega \in V \setminus \{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l \in L_i} f(u^\omega, k^i, l) \cdot (x_{i,l}^{\nu,\gamma\omega}) & u^\iota = t^\iota \\ \epsilon \cdot \sum_{u^\omega \in V \setminus \{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l \in L_i} f(u^\omega, k^i, l) \cdot (x_{i,l}^{\nu,\gamma\omega} - x_{i,l}^{\iota,\gamma\omega}) & \text{otherwise} \end{cases}$$

These macros are 0 only if no flow from turbine t^ν and t^ι enters resp. leaves node u^γ ($x_{i,l}^{\nu,\omega\gamma} = x_{i,l}^{\iota,\omega\gamma} = 0$) or both flows share the same cable copy ($x_{i,l}^{\nu,\omega\gamma} - x_{i,l}^{\iota,\omega\gamma} = 0$). Otherwise, if only flow from one turbine or flow from both turbines enters resp. leaves over two separate cables the macros differ from 0. To explain this more in detail let's assume that $x_{i,l}^{\nu,\omega\gamma}$ and $x_{i,l}^{\iota,\omega\gamma}$ can be 1 only for one distinct combination of (u^ι, k^i, l) at a node u^γ.[2] Recall that function $f(u^\omega, k^i, l)$ creates a unique value for each combination (u^ι, k^i, l). Hence, if only flow from one turbine enters there is exactly one $x_{i,l}^{t,\omega\gamma} = 1$ which is multiplied by the value of $f(u^\omega, k^i, l)$ leading to $IN_\gamma^{t^\nu,t^\iota} \neq 0$. Analogously, this holds for leaving flow and $OUT_\gamma^{t^\nu,t^\iota}$ as well. If both turbines send flow over u^γ but that flow enters or leaves using a different

[2]We ensure this assumptions by the combination of turbine output constraints and the flow conservation constraints.

link, cable and/or origin resp. destination we get a value $\epsilon \cdot (f(u^\omega, k^i, l) - f(u^\omega, k^j, l')) \neq 0$ due to the uniqueness of $f(u^\omega, k^i, l)$. The constant ϵ then ensures that $|IN_\gamma^{t^\nu,t^\iota}| < 1$ and $|OUT_\gamma^{t^\nu,t^\iota}| < 1$.

Total costs minimization Again the objective of the ILP is to minimize the total cable installation costs. Hence, we sum over the cable costs for arc (u^ι, u^ω) multiplied with the installation decision variable $a_{i,l}^{\iota\omega}$ for all arc, cable and link combinations.

$$\min \sum_{(u^\iota,u^\omega)\in E} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} dist(u^\iota, u^\omega) \cdot c_i \cdot a_{i,l}^{\iota\omega}$$

Turbine output constraint Every turbine should output its generated power. Hence, we force this for every turbine by summing over $x_{i,l}^{\nu,\nu\omega}$ for every node $u^\omega \in V\backslash\{t^\nu\}$ and all link and cable combinations and requiring this sum to be 1. This ensures that flow from t^ν to some node u^ω exists. The full formulation therefore looks the following:

$$\sum_{u^\omega\in V\backslash\{t^\nu\}} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\nu\omega} = 1 \qquad \forall t^\nu \in T$$

Sink input constraint All the output of the turbines should reach the sink r. The following constraint guarantees this by forcing the input minus the output at the sink to be as big as the total generated power B.

$$\sum_{t^\nu\in T} \sum_{u^\omega\in V} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\omega r} - B = \sum_{t^\nu\in T} \sum_{u^\omega\in V} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,r\omega}$$

Flow conservation constraint For every turbine t^ν there should be a consecutive path from t^ν to the sink. Therefore, each node $u^\iota \in V\backslash\{t^\nu, r\}$ which has an input from t^ν must have an output for t^ν as well. By the following constraint, which sums all incoming links from t^ν at u^ι on the left and all outgoing links on the right side of the equation mark, we enforce that the number of input and output links from one turbine t^ν is the same.

$$\sum_{u^\omega\in V\backslash\{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\omega\iota} = \sum_{u^\omega\in V\backslash\{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\iota\omega} \qquad \forall t^\nu \in T, u^\iota \in V \setminus \{t^\nu, r\}$$

In combination with the turbine output and the sink input constraint we make sure that only one link is used as an input and output for each turbine.

Unsplittability constraints In the following we describe the constraints which ensure the unsplittable flow throughout the network. Here we make use of the macros $IN_\gamma^{t^\nu,t^\iota}$ and $OUT_\gamma^{t^\nu,t^\iota}$ defined above and relate the variables $y_\gamma^{t^\nu,t^\iota}$ and $z_\gamma^{t^\nu,t^\iota}$ to them. Therefore, we also introduce the binary variables $\alpha_\gamma^{t^\nu,t^\iota}$ and $\beta_\gamma^{t^\nu,t^\iota}$ which we need to model the relation correctly.

The macros $IN_\gamma^{t^\nu,t^\iota}$ and $OUT_\gamma^{t^\nu,t^\iota}$ differ from 0 if the incoming resp. outgoing flow from turbines t^ν and t^ι use different links. Hence, $y_\gamma^{t^\nu,t^\iota}$ and $z_\gamma^{t^\nu,t^\iota}$ need to be 1 in that case. We ensure this by the following four constraints.

$$y_\gamma^{t^\nu,t^\iota} \geq IN_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

$$y_\gamma^{t^\nu,t^\iota} \geq -IN_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

$$z_\gamma^{t^\nu,t^\iota} \geq OUT_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

$$z_\gamma^{t^\nu,t^\iota} \geq -OUT_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

These constraints do not suffice yet to ensure that $y_\gamma^{t^\nu,t^\iota}$ and $z_\gamma^{t^\nu,t^\iota}$ are 1 only if two different links are used. In order to deal with the case that $IN_\gamma^{t^\nu,t^\iota} = 0$ resp. $OUT_\gamma^{t^\nu,t^\iota} = 0$ we introduce the following group of constraints. For the further explanation let's regard the incoming case and conesequently $y_\gamma^{t^\nu,t^\iota}$. The argumentation holds for $z_\gamma^{t^\nu,t^\iota}$ as well.

In order to guarantee that $y_\gamma^{t^\nu,t^\iota}$ is 0 if $IN_\gamma^{t^\nu,t^\iota} = 0$ let's regard the two terms $1+IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2}+(1-\alpha_\gamma^{t^\nu,t^\iota})$ and $1-IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2}+\alpha_\gamma^{t^\nu,t^\iota}$. One of both terms is smaller than 1 for $IN_\gamma^{t^\nu,t^\iota} = 0$ and for any assignment of $\alpha_\gamma^{t^\nu,t^\iota}$. Hence, setting $y_\gamma^{t^\nu,t^\iota} \leq 1 + IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + (1 - \alpha_\gamma^{t^\nu,t^\iota})$ and $y_\gamma^{t^\nu,t^\iota} \leq 1 - IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + \alpha_\gamma^{t^\nu,t^\iota}$ ensures $y_\gamma^{t^\nu,t^\iota} = 0$ for $IN_\gamma^{t^\nu,t^\iota} = 0$.

For the case $IN_\gamma^{t^\nu,t^\iota} \neq 0$ remark that $\frac{\epsilon}{2} < |IN_\gamma^{t^\nu,t^\iota}|$.[3] Depending on whether $IN_\gamma^{t^\nu,t^\iota} < 0$ or $IN_\gamma^{t^\nu,t^\iota} > 0$ either $1+IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2}+(1-\alpha_\gamma^{t^\nu,t^\iota})$ or $1-IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2}+\alpha_\gamma^{t^\nu,t^\iota}$ becomes smaller than 1 if $\alpha_\gamma^{t^\nu,t^\iota} = 1$ resp. $\alpha_\gamma^{t^\nu,t^\iota} = 0$. However, this contradicts with the first constraints for $y_\gamma^{t^\nu,t^\iota}$ defined above. So the ILP solver can only find a feasible solution if it adapts $\alpha_\gamma^{t^\nu,t^\iota}$ correctly. Hence, by introducing the following four constraints we make sure that $y_\gamma^{t^\nu,t^\iota}$ and $z_\gamma^{t^\nu,t^\iota}$ equal 0 if $IN_\gamma^{t^\nu,t^\iota} = 0$ resp. $OUT_\gamma^{t^\nu,t^\iota} = 0$ but in combination with the above constraints we also force them to be 1 in case $IN_\gamma^{t^\nu,t^\iota} = 1$ resp. $OUT_\gamma^{t^\nu,t^\iota} = 1$.

$$y_\gamma^{t^\nu,t^\iota} \leq 1 + IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + (1 - \alpha_\gamma^{t^\nu,t^\iota}) \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

$$y_\gamma^{t^\nu,t^\iota} \leq 1 - IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + \alpha_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

$$z_\gamma^{t^\nu,t^\iota} \leq 1 + OUT_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + (1 - \beta_\gamma^{t^\nu,t^\iota}) \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

$$z_\gamma^{t^\nu,t^\iota} \leq 1 - OUT_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + \beta_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

Finally, we relate $y_\gamma^{t^\nu,t^\iota}$ with $z_\gamma^{t^\nu,t^\iota}$ by the following constraints.

$$y_\gamma^{t^\nu,t^\iota} \geq z_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu, t^\iota \in T, t^\nu \neq t^\iota, u^\iota \in V \setminus \{r\}$$

This way the path of two turbines can only leave a node u^γ over two different links if it already entered u^γ over two different ones.

Capacity constraints Each edge has a capacity depending on the cables installed on it. The flow over this edge must meet these capacity limitations. Therefore, we check for every edge and the link combinations whether the installed capacity $\mu_i \cdot a_{i,l}^{\iota\omega}$ supports the accumulated flow of all turbine-sink-paths using this edge. The amount of flow thereby equals the turbine's weight b_ν.

$$\sum_{t^\nu \in T} b_\nu \cdot x_{i,l}^{\nu,\iota\omega} \leq \mu_k^i \cdot a_{i,l}^{\iota\omega} \qquad \forall (u^\iota, u^\omega) \in E, k^i \in K, l \in L_i$$

As a consequence of this formulation we also enforce $x_{i,l}^{\nu,\iota\omega}$ to be 0 if $a_{i,l}^{\iota\omega} = 0$.

[3] The smallest value for $|IN_\gamma^{t^\nu,t^\iota}|$ equals $\epsilon \cdot f(1,1,1) = \epsilon \cdot (|V|^2 + |V|) > \epsilon > \frac{\epsilon}{2}$.

The full ILP formulation is shown in the following:

$$\min \sum_{(u^\iota,u^\omega)\in E} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} dist(u^\iota,u^\omega)\cdot c_i \cdot a_{i,l}^{\iota\omega} \qquad [3.16]$$

$$\text{s.t.} \sum_{u^\omega\in V\setminus\{t^\nu\}} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\nu\omega} = 1 \qquad \forall t^\nu \in T \quad [3.17]$$

$$\sum_{u^\omega\in V\setminus\{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\omega\iota} = \sum_{u^\omega\in V\setminus\{u^\iota\}} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\iota\omega} \qquad \forall t^\nu \in T, u^\iota \in V\setminus\{t^\nu, r\} \quad [3.18]$$

$$\sum_{t^\nu\in T} \sum_{u^\omega\in V} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,\omega r} - B = \sum_{t^\nu\in T} \sum_{u^\omega\in V} \sum_{i=1}^{|K|} \sum_{l=1}^{|L_i|} x_{i,l}^{\nu,r\omega} \qquad [3.19]$$

$$\sum_{t^\nu\in T} b_\nu \cdot x_{i,l}^{\nu,\iota\omega} \le \mu_k^i \cdot a_{i,l}^{\iota\omega} \qquad \forall (u^\iota,u^\omega)\in E, k^i\in K, l\in L_i \quad [3.20]$$

$$y_\gamma^{t^\nu,t^\iota} \ge IN_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.21]$$

$$y_\gamma^{t^\nu,t^\iota} \ge -IN_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.22]$$

$$y_\gamma^{t^\nu,t^\iota} \le 1 + IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + (1-\alpha_\gamma^{t^\nu,t^\iota}) \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.23]$$

$$y_\gamma^{t^\nu,t^\iota} \le 1 - IN_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + \alpha_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.24]$$

$$z_\gamma^{t^\nu,t^\iota} \ge OUT_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.25]$$

$$z_\gamma^{t^\nu,t^\iota} \ge -OUT_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.26]$$

$$z_\gamma^{t^\nu,t^\iota} \le 1 + OUT_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + (1-\beta_\gamma^{t^\nu,t^\iota}) \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.27]$$

$$z_\gamma^{t^\nu,t^\iota} \le 1 - OUT_\gamma^{t^\nu,t^\iota} - \frac{\epsilon}{2} + \beta_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.28]$$

$$y_\gamma^{t^\nu,t^\iota} \ge z_\gamma^{t^\nu,t^\iota} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.29]$$

$$a_{i,l}^{\iota\omega} \in \{0,1\} \qquad \forall (u^\iota,u^\omega)\in E, k^i\in K, l\in L_i \quad [3.30]$$

$$x_{i,l}^{\nu,\iota\omega} \in \{0,1\} \qquad \forall t^\nu \in T \forall (u^\iota,u^\omega)\in E, k^i\in K, l\in L_i \quad [3.31]$$

$$y_\gamma^{t^\nu,t^\iota}, z_\gamma^{t^\nu,t^\iota} \in \{0,1\} \qquad \forall t^\nu,t^\iota\in T, t^\nu\ne t^\iota, u^\iota\in V\setminus\{r\} \quad [3.32]$$

3.3. A suboptimal integer linear program formulation

Uchoa et al. [37] develop a rather compact ILP for the MLCMST with few constraints and only one decision variable. We refer to this ILP as the Flow ILP to distinguish it from the previous ones. The formulation works on a directed graph $G = (V, A)$ with A having directed arcs in both directions for every edge $e = \{u^\nu, u^\iota\}, e \in E \setminus \{\{r, u^\nu\}\}$. For edges $\{r, u^\nu\}$ only arcs in direction (r, u^ν) are allowed. The nodes' weights b_ν are interpreted as demands. The ILP then looks for a capacitated tree which supports the flow from r to every turbine t^ν such that the demand at every turbine is met. Along one edge only one cable can be installed. The solution provides a cable installation which enables collection of the energy from every turbine at the sink. We only have to reverse the found flow.

The key to the compact formulation is the choice of the binary decision variable $x_{\nu,\iota}^f$ and the requirement that the capacity increases from 1 to the maximum cable capacity

M in unitary steps, i.e. $\mu_i = i \; \forall i \in [1, M]$. Variable $x^f_{\nu,\iota}$ then indicates whether arc ν, ι is present in the solution and if f amount of flow is sent along ν, ι. The amount of flow already indicates which capacity has to be installed along which edge due to the unitary increase requirement for the capacities.

Since Uchoa et al. [37] restrict their ILP to only allow one capacity, i.e. cable, on one edge and require an unitary increasing capacity we have to transform our cable input. Therefore, we calculate the optimal cable combination for every flow $f \in [1, B]$ as described in Subsection 3.3.1. By setting $M = B$ we make sure that we can support the flow from all turbines along one edge. After the transformation our cable set K' contains B cable combinations with capacity $\mu_i = i$ for all $i \in [1, B]$.

For the further description let c_f be the cost of the cable combination for a flow of f units, i.e. the costs of the cable combination $k^f \in K'$ with capacity μ_f. Due to the transformation the solution of the ILP is not necessarily feasible anymore since the cable combinations might lead to split flow.[4]

In the following we will explain the model in more detail.

Total cost minimization With the target function we minimize the total cost of a capacity installation summing over all possible combinations of edges $\nu, \iota \in A$ and flows $f \in [1, B]$. The costs for one edge are composed of the distance between the two incident nodes of the arc, u^ν and u^ι, multiplied with the costs per meter c_f of the capacity needed for flow f. If $x^f_{\nu,\iota}$ is 1, i.e. over ν, ι flows f amount of flow, the according costs are considered by the target function.

$$\min \sum_{\nu,\iota \in A} \sum_{f=1}^{B} dist(u^\nu, u^\iota) \cdot c_f \cdot x^f_{\nu,\iota}$$

One output constraints With the following constraint we want to make sure that each node has only one incoming edge and that this edge can only be used for one specific flow value f. In order to achieve this we sum over $x^f_{\nu,\iota}$ for all incoming arcs $\nu, \iota \in P(u^\iota)$ and all possible capacity values f and require it to be 1 for every node u^ι except the sink.

$$\sum_{\nu,\iota \in P(u^\iota)} \sum_{f=1}^{B} x^f_{\nu,\iota} = 1 \qquad \forall u^\iota \in V \setminus \{r\}$$

Flow conservation constraints In order to ensure the flow conservation the incoming flow must equal the sum of outgoing flow and demand b_{u^ι} for all nodes $u^\iota \in V \setminus \{r\}$. The incoming and outgoing flow of a node u^ι can be computed by summing all the flow f of all incoming $\nu, \iota \in P(u^\nu)$ resp. outgoing arcs $\nu, \iota \in S(u^\nu)$ for which $x^f_{\nu,\iota} = 1$ resp. $x^f_{\iota,\nu} = 1$. This leads to the following formulation:

$$\sum_{\nu,\iota \in P(u^\iota)} \sum_{f=1}^{B} f \cdot x^f_{\nu,\iota} = b_{u^\iota} + \sum_{(u^\iota,u^\nu) \in S(u^\iota)} \sum_{f=1}^{B} f \cdot x^f_{\iota,\nu} \qquad \forall u^\iota \in V \setminus \{r\}$$

The full ILP model is given in the following:

[4]As an example think of two cable types k^i and k^j with $(\mu_i = 2, c_i = 1)$ and $(\mu_j = 3, c_j = 3)$. Assume node u^ν has an input of 3 units. Then the optimal incoming cable combination is one cable of type k^j. If node u^ν has a weight of 1 then the output flow is 4. Since the optimal cable combination of a flow of 4 is using cable type k^i twice, the incoming flow is split when output.

$$\min \quad \sum_{\nu,\iota \in A} \sum_{f=1}^{B} dist(u^\nu, u^\iota) \cdot c_f \cdot x^f_{\nu,\iota} \qquad [3.33]$$

$$\text{s.t.} \quad \sum_{\nu,\iota \in P(u^\iota)} \sum_{f=1}^{B} x^f_{\nu,\iota} = 1 \qquad \forall u^\iota \in V \setminus \{r\} \quad [3.34]$$

$$\sum_{\nu,\iota \in P(u^\iota)} \sum_{f=1}^{B} f \cdot x^f_{\nu,\iota} = b_{u^\iota} + \sum_{(u^\iota,u^\nu) \in S(u^\iota)} \sum_{f=1}^{B} f \cdot x^f_{\iota,\nu} \qquad \forall u^\iota \in V \setminus \{r\} \quad [3.35]$$

$$x^f_{\nu,\iota} \in \{0,1\} \qquad \forall \nu,\iota \in E, f \in [1,B] \quad [3.36]$$

In order to speed up the calculation Uchoa et al. [37] also introduced some cutting planes. However, we could not achieve that speedup in some initial test but got a rather big slow down. Therefore, we decided to exclude the cutting plane version in our experiments and also omit further description here. We refer the reader to the work of Uchoa et al. [37] for more details on the cutting planes.

3.3.1. Transformation of the cable input

Since the MLCMST, and hence the formulation of Uchoa et al. [37], only allows the installation of one cable type along an edge, we define new cable types as combinations of the cable types given by the input. In order to reduce the complexity we do not consider all possible combinations but the cost minimal combination for each possible, integral amount of flow f_e along one edge e.

This local problem of finding the minimum cost cable combination which supports flow f is equivalent to the Minimum Integer Knapsack Problem (MinIKP) (or as we call it from here on the Minimum Unbounded Knapsack Problem (MinUKP)), as Salman et al. [33] state.

Following Kellerer et al. [22] we define the MinUKP as follows: Given a set if item types $N = \{1, ..., n\}$, with weight w_i and cost p_i and a minimum weight c find the cost minimal combination of item types $i \in N$ such that the weight of all item types is at least as big as the minimum weight c. Thereby, an unlimited number of copies of every item type can be used. The ILP formulation of the problem looks the following [22, p. 211],[22, p. 218]:

$$\min \sum_{i=1}^{n} p_i \cdot x_i \qquad [3.37]$$

$$\text{s.t.} \sum_{i=1}^{n} w_i \cdot x_i \geq c \qquad [3.38]$$

$$x_i \in \mathbb{N}_0 \qquad i = 1, ..., n \quad [3.39]$$

In order to solve the MinUKP we adapt the dynamic programming approach for the Unbounded Knapsack Problem (UKP) presented by Kellerer et al. [22, p. 220]. The pseudo-code for our adaption is provided in Algorithm 3.3.1. The basic idea of the approach is to loop over all input items $i \leq n$ and check for every possible knapsack capacity $\mu \leq M$ whether the addition of that item leads to a less costly knapsack item set. Therefore, the currently best cost value and the item whose addition to the knapsack item set leads to

Algorithm 3.3.1 Dynammic programming approach for MinUKP

```
 1: function MinUKP($weights$, $costs$, $M$)

 2:     $value[0] \leftarrow 0$                                    ▷ Initialization
 3:     $item[0] \leftarrow 0$
 4:     for $\mu \leftarrow 0$ to $M$ do        ▷ Initialize strucutres value and item with first item
 5:         $value[\mu] \leftarrow \lceil \frac{\mu}{weights[1]} \rceil \cdot costs[1]$
 6:         $item[\mu] \leftarrow 1$
 7:     end for

 8:     for $k^i \leftarrow 2$ to $|K|$ do          ▷ Generate optimal item per value combination
 9:         for $\mu \leftarrow 1$ to $M$ do
10:             if $value[\mu - weights[k^i]] + costs[k^i] < value[\mu]$ then
11:                 $value[\mu] \leftarrow value[\mu - weights[k^i]] + costs[k^i]$
12:                 $item[\mu] \leftarrow k^i$
13:             end if
14:         end for
15:     end for

16:     $K^* \leftarrow \emptyset$                        ▷ Calculate optimal set for capacity $M$
17:     $\mu^* \leftarrow M$
18:     repeat
19:         $k^i \leftarrow items[\mu^*]$
20:         $K^* \leftarrow K^* \cup \{k^i\}$
21:         $\mu^* \leftarrow \mu^* - values[k^i]$
22:     until $\mu^* \leq 0$

23:     return $K^*$
24: end function
```

this value are stored in two containers, which are called *value* and *item* in the pseudo-code description.

The algorithm then proceeds as follows. In an initialization phase we implement item 1 as the currently best item for all capacities $1 \leq \mu \leq M$. Looping through the rest of all items we check for every capacity μ whether the current minimum costs $value[\mu]$ can be improved by adding item i. This can be done by checking if $value[\mu - weights[i]] + costs[i] < value[\mu]$. This inequality is true if the current costs $value[\mu]$ for a capacity μ is greater than the cost of item k^i plus the cost for the rest capacity needed to equal to or surpass capacity μ. If we can improve the current solution we update the corresponding values in *value* and *item*. After we have calculated the optimal combinations per capacity, we can calculate the optimal item set by reversely traversing the items needed to reach capacity M.

Since Algorithm 3.3.1 calculates the optimal item combination for every capacity $\mu \leq M$, we can access the minimum cost item set in that range for every weight f once the containers *value* and *item* are calculated. Thus, if we now define the items' weights as the cable types' capacities, the items' costs as the cable types' costs and the knapsack's maximum capacity as the total turbine weight B we can directly map which flow is supported cost minimally by which cable combination for every possible flow in the network.

Algorithm 3.4.1 MstHeuristic

```
1: function MstHeuristic($V, K$)
2:     $P \leftarrow \{-1, \forall u^\nu \in V\}$
3:     $F \leftarrow \{-1, \forall u^\nu \in V\}$
4:     $\Lambda \leftarrow \{-1, \forall u^\nu \in V\}$
5:     $\alpha \leftarrow \mathrm{MST}(V)$                              ▷ Adjacency list of MST result
6:     $FCM \leftarrow \mathrm{MinUKP}(\{\mu_i : k^i \in K\}, \{c_i : k^i \in K\}, B)$
7:     $(P, F, \Lambda) \leftarrow \mathrm{CableInstallation}(V, P, F, \Lambda, FCM, \alpha, r, -1)$
8: end function
```

3.4. A simple minimum spanning tree heuristic

In this section we describe a first simple heuristic we want to use to find a solution to the SCIP. The pseudo-code of this algorithm is given in Algorithm 3.4.1.

Given an input instance with points V and cables K the heuristic first calculates an MST on all points. In a second step it calculates the flow from each turbine to the sink induced by the MST and then determines the cable installation along each arc using Algorithm 3.3.1.

Algorithm 3.4.1 operates on a tree all the time. Hence, we save a parent node p_ν, the outgoing flow to the parent f_ν and the cable combination carrying this flow λ_ν in the arrays P, F and Λ for each node $u^\nu \in V$. These arrays then comprise the final solution at the end of the algorithm.

In the following we have a closer look at the progress of the algorithm. First all data structures for storing the solution are initialized. Then we calculate the MST and store it in an adjacency list α.

Once we have calculated the MST we need to find the flow and corresponding cable installation on it. Therefore, we first run Algorithm 3.3.1 and save the flow to cable combination mapping int FCM.

In a next step we use the recursive function presented in Algorithm 3.4.2 to direct edges such that there is a path from every turbine to the sink. Each call of CableInstallation is parametrized with the node who should be directed and its parent node. The heuristic starts at the sink and traverses the MST tree in a depth-first manner. For the current node under observation we calculate its outgoing flow f_ν by adding its children's uplink flow to its node weight. The children's uplink flow is calculated in the further depth-first traversion. Hence, we call Algorithm 3.4.2 for each child node with the current node u^ν defined as the parent node during the process of calculating f_ν. At a leaf node the uplink flow only consists of the node's weight. Once the outgoing flow is calculated, the parent node is set and the cable combination for the outgoing edge is determined with the help of the flow-cable combination mapping. Obviously, we do not set any parent node, outgoing flow or cable installation for the sink. The updated arrays P, F and Λ are then returned.

After the execution of Algorithm 3.4.2 Algorithm 3.4.1 terminates. The solution of this heuristic is not necessarily feasible since the unsplittability constraint might be violated due to working on the Algorithm 3.3.1 flow-cable combination mapping.

The idea of using the MST algorithm for approaching the cable installation in wind farms has also been used by Dutta [9].

Algorithm 3.4.2 Cable installation along the MST

```
1: function CableInstallation($V, P, F, \Lambda, FCM, \alpha, u^\nu, u^\iota$)
2:     $f_\nu \leftarrow b_\nu$
3:     for $u^\omega \in \alpha[u^\nu]\backslash\{u^\iota\}$ do        ▷ Increase outgoing flow of $u^\nu$ by children's flow
4:         $(P, F, \Lambda) \leftarrow$ CableInstallation($V, F, \Lambda, FCM, \alpha, u^\omega, u^\nu$)
5:         $f_\nu \leftarrow f_\nu + f_\omega$        ▷ Add children's flow
6:     end for
7:     if $u^\nu \neq r$ then
8:         $p_\nu \leftarrow u^\iota$
9:         $\lambda_\nu \leftarrow FCM[f_\nu]$        ▷ Set cable combination according to the outgoing flow
10:    end if
11:    return $(P, F, \Lambda)$
12: end function
```

3.5. A randomized aggregation heuristic

Grandoni et al. [16] present in [16] an improvement of their work in [15] which is the best approximation algorithm for the splittable version of the SSBB. Using the fact that the transformation of a splittable solution to an unsplittable one is only twice as bad the optimal solution of the unsplittable optimum the algorithm has also the best approximation ratio for the unsplittable case. Hence, we adapted the approximation algorithm to our problem using the splittable algorithm itself and its transformation to the unsplittable case, as outlined in a proof in [16]. Please note that the unsplittable version of the SSBB problem does not correspond with our definition of unsplittability.

The core idea of the iterative approach by Grandoni et al. [16] is the aggregation of flow at a randomly chosen set of points. Therefore, in each iteration a collection of flow at a subset of points, the subsequent aggregation of flow along a Steiner tree and a randomized redistribution of the aggregated flow is performed. For the aggregation step they use the aggregation algorithm presented by Gupta et al. [18]. The main loop of the algorithm runs over a subset of cable types. In each iteration the currently regarded cable and the one with the next higher capacity is installed.

Our adaption of the algorithm follows the idea of iteratively collecting, aggregating and redistributing the flow. Though, we will thereby abolish the limitation to only a subset of cables and allow the installation of different cable types during the aggregation step in each iteration. In addition, for our problem it is sufficient to calculate a MST instead of a Steiner tree for the aggregation. Furthermore, we will transform the solution found after the last iteration to a tree which will not contain unnecessary flows introduced during the algorithm execution.

In the following we will go into more details about the algorithm which is also presented in pseudo-code in Algorithm 3.5.3. As an input the set of nodes and the set of cables sorted increasingly by capacity are given. The nodes' weight is either 0 or 1. If not we achieve this by duplicating the node b_ν times. Since we regard the SCIP we take this requirement as granted. The algorithm then starts with initializing its data structure as can be seen in Algorithm 3.5.2.

For the application of the aggregation algorithm of Gupta et al. [18] we have to make sure that the total aggregated flow, i.e. the total turbine weight, is a multiple of all cable capacities. For problem instances for which this is not automatically the case we can still meet this requirement by adding dummy turbines at the sink. In order to identify the

Algorithm 3.5.1 RandomizedAggregationHeuristic

```
1: function RandomizedAggregationHeuristic($V, K$)
2:     $(V', F, \Lambda, D, M, S, FCM) \leftarrow$ Initialize($V, K$)

3:     for $i = 0 ... |K|$ do
4:         $(F, \Lambda, D, M, S) \leftarrow$ Collection($V', K, i, F, \Lambda, D, M, S$)
5:         $(F, \Lambda, D) \leftarrow$ Aggregation($V', K, i, F, \Lambda, FCM, D$)
6:         $(F, \Lambda, D) \leftarrow$ Redistribution($V', K, i, F, \Lambda, D, M, S$)
7:     end for

8:     $(F, \Lambda) \leftarrow$ TransformSolution($F, \Lambda, FCM$)
9: end function
```

Algorithm 3.5.2 Initialization procedure for RandomizedAggregationHeuristic

```
1: function Initialize($V, K$)
2:     $\tilde{V} \leftarrow$ CreateDummyNodes($V, K$)                    ▷ Create dummy nodes
3:     $V' \leftarrow V \cup \tilde{V}$
4:     $F \leftarrow \{0, \forall (u^\nu, u^\iota) \in V' \times V'\}$                  ▷ Flow matrix
5:     $\Lambda \leftarrow \{\emptyset, \forall u^\nu \in V'\}$                ▷ Child node-incoming cable pairs
6:     $FCM \leftarrow$ MinUKP($\{\mu_i : k^i \in K\}, \{c_i : k^i \in K\}, B$)
7:     $D \leftarrow \{b_\nu, \forall t^\nu \in T \cup \tilde{V}\} \cup \{0, \forall u^\nu \in V \setminus T\}$        ▷ Aggregated demand
8:     $M \leftarrow V' \setminus \{r\}$
9:     $S \leftarrow \emptyset$
10:    return $(V', F, \Lambda, D, M, S, FCM)$
11: end function
```

number of dummy turbines needed we calculate the least common multiple of all cable capacities which is greater or equal to the total turbine weight and subtract the total turbine weight from it. Let $\tilde{V}$ be the set of dummy turbines. Then the adapted set of nodes V' is the union of the original set of nodes V and $\tilde{V}$.

During the algorithm we will save the flow between all nodes including the dummy ones in an adjacency matrix F and the cable installation in Λ. Thereby, Λ stores all incoming cables and the corresponding start node for every node $u^\nu \in V'$. Furthermore, D holds the currently aggregated flow at a node. Initially, this value is set to the turbine's weight b_ν for every turbine $t^\nu \in T \cup \tilde{V}$ and 0 for every other node. Throughout the algorithm we will mark nodes at which we collect flow. The array M stores those marked nodes u^ν and with the lists $s_\nu \in S$ we keep track of all nodes which send their flow to the corresponding marked node. In the initial iteration every node except the sink is marked.

In addition, we calculate the flow-cable combination mapping for the given cables and all possible flows with maximum flow being B at the beginning of the algorithm. Again, we will use Algorithm 3.3.1 for this. We will use this mapping throughout the aggregation step and the final transformation of the solution.

After these initial steps we start the main loop. In the main loop we consecutively process the steps of collecting, aggregating and redistributing flow. In the following let $k^0 = k^1$.

Collection In the collection step we mark a subset of nodes randomly and send all aggregated flow from the non-marked nodes to the closest marked node. Let i be the index of the cable k^i of K we currently regard. We first reset the set of marked nodes M to

Algorithm 3.5.3 Collection step of RandomizedAggregationHeuristic

```
function Collection($V', K, i, F, \Lambda, D, M, S$ )
    if $i \neq 0$ then
        $M \leftarrow \emptyset$
        if $i \neq |K|$ then
            for $u^\nu \in V', d_\nu > 0$ do                    ▷ Randomly mark nodes
                if RandomNumber$(0, 1) < \frac{0.531 \cdot \mu_i}{\mu_{i+1}}$ then
                    $M \leftarrow M \cup \{u^\nu\}$
                end if
            end for
        end if
        $M \leftarrow M \cup \{r\}$
        $S \leftarrow \{s_j = \emptyset : j = 0...|M|\}$
        for $u^\nu \in V' \setminus M, d_\nu > 0$ do     ▷ Collect flow of non-marked at marked nodes
            $u^\iota \leftarrow \arg\min_{u^\omega \in M}\{dist(u^\nu, u^\omega)\}$
            $s_\iota \leftarrow s_\iota \cup \{u^\nu\}$
            $F[u^\nu][u^\iota] \leftarrow F[u^\nu][u^\iota] + d_{u^\nu}$
            $\Lambda[u^\iota] \leftarrow \Lambda[u^\iota] \cup \{(u^\nu, \{K_i\})\}$
            $d_\iota \leftarrow d_\iota + d_\nu$
            $d_\nu \leftarrow 0$
        end for
    end if
    return $(F, \Lambda, D, M, S)$
end function
```

an empty set. Looping through all nodes with a flow surplus we randomly mark nodes by adding them to M. Every node gets marked with a probability of $\frac{\alpha \cdot \mu_i}{\mu_{i+1}}$, where we choose $\alpha = 0.531$ in analogy to Grandoni et al. [16]. Furthermore, we add the sink r to the set of marked nodes.

In the next step we collect the flow from the non-marked nodes. Therefore, we first initialize the set S and initialize an empty set for every node in M. Going over all non-marked nodes u^ν with positive aggregated flow we calculate the closest marked node u^ι. We add u^ν to s_ι and update the flow from u^ν to u^ι by adding d_ν to the current flow along that arc. Furthermore, we add a new incoming node-cable pair to λ_{u^ι}. The origin of the cable is u^ν and the cable type is k^i. Note that this cable type is sufficient since at all nodes with $d_\omega > 0$ it holds that $d_\omega = \mu_i$ due to the redistribution step of the last round. At last we adapt the aggregated flow at nodes u^ν and u^ι.

Aggregation For the aggregation part of an iteration we regard all nodes with aggregated flow surplus. On these nodes we calculate a MST. We perform the aggregation algorithm using this MST.

The aim of the *aggregation algorithm* is to push the flow along the MST such that at each node of the MST the aggregated flow afterwards is either 0 or a multiple of the capacity of the next higher cable type k^{i+1}. Hence, we make a copy of the aggregated nodes for all marked nodes, i.e. nodes u^ν with $d_\nu > 0$, and adapt it such that each node's aggregated demand in this copy is $d_\iota \mod \mu_{i+1}$. The part of the aggregated demand which we now do not consider anymore is therefore dividable by μ_{i+1} and can be redistributed completely by using copies of cable type k^{i+1}. The capacity of these cables is then fully exhausted.

Let d'_ν be the remaining aggregated flow at each node of the MST. Working with d'_ν the

Algorithm 3.5.4 Aggregation step of RandomizedAggregationHeuristic

```
function Aggregation($V', K, i, F, \Lambda, FCM, D$ )
    if $i < |K|$ then
        $\alpha \leftarrow \text{MST}(\{u^\nu \in V' : d_\nu > 0\})$
        $D' \leftarrow \{d_\nu \mod \mu_{i+1} : d_\nu > 0\}$
        $F' \leftarrow$ AggregationAlgorithm($\alpha, D', r, \mu_{i+1}$)
        for $(u^\nu, u^\iota) : F'[u^\nu][u^\iota] > 0$ do
            $F[u^\nu][u^\iota] \leftarrow F[u^\nu][u^\iota] + F'[u^\nu][u^\iota]$
            $\Lambda[u^\iota] \leftarrow \Lambda[u^\iota] \cup \{(u^\nu, FCM[F'[u^\nu][u^\iota]])\}$
        end for
    end if
    return $(F, \Lambda, D)$
end function
```

aggregation algorithm pushes the flow along the MST until each node is has an aggregated demand of 0 or μ_{i+1}. Thereby, the algorithm processes the following way. First we calculate a random barrier Y as a random integer between 1 and μ_{i+1}. We use Y later for choosing the nodes at which the remaining aggregated flow is assembled. Then we traverse the MST in a depth-first manner starting at the sink. In a counter Q we accumulate the flow we have pushed through the tree so far. At each node u^ν which we have not visit so far we add d_ν to Q. Let Q_{old} be the value of the counter before this addition and Q_{new} after the addition. If there is a value x for which $x \cdot \mu_{i+1} + Y \in (Q_{old}, Q_{new}]$ we choose u^ν to be one of the nodes at which the flow is aggregated. Otherwise, the flow at the node is sent further along the depth-first cycle to the next chosen node. Using the set of chosen nodes and the cycle induced by the depth-first traversing we get a flow such that at a randomly chosen set of points the aggregated flow is μ_{i+1} and at every other node it is 0.

However, it is possible that we use one edge of the MST in two directions for a path because we visit every edge twice while building the depth-first cycle. Though, we can limit the usage to one direction by identifying edges $\{u^\nu, u^\iota\}$ with flow in both directions and then reducing the flow on both arcs. The reduction is done such that the flow on the arc with lower flow, let's say arc (u^ν, u^ι), is set to 0 and the flow on arc (u^ι, u^ν) is adjusted to the difference of the flow over (u^ι, u^ν) and the flow over (u^ν, u^ι). After this process of flow reduction the paths only contain necessary flows.

Having calculated the flow for an aggregation of μ_{i+1} amounts of flow at a random subset of the marked nodes we install this flow in F for every arc with additional flow. We add the according cable to support this new flow as well by adding the sender node and the cable combination mapping entry for this flow in Λ.

The aggregation algorithm of course only works if the total aggregated flow in the network, which is equivalent to the total turbines' weight B, is a multiple of μ_{i+1}. This is the reason why we create the dummy turbines in the initialization step.

Redistribution After the aggregation the set of marked nodes M contains only nodes with aggregated demand d_ν equaling a multiple of μ_{i+1} or 0. In the redistribution step we send the aggregated flow of every currently marked node $u^\nu \in M$ to a random subset of nodes $u^\iota \in s_\nu$ which sent their flow to u^ν in the collection step. Therefore, we select $\frac{d_\nu}{\mu_{i+1}}$ nodes of $s_\nu \cup \{u^\nu\}$ randomly. If we regard the sink we only chose the nodes from the set s_r and do not include the sink itself. Let s'_ν be the random subset. We now send μ_{i+1} amount of flow from u^ν to every $u^\iota \in s'_\nu$. The flow is supported by installing cable type μ_{i+1} along (u^ν, u^ι).

Algorithm 3.5.5 Redistribution step of RandomizedAggregationHeuristic

```
1: function Redistribution($V', K, i, F, \Lambda, D, M, S$)
2:     if $i < |K|$ then
3:         for $u^\nu \in M$ do
4:             if $u^\nu \neq r$ then
5:                 $s'_\nu \leftarrow$ RandomSubset($s_\nu \cup \{u^\nu\}, \frac{d_{u^\nu}}{\mu_{i+1}}$)
6:             else
7:                 $s'_\nu \leftarrow$ RandomSubset($s_\nu, \frac{d_{u^\nu}}{\mu_{i+1}}$)
8:             end if
9:             for $u^\iota \in s'_\nu$ do
10:                if $u^\iota \neq u^\nu$ then
11:                    $F[u^\nu][u^\iota] \leftarrow F[u^\nu][u^\iota] + \mu_{i+1}$
12:                    $\Lambda[u^\iota] \leftarrow \Lambda[u^\iota] \cup \{(u^\nu, k^{i+1})\}$
13:                end if
14:            end for
15:        end for
16:    end if
17:    return $(F, \Lambda, D)$
18: end function
```

As can be seen from the pseudo-codes already, the main loop contains two special cases: the initial round, i.e. $i = 0$, and the final round, i.e. $i = |K|$. Basically, the initial and final phases only differ from the other rounds by excluding some parts of the iteration's steps.

In the initial round we omit the collection step. Hence, in the aggregation step all nodes are considered for the aggregation algorithm. After the redistribution step the aggregated flow at all nodes is either 0 or equals the capacity of the lowest capacity in the cable set K, namely k^1.

In the final round we do not do any marking of nodes. Hence, only the sink is regarded as a collection destination. Furthermore, we do not aggregate or redistribute flow in this round. Therefore, we send the aggregated flow directly over the shortest path to the sink.

Solution transformation The solution we get after the main loop is not necessarily a tree solution. Furthermore, we might install unnecessary high capacity cable types due to the collection and redistribution step where we first send flow to a marked node and later send flow back from it to some nodes. In addition, we also have to remove the additional flow from the dummy turbines.

Consequently, we do a post-processing step in the algorithm in analogy to the proof of lemma 2.1 and 2.3 in [16]. First, we direct all edges. This means that we identify all edges on which flow in both directions exists. We then reduce this flow the same way we do in the aggregation algorithm.

After this step we know that for each node in the network there is at least one path to the sink. We use this to identify all nodes which have more than one of those paths. At each node u^ν we depict two paths P_1 and P_2. We now have two possibilities: either delete one arc in P_1 or in P_2. In particular, this means that we find the arc with minimum flow of one path and reduce the path by that flow. The flow channeled out over this path from u^ν then has to reach the sink over the other path. This way we remove one path from u^ν to the sink . Using Algorithm 3.3.1 every flow adaption induces a cable combination which replaces the current one. We calculate the benefit of the two deletions and implement the one which leads to the least costs. This procedure is done for all nodes with more than two paths to the sink until no more of these nodes exist. As a result we obtain a tree directed to the sink.

Using this tree we remove the dummy flow. Therefore, we traverse every path from a dummy turbine to the source and reduce the flow along it by the turbine's weight. We adapt the cables as well by installing the cable combination given by the mapping from Algorithm 3.3.1.

After this transformation the data structures for the flow, i.e. F, and for the cables, i.e. Λ, contain a tree solution to the input problem instance which has no dummy flow and unnecessary capacity surplus anymore.

3.6. Three splittable upgrade heuristics

In a next step we have implemented the three heuristics presented by Pappas et al. [29] for the MLCMST. The heuristics retain a tree directed from the turbines to the sink throughout their execution and iteratively improve the current solution. Therefore, they replace a node's outgoing cable by one with an equal or greater capacity and use the additional capacity for linking new children to the node. This step is called *node upgrade* by the authors. In each iteration the heuristics implement the node upgrade with the highest benefit and hence operate in a greedy manner.

The heuristics only allow the installation of one cable type along an edge. Therefore, we transform the input the same way we do in Subsection 3.3.1. Hence, a combination of cables supporting each possible amount of flow exists. The adapted set of cable types K' therefore contains B cable combinations with capacity $\mu_i = i$ again.

Since the heuristics operate on a tree we know that every node has exactly one outgoing edge to one parent node. We store this relation for every node. Therefore, we save the parent of node u^ν in p_ν, its outgoing flow in f_ν and the cable combination supporting this flow in λ_ν. For the further explanation let P_ν be the path from u^ν to the sink r defined as

$$P_\nu = \begin{cases} \{u^\nu\} & p_\nu = r \\ \{u^\nu\} \cup P_{p_\nu} & \text{otherwise} \end{cases}$$

Additionally, let V_ν be the nodes in the subtree of u^ν, i.e. $V_\nu = \{u^\iota \in V : u^\nu \in P_\iota\}$.

For every node u^ν there are costs when upgrading the uplink cable λ_ν to another cable k^i, namely $C^i_\nu = (c_{\lambda_\nu} - c_i) \cdot dist(u^\nu, p_\nu)$. Though, every upgrade might also yield some profit. For every node u^ι we can calculate the benefit of relinking it to a node u^ν, i.e. replacing its parent p_ι by u^ν, by $d_{\nu\iota} = c_{\lambda_\iota} \cdot (dist(u^\iota, p_\iota) - dist(u^\iota, u^\nu))$. Let H be the set of all nodes reconnected to u^ν. Then the total profit of a node upgrade at u^ν is $D^i_{\nu H} = \sum_{u^\iota \in H} d_{\nu\iota} - C^i_\nu$. During a node upgrade set H can only contain nodes which do not increase the uplink flow f_ν of u^ν higher than the least free capacity of the uplink cables along path P_ν.

In order to find the feasible upgrade set which maximizes the total profit we store for each node u^ν a list q_ν decreasingly sorted according to the value $\frac{d_{\nu\iota}}{f_\iota}$. The list only contains profit-flow ratios which are positive, i.e. only combinations with nodes whose reconnection to u^ν is profitable. Working with this list we add the nodes which are most profitable per flow to H first when looking for an optimal set of nodes to connect to u^ν.

The heuristics start from a star topology connecting all nodes directly to the sink. So initially p_ν is set to r, f_ν to b_ν and λ_ν to the cable with the minimum cost providing enough capacity for transporting f_ν flow. Iteratively, the most profitable upgrade with profit $D^i_{\nu H}$ and its corresponding set H, its cable type k^i and its node u^ν is determined and implemented if the profit is greater than 0. Thereby, not all nodes are considered for

Algorithm 3.6.1 Initialization function for UPGRADEHEURISTIC1, UPGRADEHEURISTIC2 and UPGRADEHEURISTIC3

```
 1: function INITIALIZE($V$, $K$)
 2:     $FCM \leftarrow$ MINUKP($\{\mu_i : k^i \in K\}, \{c_i : k^i \in K\}, B$)      ▷ Transformed cable input
 3:     for $u^\nu \in V \backslash \{r\}$ do                                   ▷ Initial star solution
 4:         $p_\nu \leftarrow r$
 5:         $f_\nu \leftarrow b_\nu$
 6:         $\lambda_\nu \leftarrow FCM[f_\nu]$
 7:     end for

 8:     for $u^\nu \in V \backslash \{r\}$ do                                   ▷ Initial profit lists
 9:         for $u^\iota \in V \backslash \{r, u^\nu\}$ do
10:             if $d_{\nu\iota} > 0$ then
11:                 $q_\nu$.INSERT($\frac{d_{\nu\iota}}{f_\iota}, u^\iota$)     ▷ Insert $u^\iota$ in sorted list according to $\frac{d_{\nu\iota}}{f_\iota}$
12:             end if
13:         end for
14:     end for

15:     return ($\{p_1 ... p_{|V|}\}, \{f_1 ... f_{|V|}\}, \{\lambda_1 ... \lambda_{|V|}\}, \{q_1 ... q_{|V|}\}, FCM$)
16: end function
```

node upgrades but a subset which we call the candidate set S. We explain the details for this algorithm in the following sections for every heuristic.

3.6.1. UpgradeHeuristic1

UPGRADEHEURISTIC1 is a non-unit demand version of the heuristic presented by GAMVROS ET AL. [12]. The application to the non-unit demand is done by working on sorted lists with regard to the value $\frac{d_{\nu\iota}}{f_\iota}$ instead of only $d_{\nu\iota}$. A further adaption is that the sorted profits are calculated at the beginning of the algorithm and adapted after each iteration when necessary instead of generating that sorting during the computation of $D^i_{\nu H}$.

The heuristic starts by calling Algorithm 3.6.1. In this initialization function we create the initial star solution and calculate all sorted lists q_ν for all nodes u^ν. Additionally, we create a candidate set S which is filled with all turbines t^ν.

Starting with the highest capacity, i.e. $i = B$, we compute in each iteration the most profitable upgrade set H for every node $u^\nu \in \{u^\iota \in S : \mu_{\lambda_\iota} \leq \mu_i\}$. In order to do so, we check for every node $u^\iota \in q_\nu$ whether its reconnection to u^ν would meet all capacity constraints along P_ν after replacing the uplink cable λ_ν by k^i. If so, u^ι is added to H. Since q_ν is sorted such that the node u^ι with the highest profit $d_{\nu\iota}$ is added first the computation of H increases the total profit $D^i_{\nu H}$ greedily but not necessarily optimally. The procedure of finding the most profitable set is also illustrated in pseudo-code in Algorithm 3.6.2.

Let $u^{\bar{\nu}}$ and $\bar{H}$ be the node and upgrade set which yield the largest total profit $D^i_{\bar{\nu}\bar{H}}$. If $D^i_{\bar{\nu}\bar{H}} > 0$ the node upgrade improves the current solution. Hence, we reconnect all nodes $u^\iota \in \bar{H}$ to $u^{\bar{\nu}}$ by firstly decreasing the uplink flow along P_{p_ι} by f_ι to take into account that less flow leaves the current parent of u^ι after the upgrade. Then we set the new parent of u^ι to $p_\iota = u^{\bar{\nu}}$. Furthermore, we increase the uplink flow $f_{u^{\bar{\nu}}}$ by $\sum_{u^\iota \in \bar{H}} f_\iota$ and set the new uplink cable of $u^{\bar{\nu}}$ to $\lambda_{u^{\bar{\nu}}} = k^i$. Finally, we update the values of all nodes $u^\iota \in \bar{H}$ in the sorted lists q_{u^ω} for all $u^\omega \in S$ and remove $u^{\bar{\nu}}$ from the candidate set S. This procedure is also shown in Algorithm 3.6.3. If on the other hand $D^i_{\bar{\nu}\bar{H}} \leq 0$ no positive total profit for

Algorithm 3.6.2 Greedy heuristic for finding the most profitable H for a given node u^ν and a cable k^i

```
1: function CalculateMaximumProfit($u^\nu$, $i$)
2:     $H \leftarrow \emptyset$
3:     $D^i_{u^\nu H} \leftarrow -C^i_\nu$
4:     $f'_\nu \leftarrow f_\nu$                         ▷ Temporary variable for saving the new uplink flow
5:     for $u^\iota \in q_\nu \cap S$ do                          ▷ Determine most profitable ($u^\nu$, $H$)
6:         if $\min\{\min_{u^\omega \in P_\nu}\{\mu_{\lambda_\omega} - f_\omega\} + f_\nu, i\} - f'_\nu \geq f_\iota$ then       ▷ Capacity constraint
7:             $H \leftarrow H \cup \{u^\iota\}$
8:             $D^i_{\nu H} \leftarrow D^i_{\nu H} + d_{\nu\iota}$
9:         end if
10:    end for
11:    return $(H, D^i_{\nu H})$
12: end function
```

Algorithm 3.6.3 Function implementing a node upgrade

```
1: function ImplementUpgrade($u^\nu$, $H$, $k^i$, $\{p_1...p_{|V|}\}$, $\{f_1...f_{|V|}\}$, $\{\lambda_1...\lambda_{|V|}\}$)
2:     $f'_\nu \leftarrow f_\nu$
3:     for $u^\iota \in H$ do                                          ▷ Reconnection of child nodes
4:         DecreaseUplinkFlow($u^\iota$)                            ▷ Decrease flow along $P_{p_\iota}$
5:         $p_\iota \leftarrow u^\nu$
6:         $f'_\nu \leftarrow f'_\nu + f_\iota$
7:     end for
8:     $\lambda_\nu \leftarrow k^i$                          ▷ Install cable and additional flow uplink of $u^\nu$
9:     $f_\nu \leftarrow f'_\nu$
10:    return $(\{p_1...p_{|V|}\}, \{f_1...f_{|V|}\}, \{\lambda_1...\lambda_{|V|}\}, \{q_1...q_{|V|}\})$
11: end function
```

the currently regarded capacity exists so we reduce the capacity by 1.

The iteration stops when the capacity i is set to 0. The pseudo-code for UpgradeHeuristic1 is presented in Algorithm 3.6.4.

Note that UpgradeHeuristic1 only considers leaf nodes of the current solutions for node upgrades since an upgraded node is removed from the candidate set. Additionally, the computation of the most profitable upgrade set H and node u^ν is fixed to the capacity, i.e. its cable combination, of the main loop. UpgradeHeuristic2 loosens this fixation on the cable combination in one iteration as we describe in the following section.

3.6.2. UpgradeHeuristic2

UpgradeHeuristic1 and UpgradeHeuristic2 only differ in one idea: in UpgradeHeuristic2 the fixation to one cable combination in one iteration is abolished. Hence, we no longer fix one cable type in the main loop but compute the largest profit in one iteration for every candidate node and for every cable combination whose capacity is equal or higher than the currently installed uplink capacity μ_{λ_ν} of the node. Consequently, we do not process Algorithm 3.6.2 only once for each node. We rather execute it for each relevant cable combination. This opens further opportunities to gain a more profitable combination of an upgrade set H, a node u^ν and a cable combination i.

Since the cable combinations are not fixed for each loop we do not increment one iteration variable anymore. As a consequence, we need another stop criteria: Now the main loop

Algorithm 3.6.4 UpgradeHeuristic1

```
 1: function UpgradeHeuristic1($V$, $K$)
 2:     $(P, F, \Lambda, Q, FCM) \leftarrow$ Initialize$(V, K)$
 3:     $S \leftarrow V \backslash \{r\}$
 4:     $i \leftarrow B$
 5:     repeat
 6:         $\bar{H} \leftarrow \emptyset$
 7:         $u^{\bar{\nu}} \leftarrow -1$
 8:         $D^i_{\bar{\nu}\bar{H}} \leftarrow 0$

 9:         for $u^\nu \in S$ do                              ▷ Determine most profitable $(u^\nu, H)$
10:             if $i \geq \mu_{\lambda_\nu}$ then
11:                 $(H, D^i_{\nu H}) \leftarrow$ CalculateMaximumProfit$(u^\nu, i)$
12:                 if $D^i_{\nu H} > D^i_{\bar{\nu}\bar{H}}$ then
13:                     $\bar{H} \leftarrow H$
14:                     $u^{\bar{\nu}} \leftarrow u^\nu$
15:                     $D^i_{\bar{\nu}\bar{H}} \leftarrow D^i_{\nu H}$
16:                 end if
17:             end if
18:         end for

19:         if $D^i_{\bar{\nu}\bar{H}} > 0$ then
20:             $(P, F, \Lambda) \leftarrow$ ImplementUpgrade$(u^{\bar{\nu}}, \bar{H}, i, P, F, \Lambda)$
21:             $S \leftarrow S \backslash \{u^{\bar{\nu}}\}$
22:             for $u^\iota \in H, u^\omega \in S$ do                ▷ Update the value of $u^\iota$ in $q_\omega$
23:                 $q_\omega$.update$(u^\iota, \frac{d_{\omega\iota}}{f_\iota})$
24:             end for
25:         else
26:             $i \leftarrow i - 1$
27:         end if
28:     until $l = 0$
29: end function
```

stops as soon as no positive $D^i_{u^\nu H}$ can be found.

The initialization and general process in each iteration stays therefore the same. Under all candidate nodes the largest total profit and the corresponding upgrade set H, node u^ν and cable combination i is determined. Since we regard several cable combinations in each iteration this computation regards more combinations than in UpgradeHeuristic1. The second step, i.e. the implementation of the node upgrade, stays the same as well.

Hence, the pseudo-code presented in Algorithm 3.6.5 displays mostly similarities with the one of UpgradeHeuristic1.

3.6.3. UpgradeHeuristic3

Taking a further step for more variability, UpgradeHeuristic3 allows the upgrade of previously upgraded nodes, i.e. of non-leaf nodes. This means that we no longer restrict node upgrades to nodes in a candidate set which only holds leaf nodes. In fact, we compute the total profit $D^i_{\nu H}$ for every node $u^\nu \in V \backslash \{r\}$.

However, this change has a deeper impact on the general heuristic than the one by

Algorithm 3.6.5 UpgradeHeuristic2

```
function UpgradeHeuristic2($V$, $K$)
    $(P, F, \Lambda, Q, FCM) \leftarrow$ Initialize$(V, K)$
    $S \leftarrow V \backslash \{r\}$
    repeat
        $\bar{H} \leftarrow \emptyset$
        $u^{\bar{\nu}} \leftarrow -1$
        $i^* \leftarrow -1$
        $D^i_{\bar{\nu}\bar{H}} \leftarrow 0$

        for $u^\nu \in S$ do                                  ▷ Determine most profitable $(u^\nu, H)$
            for $i = \mu_{\lambda_\nu}$ to $B$ do
                $(H, D^i_{\nu H}) \leftarrow$ CalculateMaximumProfit$(u^\nu, i)$
                if $D^i_{\nu H} > D^i_{\bar{\nu}\bar{H}}$ then
                    $\bar{H} \leftarrow H$
                    $u^{\bar{\nu}} \leftarrow u^\nu$
                    $i^* \leftarrow i$
                    $D^i_{\bar{\nu}\bar{H}} \leftarrow D^i_{u^\nu H}$
                end if
            end for
        end for

        if $D^i_{\bar{\nu}\bar{H}} > 0$ then
            $(P, F, \Lambda) \leftarrow$ ImplementUpgrade$(u^{\bar{\nu}}, \bar{H}, i^*, P, F, \Lambda)$
            $S \leftarrow S \backslash \{u^{\bar{\nu}}\}$
            for $u^\iota \in H, u^\omega \in S$ do                   ▷ Update the value of $u^\iota$ in $q_\omega$
                $q_\omega$.update$(u^\iota, \frac{d_{\omega\iota}}{f_\iota})$
            end for
        end if
    until $D^i_{\bar{\nu}\bar{H}} \leq 0$
end function
```

UpgradeHeuristic2. Since the previous variants only upgrade leaf nodes they do not introduce any cycles to the current solution. With the upgrade of non-leaf nodes this is now possible. Therefore, the computation of the largest profit $D^i_{\nu H}$ and the corresponding upgrade set H for a node u^ν and a cable combination k^i has to make sure no cycles are added. In addition, in UpgradeHeuristic1 and UpgradeHeuristic2 we can neglect the update of the profit-flow ratio values of nodes $u^\iota \in P_{\bar{\nu}} \cup \{P_{p_\omega} : u^\omega \in \bar{H}\}$ since they cannot be linked to any node anymore because they are all non-leaf nodes. In UpgradeHeuristic3 these nodes have to be updated for every sorted list q_ν as well.

The updated version of the largest profit computation is presented in Algorithm 3.6.6. The first change we can see here is in Line 2 where we create a temporary copy of the sorted profit-flow ratio list q_ν and remove every node $u^\iota \in P_{p_\nu}$ which is uplink of u^ν. The reason for this is twofold: On the one hand, we need a copy of q_ν since the addition of a node u^ι to H reduces the uplink flow f_ω of nodes $u^\omega \in P_{p_\iota}$. Hence, their value $\frac{d_{\nu\omega}}{f_\omega}$ in q_ν needs to be changed temporarily for the calculation of H and $D^i_{\nu H}$. On the other hand, we exclude nodes $u^\iota \in P_{p_\nu}$ from q'_ν in order to prevent the introduction of cycles to the solution by adding such a node u^ι as a child of u^ν in the further process. Going on, we loop over all

Algorithm 3.6.6 Greedy heuristic for finding the most profitable H for a given node u^ν and a cable k^i

```
1: function CalculateMaximumProfit($u^\nu$, $i$)
2:     $q'_\nu \leftarrow q_\nu \backslash P_{p_\nu}$                ▷ Temporary copy of $q_\nu$ excluding nodes in $P_{p_\nu}$
3:     $H \leftarrow \emptyset$
4:     $D^i_{\nu H} \leftarrow -C^i_\nu$
5:     $f'_\nu \leftarrow f_\nu$                ▷ Temporary variable for saving the new uplink flow
6:     while $q'_\nu$ not empty do                ▷ Determine most profitable ($u^\nu$, $H$)
7:         $u^\iota \leftarrow q'_\nu$.pop()
8:         if $\min\{i, \min_{u^\omega \in P_\nu}\{\mu_{\lambda_\omega} - f_\omega\} + f_\nu\} - f'_\nu \geq f_\iota$ then                ▷ Capacity constraint
9:             $H \leftarrow H \cup \{u^\iota\}$
10:            $D^i_{\nu H} \leftarrow D^i_{\nu H} + d_{\nu\iota}$
11:            for $u^\omega \in P_\iota \backslash \{u^\iota\} \cap q'_\nu$ do                ▷ Update the profit flow ratios in $q'_\nu$
12:                $q'_\nu$.update($u^\omega$, $\frac{d_{\nu\omega}}{f_\omega - f_\iota}$)
13:            end for
14:            for $u^\omega \in V_\iota \backslash \{u^\iota\} \cap q'_\nu$ do                ▷ Add nodes of $V_\iota$ with positive $d_{\nu\omega}$ directly
15:                $H \leftarrow H \cup \{u^\omega\}$
16:                $q'_\nu$.remove($u^\omega$)
17:            end for
18:        end if
19:    end while
20:    return $(H, D^i_{\nu H})$
21: end function
```

nodes $u^\iota \in q'_\nu$, pop the currently first one, i.e. the one with the highest profit-flow ratio, and check whether its addition to H meets the capacity constraints. If the constraints are met we update H and $D^i_{\nu H}$ as before. Additionally, we update the entries in q'_ν of nodes $u^\omega \in P_\iota$ since they have less uplink flow now. Moreover, we can add all nodes $u^\omega \in V_\iota \backslash \{u^\iota\}$ which are also in q_ν to H because their addition does not increase the flow from u^ι to u^ν but increases the total profit. In order to not regard them again we remove nodes u^ω from q'_ν after their addition.

As mentioned above already, UpgradeHeuristic3 has to update the profit-flow ratio of every node whose uplink flow changes due to the new node upgrade. Therefore, we update the ratio of all nodes in $U = P_{\bar{\nu}} \cup \{P_{p_\omega} : u^\omega \in \bar{H}\}$ in each list q_ι. This way we take into account all nodes whose uplink flow increase by the new uplink flow of $u^{\bar{\nu}}$, i.e. $u^\omega \in P_{\bar{\nu}}$, and the ones whose uplink flow decreases since they have less input, i.e. nodes $u^\omega \in \{P_{p_\omega} : u^\omega \in \bar{H}\}$.

3.7. An unsplittable upgrade heuristics

From UpgradeHeuristic3 we derive our own heuristic which works on the original cable set and makes sure that with each improvement the unsplittability constraint is met as well. We call this heuristic UnsplittableUpgradeHeuristic.

Working with the original cable set leads to several changes to the heuristics by Pappas et al. [29]. First of all we cannot store one uplink capacity for each node in λ_ν but need to provide a list of cable types for each node. Furthermore, we have to store the parent cable ψ^i_ν as well as the free capacity δ^i_ν for each uplink cable type k^i. These data structures document how the unsplittable flow is conserved during the heuristic's progress.

Algorithm 3.6.7 UpgradeHeuristic3

```
 1: function UpgradeHeuristic3($V$, $K$)
 2:     $(P, F, \Lambda, Q, FCM) \leftarrow$ Initialize$(V, K)$
 3:     repeat
 4:         $\bar{H} \leftarrow \emptyset$
 5:         $u^{\bar{\nu}} \leftarrow -1$
 6:         $i^* \leftarrow -1$
 7:         $D^i_{\bar{\nu}\bar{H}} \leftarrow 0$

 8:         for $u^\nu \in V\backslash\{r\}$ do                ▷ Determine most profitable $(u^\nu, H)$
 9:             for $i = \mu_{\lambda_\nu}$ to $B$ do
10:                 $(H, D^i_{\nu H}) \leftarrow$ CalculateMaximumProfit$(u^\nu, i)$
11:                 if $D^i_{\nu H} > D^i_{\bar{\nu}\bar{H}}$ then
12:                     $\bar{H} \leftarrow H$
13:                     $u^{\bar{\nu}} \leftarrow u^\nu$
14:                     $i^* \leftarrow i$
15:                     $D^i_{\bar{\nu}\bar{H}} \leftarrow D^i_{\nu H}$
16:                 end if
17:             end for
18:         end for

19:         if $D^i_{\bar{\nu}\bar{H}} > 0$ then
20:             $U \leftarrow \bar{H} \cup P_{\bar{\nu}} \cup \{P_{p_\omega} : u^\omega \in \bar{H}\}$      ▷ Nodes $u^\iota$ whose $\frac{d_{\omega\iota}}{f_\iota}$ changes
21:             $(P, F, \Lambda) \leftarrow$ ImplementUpgrade$(u^{\bar{\nu}}, \bar{H}, i^*, P, F, \Lambda)$
22:             for $u^\iota \in U, u^\omega \in V\backslash\{r\}$ do      ▷ Update the value of $u^\iota$ in $q_\omega$
23:                 $q_\omega$.update$(u^\iota, \frac{d_{\omega\iota}}{f_\iota})$
24:             end for
25:         end if
26:     until $D^i_{\bar{\nu}\bar{H}} \leq 0$
27: end function
```

Analogously to the path to the sink P_ν from a node u^ν, let $P_{\nu,i}$ be the path from node u^ν over the ith outgoing cable to the sink. Therefore we define $P_{\nu,i}$ as

$$P_{\nu,i} = \begin{cases} \{(u^\nu, i)\} & p_\nu = r \\ \{(u^\nu, i)\} \cup P_{p_\nu, \psi^i_\nu} & \text{otherwise} \end{cases}$$

Additionally, we now have two options on how to increase a capacity. On the one hand, we can replace a currently installed cable by one with an equal or greater capacity. On the other hand, there is the opportunity add a cable to the currently installed ones. This also impacts the cost calculation of the total profit $D^i_{\nu H}$. For a cable replacement of cable type k^i by k^j at node u^ν the costs are the same as for an upgrade in the heuristics of Pappas et al. [29], namely $C^{i,j}_\nu = (c_j - c_i) \cdot dist(u^\nu, p_\nu)$. The costs for an additional cable k^j at node u^ν, however, are the full cost of the cable type for the distance from u^ν to its parent p_ν, i.e. $C^j_\nu = c_j \cdot dist(u^\nu, p_\nu)$.

Moreover, we have to define now how we select the parent cable type for each uplink cable type of a potential child node during a node upgrade. Since we always improve an already found feasible, i.e. unsplittable, solution we know that for each cable k^j of a child node u^ι the unsplittability constraint is met in its subtree for all cables. The same holds

Algorithm 3.7.1 Initialization function UNSPLITTABLEUPGRADEHEURISTIC

```
 1: function INITIALIZE($V$, $K$)
 2:     for $u^\nu \in V \backslash \{r\}$ do                                    ▷ Initial star solution
 3:         $p_\nu \leftarrow r$
 4:         $f_\nu \leftarrow b_\nu$
 5:         $\lambda_\nu \leftarrow \{\arg\min_{k^i \in K}\{c_i : \mu_i \geq f_\nu\}\}$
 6:         $\delta_\nu \leftarrow \{\mu_{\arg\min_{k^i \in K}\{c_i : \mu_i \geq f_\nu\}} - f_\nu\}$
 7:         $\psi_\nu \leftarrow \{-1\}$
 8:     end for

 9:     for $u^\nu \in V \backslash \{r\}$ do                                    ▷ Initial profit lists
10:         for $u^\iota \in V \backslash \{r, u^\nu\}$ do
11:             if $d_{\nu\iota} > 0$ then
12:                 $q_\nu$.INSERT($\frac{d_{\nu\iota}}{f_\iota}$, $u^\iota$)        ▷ Insert $u^\iota$ in sorted list according to $\frac{d_{\nu\iota}}{f_\iota}$
13:             end if
14:         end for
15:     end for

16:     return $(\{p_1 \ldots p_{|V|}\}, \{f_1 \ldots f_{|V|}\}, \{\lambda_1 \ldots \lambda_{|V|}\}, \{q_1 \ldots q_{|V|}\}, \{\psi_1 \ldots \psi_{|V|}\}, \{\delta_1 \ldots \delta_{|V|}\})$
17: end function
```

for all outgoing cables and their connection to the sink over $P_{\nu,i}$. Connecting u^ι to node u^ν we therefore have to find for each k^j uplink of u^ι a cable k^i at u^ν for which the minimal free capacity δ^κ_ω for all cables k^κ along $P_{\nu,i}$ is at least as high as the uplink flow over cable k^j. Thereby, we use a first fit approach choosing always the first cable meeting this constraint. The flow of a cable which is replaced is channeled through the new cable as well such that only new children's flow has to be distributed.

Having presented the general idea and changes we now look into the algorithm in more detail. The pseudo-code of UNSPLITTABLEUPGRADEHEURISTIC can be found in Algorithm 3.7.8.

In an initialization step we create an initial star solution again. Since UNSPLITTABLE-UPGRADEHEURISTIC uses a different data structure Algorithm 3.7.1 differs from Algorithm 3.6.1. In a first step we set the parent of every node u^ν again to $p_\nu = r$ and the uplink flow f_ν to the node's weight b_ν. What changes is that we now save a list of uplink cables in λ_ν. This list initially contains only the cheapest cable k^i which has a capacity able to carry f_ν. For this cable we store the free capacity in δ_ν and set the parent cable in ψ_ν to -1 since the sink does not have any outgoing cables. In a next step we calculate the sorted profit-flow ratio lists. The calculation is the same as in the heuristics of PAPPAS ET AL. [29] (see Subsection 3.6.1).

After the initialization we start the main loop which we know from UPGRADEHEURISTIC2 and UPGRADEHEURISTIC3. This means we only stop when no further improvement can be found. In each loop the most profitable upgrade set $\bar{H}$, node $u^{\bar{\nu}}$, cable $k^{\bar{i}}$, total profit $D^{\bar{i}}_{\bar{\nu}H}$ are set to initial values. Furthermore, a boolean called *replacement* is initialized to indicate whether a node upgrade is done by a node replacement or a node addition.

The computation of the largest total profit now contains two similar steps for each node u^ν and cable k^i. First, the largest profit for an addition is computed and then the one for an replacement. If the newly found profit is greater than the currently best one then its parameters and its upgrade method are saved. Note that for the replacement policy we do

Algorithm 3.7.2 Greedy heuristic for finding the most profitable H for a given node u^ν and a cable k^i

```
1: function CalculateMaximumProfitAddition($u^\nu$, $k^i$)
2:     $\lambda'_\nu \leftarrow \lambda_\nu \cup \{k^i\}$
3:     $\delta'_\nu \leftarrow \{\min_{(u^\iota,\kappa)\in P_{\nu,j}}\{\delta^\kappa_\iota\}, \forall j \in [1...|\lambda_\nu|]\}$
                              ▷ Temporary list of free capacity along $P_\nu$ for all $k^j \in \lambda_\nu$
4:     $\delta'_\nu \leftarrow \delta'_\nu \cup \{\min\{\mu_i, \max_{(u^\iota,\kappa)\in P_{\nu,j}}\{\delta^\kappa_\iota, \forall j \in [1...|\lambda_{p_\nu}|]\}\}$

5:     return CalculateMaximumProfit($u^\nu$,$k^i$,$\lambda'_\nu$,$\delta'_\nu$, $C^{k^i}_\nu$)
6: end function
```

Algorithm 3.7.3 Greedy heuristic for finding the most profitable H for a given node u^ν and a cable k^i

```
1: function CalculateMaximumProfitReplacement($u^\nu$, $k^i$, $j$)
2:     $\lambda'_\nu \leftarrow \lambda_\nu$
3:     $\lambda'_\nu$.Replace($j, k^i$)              ▷ Replace cable at position $j$ with cable type $k^i$
4:     $\delta'_\nu \leftarrow \{\min_{(u^\iota,\tau)\in P_{\nu,\kappa}}\{\delta^\tau_\iota\}, \forall \kappa \in [1...|\lambda_\nu|]\}$
         ▷ Temporary list of free capacity along $P_\nu$ for all $k^j \in \lambda_\nu$ and additional cable $k^i$
5:     $\delta'_\nu$.Replace($j, \min\{\mu_{k^i} - f_{\nu,j}, \min_{(u^\iota,j)\in P_{p_\nu,j}}\{\delta^j_\iota\}$)  ▷ Replace entry at $j$ with least
   free capacity
6:     return CalculateMaximumProfit($u^\nu$,$k^i$,$\lambda'_\nu$,$\delta'_\nu$, $C^{k^i,\lambda^j_\nu}_\nu$)
7: end function
```

not only loop over all nodes and all cable types available for a replacement but also over all entries of the uplink cables λ_ν of u^ν to consider one cable for a replacement at a time.

In Algorithm 3.7.2 and Algorithm 3.7.3 the calculation of the largest profit for both methods is explained in more detail. Both methods create a temporary copy of the uplink cables λ_ν and of the available capacity along $P_{\nu,j}$ from node u^ν to the sink r for each cable entry λ^j_ν. The temporary copies in Algorithm 3.7.2 and Algorithm 3.7.3 differ in respect to their upgrading method.

In Algorithm 3.7.2 cable k^i is appended to the temporary cable list λ'_ν. Its corresponding entry in the available capacity list δ'_ν is set to the minimum between the cable's capacity μ_i and the highest uplink capacity available at the parent node p_ν. This way we account for the actual available uplink capacity for node u^ν.

The replacement idea in Algorithm 3.7.3 leads to a temporary uplink cable list λ'_ν in which entry j is replaced by cable type k^i. The available capacity in δ'_ν has to be adapted accordingly. Hence, we check whether the additionally implemented capacity $\mu_i - f_{\nu,j}$ or the available uplink capacity at $\psi_{\nu,j}$, namely $\min_{(u^\iota,j)\in P_{u^\nu,j}}\{\delta^j_{u^\iota}\}$, is the bottleneck for additional flow and set the smaller value of both as available capacity for $\lambda^j_{u^\nu}$.

The temporary copies are used by Algorithm 3.7.4 to check the unsplittability constraint. Both functions Algorithm 3.7.2 and Algorithm 3.7.3 call Algorithm 3.7.4 to calculate the largest profit with their settings.

Algorithm 3.7.4 then works similar to Algorithm 3.6.6. First of all, a temporary copy of the profit-flow ratio list is created to be able to account for flow changes during the addition of nodes to H. In addition, H is initialized by an empty set, the total profit $D^i_{\nu H}$ is set by the addition resp. replacement costs given by parameter C and a temporary copy of the uplink flow is created.

Algorithm 3.7.4 Greedy heuristic for finding the most profitable H for a given node u^ν and a cable k^i

```
1:  function CalculateMaximumProfit($u^\nu$, $k^i$, $\lambda'_\nu$, $\delta'_\nu$, $C$)
2:      $q'_\nu \leftarrow q_\nu \backslash P_{p_\nu}$                               ▷ Temporary copy of $q_\nu$
3:      $H \leftarrow \emptyset$
4:      $D^i_{\nu H} \leftarrow -C$
5:      $f'_\nu \leftarrow f_\nu$                   ▷ Temporary variable for saving the new uplink flow
6:      while $q'_\nu$ not empty do                     ▷ Determine most profitable ($u^\nu$, $H$)
7:          $u^\iota \leftarrow q'_\nu$.pop()
8:          if $\min\{\mu_{k^i}, \min_{u^\omega \in P_\nu}\{\mu_{\lambda_\omega} - f_\omega\} + f_\nu\} - f'_\nu \geq f_\iota$ then    ▷ Capacity constraint
9:              if UnsplitabilityConstraintMet($\lambda'_\nu, \delta'_\nu, f_\iota$) then
10:                 $H \leftarrow H \cup \{u^\iota\}$
11:                 $D^i_{\nu H} \leftarrow D^i_{\nu H} + d_{\nu\iota}$
12:                 for $u^\omega \in P_\iota \backslash \{u^\iota\} \cap q'_\nu$ do          ▷ Update the profit flow ratios in $q'_\nu$
13:                     $q'_\nu$.update($u^\omega$, $\frac{d_{\nu\omega}}{f_\omega - f_\iota}$)
14:                 end for
15:                 for $u^\omega \in V_\iota \backslash \{u^\iota\} \cap q'_\nu$ do    ▷ Add nodes of $V_\iota$ with positive $d_{\nu\omega}$ directly
16:                     $H \leftarrow H \cup \{\omega\}$
17:                     $q'_\nu$.remove($u^\omega$)
18:                 end for
19:                 for $j \in [1...|\lambda_\iota|]$ do
20:                     $\kappa \leftarrow$ FirstFitCable($\delta'_\nu, f_{\iota,j}$)    ▷ First $i \in [1...|\lambda'_\nu|]$ with $\mu_{\lambda'^\kappa_\nu} \geq f_{\iota,j}$
21:                     $\delta'^\kappa_\nu \leftarrow \delta'^\kappa_\nu - f_{\iota,j}$
22:                 end for
23:             end if
24:         end if
25:     end while
26:     return $(H, D^i_{\nu H})$
27: end function
```

After this initialization step the main loop of Algorithm 3.7.4 is processed. The loop only differs from the one in Algorithm 3.6.6 in the check for unsplittability in Line 9 and the update of δ'_ν in Line 19. In the unsplittability constraint we check for all uplink cables λ^κ_ι if there exists a cable λ'^j_ν with free capacity for the uplink flow $f_{\iota,\kappa}$ and the uplink flow of potentially added uplink flow $f_{\iota,\tau}$ for $\tau < \kappa$. Thereby, we take into account that our strategy to chose a parent cable is first fit. We note that this causes the check of unsplittability of a node to be false even if it would not cause a violation if another parent choice procedure was applied. If the unsplittability constraint is met we can add node u^ν to the upgrade set H and adapt the temporary list of available capacities as shown in Line 19. This means that we select the first entry κ in δ'_ν which can support the uplink flow $f_{\iota,j}$ and reduce the available capacity for entry κ by $f_{\iota,j}$. As before we return the upgrade set H and the corresponding largest profit $D^i_{\nu H}$ at the end. Note that the direct addition of nodes in the subtree V_ι of u^ι is still valid. The capacity at node u^ν occupied for each subtree node which was accounted for by the check of unsplittability can directly be used for the direct addition.

The next step in the algorithm is the implementation of the largest found profit. As in Algorithm 3.6.7 we update all entries in Q of nodes u^ι whose parent or uplink flow changed. The new part can be found in Algorithm 3.7.5 and Algorithm 3.7.6 in which the

Algorithm 3.7.5 Function implementing a node upgrade

```
1: function ImplementReplacement($u^\nu, H, k^i, j, P, F, \Lambda, \Psi, \Delta$)
2:     $\delta_\nu^j \leftarrow \delta_\nu^j + \mu_i - \mu_{\lambda_\nu^j}$
3:     $\lambda_\nu^j \leftarrow k^i$
4:     return ImplementUpgrade($u^\nu, H, k^i, P, F, \Lambda, \Psi, \Delta$)
5: end function
```

Algorithm 3.7.6 Function implementing a node upgrade

```
1: function ImplementAddition($u^\nu, H, k^i, P, F, \Lambda, \Psi, \Delta$)
2:     $\lambda_\nu \leftarrow \lambda_\nu \cup \{k^i\}$
3:     $\psi_\nu \leftarrow \psi_\nu \cup \{\text{FirstFitCable}(\min\{\mu_i, \max_{(u^\iota,\kappa)\in P_{p_\nu,j}}\{\delta_\iota^\kappa, \forall j \in [1, |\lambda_{p_\nu}|]\}\})\}$
4:     $\delta_\nu \leftarrow \delta_\nu \cup \{\mu_i\}$
5:     return ImplementUpgrade($u^\nu, H, k^i, P, F, \Lambda, \Psi, \Delta$)
6: end function
```

implementation of the replacement resp. addition is done. Depending on which upgrade method was used for achieving the largest profit the corresponding function is called.

Again these two functions only differ slightly and the main computation is done by the same function, namely Algorithm 3.7.7. For a replacement Algorithm 3.7.5 adapts the $\lambda_{u^\nu}^j$ at position j and replaces the current cable with new cable k^i. The free capacity entry $\delta_{u^\nu}^j$ is adapted accordingly by adding the gained capacity surplus. If a cable k^i is added Algorithm 3.7.6 appends k^i to λ_ν and creates the corresponding entries in δ_ν and ψ_ν. The parent cable is the outgoing cable at the parent node found first which covers the new capacity or which has the most free capacity.

After these adjustments to the data structure the rest of the implementation is done by Algorithm 3.7.7. First we make a copy of the current uplink flow of u^ν. This copy is increased by the flow from every $u^\iota \in H$ and later on set as the new uplink flow. For every node $u^\iota \in H$ we first decrease the flow uplink its current parent. We do this for the total flow f_ι as well as for every cable uplink flow which means that we increase the free capacity δ_ν of all $(u^\omega, j) \in P_{p_\iota,\kappa}$ for all $\kappa \in \lambda_{p_\iota}$.

Furthermore, we connect u^ι to u^ν by setting $p_{u^\iota} = u^\nu$ for every node in H. The parent cable λ_ν^τ of each cable $k^i \in \lambda_\iota$ is determined via first fit. The free capacity uplink of that parent cable is decreased by the uplink flow from u^ι on cable $\lambda_{u^\iota}^\tau$. At last we increase the additional overall flow.

After these adaptions have been done for all nodes $u^\iota \in H$ we update the uplink flow with the additional flow from nodes u^ι and return the updated data structures.

The main loop stops when no improvement can be achieved. The cable installation can then be derived from P and Λ again.

Algorithm 3.7.7 Function implementing a node upgrade

```
function ImplementUpgrade($u^\nu, H, k^i, P, F, \Lambda, \Psi, \Delta$)
    $f'_\nu \leftarrow f_\nu$
    for $u^\iota \in H$ do                                    ▷ Reconnection of child nodes
        DecreaseUplinkFlow($u^\iota$)
        $p_\iota \leftarrow u^\nu$
        for $j \in [1...|\lambda_\iota|]$ do
            $\tau \leftarrow$ FirstFitCable($f_{\iota,j}$)
            $\psi_\iota^j \leftarrow \tau$
            DecreaseFreeUplinkCapacity($u^\nu$,$\tau$, $f_{\iota,j}$)
        end for
        $f'_\nu \leftarrow f'_\nu + f_\iota$
    end for
    $f_\nu \leftarrow f'_\nu$
    return ($\{p_1...p_{|V|}\}, \{f_1...f_{|V|}\}, \{\lambda_1...\lambda_{|V|}\}, \{q_1...q_{|V|}\}, FCM$)
end function
```

Algorithm 3.7.8 UnsplittableUpgradeHeuristic

```
1: function UnsplittableUpgradeHeuristic($V$, $K$)
2:     $(P, F, \Lambda, Q, \Psi, \Delta) \leftarrow$ Initialize$(V, K)$
3:     repeat
4:         $\bar{H} \leftarrow \emptyset$
5:         $u^{\bar{\nu}} \leftarrow -1$
6:         $k^{\bar{i}} \leftarrow -1$
7:         $D^{\bar{i}}_{\bar{\nu}\bar{H}} \leftarrow 0$
8:         $replacement \leftarrow FALSE$

9:         for $u^{\nu} \in V\backslash\{r\}$ do                ▷ Determine most profitable ($u^{\nu}$, $H$)
10:            for $k^i \in K$ do
11:                $(H, D^i_{\nu H}) \leftarrow$ CalculateMaximumProfitAddition$(u^{\nu}, k^i)$
12:                if $D^i_{\nu H} > D^{\bar{i}}_{\bar{\nu}\bar{H}}$ then
13:                    $\bar{H} \leftarrow H$
14:                    $u^{\bar{\nu}} \leftarrow u^{\nu}$
15:                    $k^{\bar{i}} \leftarrow k^i$
16:                    $D^{\bar{i}}_{\bar{\nu}\bar{H}} \leftarrow D^i_{\nu H}$
17:                    $replacement \leftarrow FALSE$
18:                end if
19:                for $j \in [1...|\lambda_{\nu}|], \mu_{\lambda^j_{\nu}} \leq \mu_{k^i}$ do
20:                    $(H, D^i_{\nu H}) \leftarrow$ CalculateMaximumProfitReplacement$(u^{\nu}, k^i, j)$
21:                    if $D^i_{\nu H} > D^{\bar{i}}_{\bar{\nu}\bar{H}}$ then
22:                        $\bar{H} \leftarrow H$
23:                        $u^{\bar{\nu}} \leftarrow u^{\nu}$
24:                        $k^{\bar{i}} \leftarrow k^i$
25:                        $j^* \leftarrow j$
26:                        $D^{\bar{i}}_{\bar{\nu}\bar{H}} \leftarrow D^i_{\nu H}$
27:                        $replacement \leftarrow TRUE$
28:                    end if
29:                end for
30:            end for
31:        end for

32:        if $D^{\bar{i}}_{\bar{\nu}\bar{H}} > 0$ then
33:            $U \leftarrow \bar{H} \cup P_{\bar{\nu}} \cup \{P_{p_{\omega}} : u^{\omega} \in \bar{H}\}$     ▷ Nodes $u^{\iota}$ whose $\frac{d_{\omega\iota}}{f_{\iota}}$ changes
34:            if replacement then
35:                $(P, F, \Lambda, \Psi, \Delta) \leftarrow$ ImplementReplacement$(u^{\bar{\nu}}, \bar{H}, k^{\bar{i}}, j^*, P, F, \Lambda, \Psi, \Delta)$
36:            else
37:                $(P, F, \Lambda, \Psi, \Delta) \leftarrow$ ImplementAddition$(u^{\bar{\nu}}, \bar{H}, k^{\bar{i}}, P, F, \Lambda, \Psi, \Delta)$
38:            end if
39:            for $u^{\iota} \in U, u^{\omega} \in V\backslash\{r\}$ do          ▷ Update the value of $u^{\iota}$ in $q_{\omega}$
40:                $q_{\omega}$.update$(u^{\iota}, \frac{d_{\omega\iota}}{f_{\iota}})$
41:            end for
42:        end if
43:    until $D^{\bar{i}}_{\bar{\nu}\bar{H}} \leq 0$
44: end function
```

4. Experiments

In the following we will discuss the experiments done for the approaches from Chapter 3. We first describe the problem instances for the experiments. Afterward, we present the results of the experiments separately for the MILP resp. ILP formulations and the heuristics.

4.1. Benchmark instances

For our experiments we created several types of problem instances to test the algorithms on different settings. The parameters which differ are the number of turbines, the turbine distribution, the sink distribution, the number of cable types and the characteristic of the cable types.

We created sets with 25, 50 and 100 turbines. Thereby, we used two different types of distribution. The first one was a uniform random placement of points in a defined area in the Euclidean plane. The second one was oriented at the actual positioning of turbines. To distinguish the two distributions better in the following we call them random distribution and array distribution, respectively.

The geometry of a wind farm depends mostly on the optimal wind capture aiming at the reduction of energy capture losses [26, p. 423]. One key aspect of this is the downwind and crosswind spacing between turbines. Hau [19, p. 733] state that in downwind direction a distance of eight to ten times the rotor diameter and in crosswind three to five times the rotor diameter is reasonable. Apart from that the topography and the land boundary of the area for the wind farm are factors affecting the position of the wind turbines in a wind farm [19, p. 732]. From an expert's input and the data provided in [30] we derive that we can approximate the array of turbines by groups of parallel lines along which the turbines are distributed.

Hence, we simulated the turbine positions for the array distribution as follows: We created a random number of groups of turbines. For each group a random number of parallel lines was created along which the points were distributed. The number of turbines for each parallel line was chosen at random as well. The distance between two lines depends on the rotor diameter. We regarded three length of rotor diameter, namely 80, 82 and 90 m. The distribution along a line took into account the rotor diameter as well. Furthermore, the coordinates of each point induced by its assigned line was slightly randomized again as well.

We applied two different distribution strategies to locate the substation. In the first one the substation can be randomly positioned. In the second one it can only be placed somewhere at the border.

For each number of turbines we created 10 instances for the random distribution and the three array distributions, leading to 120 different turbine settings. Combined with five sinks positioned at random and five placed near the border we got a total of 1200 different wind farm points instances.

Furthermore, we created two sets of cable types, namely one which follows economies of scales as defined for SSBB problems and one which does not. The used cable types were based on realistic examples provided by a cable supplier. From their specifications we derived their capacity in terms of number of turbines. In order to obtain cable sets following economies of scale we had to adapt the prices accordingly. In the end, we created cable type sets of size three and five for each cable type property.

4.2. Results of experiments

We implemented the algorithms with C++. Thereby, we made use of parts of the boost libraries[1] in version 1.55.0. For the implementation of the MILP and ILP formulations we used the Gurobi Optimizer[2] and its C++ interface in version 5.6.2. The implementation was compiled with gcc in version 4.7.1.

The experiments were run on a computer with 4 12-Core AMD Opteron(tm) Processor 6172, 2.1 GHz with 256 GB RAM with openSUSE 12.2 (Mantis) (x86_64) as the operating system. We restricted each experiment to one core and one thread. By using the openSUSE command *ulimit* we restricted the virtual memory for each experiment to 16 GB. For the parallel execution of the experiments on the different cores we used the program *GNU Parallel* of Tange [36].

We describe the results of the experiments in the following. We analyze them regarding their quality and running time.

We measure the quality in terms of the amount of feasible solutions found, the amount of lower bounds provided and the gap of the solution to the best found lower bound. Thereby, the best found lower bound is the optimal solution calculated by the Two-Path MILP or the Path ILP if an optimal solution was found. If they could not provide any then the better lower bound of these formulations is taken. If the exact solutions did not find any lower bound or optimal solution the smallest feasible solution value of all approaches is taken as the reference value for the gap calculation. The gap is expressed in percent and calculated by $(\frac{SOL}{OPT} - 1) * 100$ with SOL being the value of the found solution and OPT the lower bound. Since not every solution approach regarded necessarily provides a feasible solution we only consider feasible solutions when calculating benchmarks based on the gap.

Furthermore, we consider the running time of every algorithm. Here, we do not restrict ourselves to runs which produced a feasible solution. We measure the running time in CPU seconds. For the ILP and MILP solutions we additionally check whether the algorithms found an optimal solution within the set time limit of 30 minutes and within the set memory restrictions.

In Subsection 4.2.1 we present the results for the ILP and MILP approaches and in Subsection 4.2.2 the performance of the heuristics is discussed.

4.2.1. ILP and MILP approaches

We ran the MILP adapted from Hertz et al. [20], our ILP and the ILP formulation of Uchoa et al. [37] on the random problem instances with 25 turbines and the sets with three and five cable types. The results on the three cable types instances are summarized in Table 4.1 and Table 4.2.

[1]http://www.boost.org/
[2]http://www.gurobi.com/

	Array distribution									
	Sink at border									
Algorithms	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	150	0.00	0.00	1300.00	150	150	3.45	0.00	1300.00	150
Path ILP	150	675.56	297.08	1119.60	0	150	703.26	328.12	1090.87	0
Flow ILP	140	0.00	0.00	76.61	89	140	2.62	0.00	61.62	63
	Sink uniform									
Algorithms	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	150	0.00	0.00	1300.00	150	150	2.38	0.00	1300.00	150
Path ILP	150	610.80	279.66	1166.97	0	150	640.70	301.17	1086.33	0
Flow ILP	141	0.00	0.00	89.84	98	142	2.48	0.00	87.95	63
	Random distribution									
	Sink at border									
Algorithms	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	50	0.00	0.00	1300.00	50	50	3.28	0.00	1300.00	50
Path ILP	50	379.33	322.65	1112.54	0	50	418.19	336.76	1089.00	0
Flow ILP	50	0.00	0.00	158.89	27	50	3.28	0.00	165.37	20
	Sink uniform									
Algorithms	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	50	0.00	0.00	1300.00	50	50	0.00	0.00	1300.00	50
Path ILP	50	343.87	233.54	1119.40	0	50	364.82	246.72	1086.19	0
Flow ILP	50	0.00	0.00	168.68	29	50	0.00	0.00	146.91	28

Table 4.1.: **Quality benchmarks for the ILP and MILP formulations on scenarios with 25 turbines and the three cable types**

	Array distribution									
Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1429.05	480.83	1853.85	90	60	1810.72	562.61	1904.92	63	87
Path ILP	1829.31	1826.45	1880.25	0	150	1834.26	1826.10	1880.77	0	150
Flow ILP	145.20	1.79	1687.16	150	0	229.51	5.72	1767.45	150	0
Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1374.71	517.43	1950.29	100	50	1810.47	480.99	1869.38	69	81
Path ILP	1828.59	1826.59	1881.46	0	150	1828.06	1826.51	1880.62	0	150
Flow ILP	95.70	2.20	1793.28	149	1	137.22	2.06	1793.85	145	5
	Random distribution									
Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1685.97	824.58	1840.82	28	22	1810.80	831.40	1928.39	20	30
Path ILP	1831.35	1826.71	1879.06	0	50	1832.18	1826.70	1879.69	0	50
Flow ILP	119.30	26.28	674.70	50	0	137.50	23.32	711.51	50	0
Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1712.84	900.11	1848.18	29	21	1659.29	703.94	1844.11	28	22
Path ILP	1828.82	1826.75	1879.86	0	50	1829.25	1826.42	1879.76	0	50
Flow ILP	127.67	30.51	638.87	50	0	108.94	12.99	935.94	50	0

Table 4.2.: **Running time benchmarks for the ILP and MILP formulations on scenarios with 25 turbines and the three cable types**

In Table 4.1 we subsume the number of feasible solutions, the median gap, the minimum gap, the maximum gap and the number of instances for which the found solution is equal to the cost-minimal solution or best lower bound found. The upper part shows the results on instances with an array distribution and the second one on instances with randomly placed turbines. The table splits the results again depending on the sink position and the cable type properties. Table 4.2 presents the running time of the algorithms and is partitioned in the same manner.

The benchmarks provided in these tables show the hardness of the problem of the SCIP. None of the formulation could be solved optimally. The Path ILP formulation could never find any optimal solution. For each problem instance setting the calculation exceeds the time limit of 30 minutes. The adaption of the MILP of Hertz et al. [20] performs better. Though, it cannot find the optimal solution to every problem instance within the given time frame either. In fact, 46.6 % remain unsolved. Even the less complex formulation by Uchoa et al. [37] cannot solve all instances. Six of the instances with a uniform turbine and sink positioning needed a calculation time exceeding the time limit. Hence, for this approach instances with randomly positioned points seem to be harder to solve.

Comparing the uniform and wind farm oriented turbine placement setting for the MILP we cannot distinguish if one is harder to solve. Looking at the percentage of unsolved instances we get the following values for the the different settings on scenarios with a array distribution: 40%, 58%, 30% and 54%. In the uniform turbine placement the corresponding ratios are 44%, 60%, 42% and 44%, respectively. Hence, in the wind farm and uniform turbine placement the percentage of unsolved problem instances is quite similar for a sink position at the border. If the sink is placed randomly in the plane the percentage of found solutions differ more but no setting is harder to solve in both cable settings.

Apparently, cable types with no economies of scale lead to harder problem instances since the percentage of unsolved instances is greater than their counterparts settings with economies of scale. Positioning the sink at the border does not lead to less optimal solutions for all settings. Hence, this does not seem to have a big impact on the hardness of the problem.

Regarding the gap, the Flow ILP finds many and good feasible solutions on the two given cable sets. The median gap is similar to the one of the MILP for all settings. In addition, if the MILP finds an optimal and the ILP of Uchoa et al. [37] a feasible solution their values are the same.

Nevertheless, we can observe that the maximum gap between a lower bound and a feasible solution found can be huge. For the Flow ILP the maximum gap differs between 61.62 % and 168.68 % while for the Path ILP and the Two-Path MILP the maximum gap can be up to 1166 % and 1300 %, respectively. This is an indicator of the slow convergence of the lower bound and feasible solution during the ILP and MILP solving.

The running time of the Flow ILP outperforms the other two formulations significantly. The median of all running times is always a fraction of the one of the Two-Path MILP. Even the lowest running time of the Two-Path MILP is always much higher than the median of the Flow ILP.

Running these approaches on problem instances with five cable types let to even less optimal solutions and lower bounds. Both the Two-Path MILP and the Path ILP could not solve any problem instance optimally. They ran out of time on all instances. Furthermore, the Path ILP did not find any lower bound either. The Two-Path MILP, however, still found lower bounds for 772 instances. On these problem instances the ILP of Uchoa et al. [37] could calculate feasible solutions for all instances. Its running times were even better than for the sets with three cable types. However, this is still reasonable since the main driving factors for the complexity of the Flow ILP are the number of turbines since they also determine the number of "cable types", i.e. capacities, which are regarded by the model. In this case there was no increase in complexity for the Flow ILP.

Since the number of optimally solved instances is zero the median gap and minimum gap for Two-Path MILP increase. More interesting is the fact that the maximum gap rises to high values.

From the results in this subsection we observe that it is hard to find an optimal solution even on small instances with 25 turbines. For even bigger problem instances the Path ILP and Two-Path MILP formulations are not reasonable anymore. Hence, we need other means to calculate good solutions faster. One way are the previously described heuristics whose performance we discuss in the next subsection.

4.2.2. Heuristics

In experiments for the heuristics we used the same instances we already regarded in the previous subsection and also included some bigger instances. We ran each heuristic on every instance. One special case is RandomizedAggregationHeuristic. Since it is randomized we ran it 10 times on each instance in order to get enough data to calculate the expected benchmarks.

As for the ILP and MILP formulations we ran the heuristics on the instances with 25 turbines and three cable types. Let us regard these results first. Table 4.3 shows the quality benchmarks for the array distribution and Table 4.4 for the random distribution. For the RandomizedAggregationHeuristic the number of feasible solutions and the number of lower bounds found are always expected values. The calculation of the median, minimum and maximum gap and running time is always done over all instances and all runs. Apart from that the tables provide the same information and have the same structure as in Subsection 4.2.1.

Each calculation of the heuristics take only a few milliseconds no matter which instances setting. Since the running times are so small we leave the detailed discussion about them for bigger instances where the running times are higher and better comparable and omit a presentation in a table at this point.

Regarding the quality on these instances the heuristics differ more than in the running time. The MstHeuristic finds the least number of feasible instances. However, the MstHeuristic is the only heuristic apart from the RandomizedAggregationHeuristic which found solutions with a gap of 0%. Additionally, the feasible solutions found by the MstHeuristic approximate the lower bound better than any other heuristic. This can be seen by the median and maximum gap which are lower than the ones of the other heuristics for all problem instance settings.

The heuristics of Pappas et al. [29] solve more instances than the MstHeuristic even if they use the same cable mapping. In fact, UpgradeHeuristic1 and UpgradeHeuristic2 give feasible solutions for all instances. Only UpgradeHeuristic3 does not provide a feasible solution for each instance. However, the feasible solutions of UpgradeHeuristic3 provide the better gaps regarding median, minimum and maximum gap. In fact, it provides the second best median gaps after the MstHeuristic together with the RandomizedAggregationHeuristic. UpgradeHeuristic1 on the other side performs the worst regarding median, minimum and maximum gap in all problem settings.

The UnsplittableUpgradeHeuristic also provides only feasible solutions. Though, ensuring the feasibility of the solution at all time during the algorithm does not result in better gaps. UpgradeHeuristic3, from which UnsplittableUpgradeHeuristic is derived, produces median gaps which are nearly twice as good for the array distribution. In the random distribution, however, the UnsplittableUpgradeHeuristic performs better and its median gaps are closer to the one of UpgradeHeuristic3.

The RandomizedAggregationHeuristic provides the second best expected minimum gaps among the heuristics. Despite its randomized algorithm which aggregates the flow at random nodes it provides among the second best median gaps. The expected percentage of feasible solution is between 95.2 % and 98.2 %. The maximum gaps, however, are big as well. Only the maximum gaps of the UpgradeHeuristic1 are bigger.

Remarkable in Table 4.3 and Table 4.4 are the high maximum gaps. Since these are in relation to the lower bound found by the Two-Path MILP and were even big for the approaches from Subsection 4.2.1 we provide in Table A.5 and Table A.6 gaps based on optimal solutions found by the Two-Path MILP only. In these tables we can observe that the maximum gap drops significantly when neglecting the lower bounds of the Two-Path MILP.

For the instances with 25 turbines and a cable type set with 5 different types the observation made so far still apply.

For the bigger instances we have no more lower bounds from any ILP or MILP formulation and therefore the minimum value of all feasible solution of any heuristic is used as the reference value for the gap. For the instances with 50 turbines we can observe that the number of found feasible solutions decreases. The MstHeuristic does not provide any feasible solution anymore. The three heuristics of Pappas et al. [29] also compute less feasible solution. Especially, for UpgradeHeuristic3 only a third or even less of the calculated solutions are feasible. However, the median gap is still better than the ones of UpgradeHeuristic2 and UpgradeHeuristic3. Now UnsplittableUpgradeHeuristic is the only heuristic which sill finds a feasible solution for every instance. However, its maximum gap and, hence, the gaps to feasible solutions found by other heuristics is still quite large.

On the bigger instances we can now also see how much time each heuristic needs to calculate a solution. As an example the running times for instances with 100 turbines are given in the appendix. All heuristics are fast even on instances with 100 turbines. The MstHeuristic is fast even on instances with 100 turbines. On average it needs less than 0.01 seconds. The heuristics of Pappas et al. [29] take more time the more complex they are. Nevertheless, within 7 seconds a solution for an instance was calculated Hence, UpgradeHeuristic1 needs the fewest time and UpgradeHeuristic3 the most. The UnsplittableUpgradeHeuristic is also faster than UpgradeHeuristic3 and even UpgradeHeuristic2. The most time consuming heuristic is the RandomizedAggregationHeuristic. Especially on instances with cables which do not obey economies of scale the running time is comparably high with a median runtime between 12.27 and 13.07 seconds.

In conclusion, we observed that on our benchmark instances the heuristics solve instances with up to 100 turbines within a few seconds. Thereby, only the UnsplittableUpgradeHeuristic provides feasible solutions reliably. However, if the MstHeuristic, UpgradeHeuristic3 or RandomizedAggregationHeuristic find a feasible solution its gap is mostly better. Though, especially the MstHeuristic find feasible solutions very rarely, especially on bigger instances. Also UpgradeHeuristic2 outperforms UnsplittableUpgradeHeuristic on some instances but is worse in others. The worst solutions are given by UpgradeHeuristic1. Even if most of its solutions are feasible and it is one of the fastest heuristics the application of UpgradeHeuristic1 to the SCIP is not practical since much better solutions in not much more time can be computed by

UnsplittableUpgradeHeuristic for example.

Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	20	5.82	0.00	100.63	2	20	6.62	0.00	18.85	2
UpgradeHeuristic1	150	103.39	43.64	251.66	0	148	107.49	52.11	239.44	0
UpgradeHeuristic2	150	38.67	16.97	145.48	0	148	37.28	20.76	132.26	0
UpgradeHeuristic3	130	24.29	4.13	130.72	0	130	24.15	4.20	109.69	0
UnsplittableUp-gradeHeuristic	150	41.40	15.43	139.89	0	150	41.05	17.46	132.14	0
RandomizedAggre-gationHeuristic	145.3	30.34	3.69	184.61	0.00	147.2	26.57	1.34	186.49	0.00

Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	40	5.54	0.00	65.90	2	40	7.98	0.00	120.36	2
UpgradeHeuristic1	150	105.69	43.71	396.58	0	149	120.60	51.45	313.89	0
UpgradeHeuristic2	149	36.99	15.68	184.21	0	149	37.70	17.07	156.15	0
UpgradeHeuristic3	137	23.18	1.86	135.77	0	140	25.05	2.36	152.78	0
UnsplittableUp-gradeHeuristic	150	40.05	6.09	181.05	0	150	41.53	13.72	147.05	0
RandomizedAggre-gationHeuristic	146.8	25.30	0.00	177.56	0.10	146.4	25.15	0.42	171.17	0.00

Table 4.3.: **Quality benchmarks for the heuristics on the scenario with 25 turbines, array distribution and three cable types**

Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	7	12.19	0.00	203.56	1	7	17.95	2.25	182.20	0
UpgradeHeuristic1	50	94.75	71.75	374.50	0	50	106.46	67.22	385.18	0
UpgradeHeuristic2	50	29.28	18.01	233.76	0	50	31.43	17.38	226.62	0
UpgradeHeuristic3	42	20.12	5.45	215.21	0	41	23.76	6.40	190.95	0
UnsplittableUp-gradeHeuristic	50	26.55	15.29	344.37	0	50	29.95	19.55	216.63	0
RandomizedAggre-gationHeuristic	48.4	26.70	2.11	296.43	0.00	48.1	27.97	4.02	304.36	0.00

Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	12	7.14	0.00	208.26	1	12	8.11	0.00	33.88	1
UpgradeHeuristic1	50	92.69	60.45	420.47	0	50	99.15	67.01	390.03	0
UpgradeHeuristic2	50	24.11	11.74	245.33	0	50	24.03	10.79	220.96	0
UpgradeHeuristic3	42	15.04	4.30	235.15	0	40	17.03	5.54	42.52	0
UnsplittableUp-gradeHeuristic	50	24.96	11.77	224.64	0	50	25.78	13.40	214.87	0
RandomizedAggre-gationHeuristic	49	22.76	1.11	313.26	0.00	49.4	23.78	1.47	296.93	0.00

Table 4.4.: **Quality benchmarks for the heuristics on the scenario with 25 turbines, random distribution and the three cable types**

5. Conclusion

In the context of wind energy and wind farms interesting and complex problems exists. In this thesis we have regarded one of them, namely the SCIP, which is a subproblem of the WFCIP. Thereby, we dealt the SCIP by identifying suitable algorithms for the problem and evaluating them in an experimental study.

Therefore, we have identified approaches which can be adapted to the SCIP during a literature survey. We then implemented nine different algorithms to solve the SCIP. Thereby, we regarded two exact MILP resp. ILP formulations, one suboptimal ILP and six heuristics.

The presented exact MILP formulation TWO-PATH MILP is an adaption of the work of HERTZ ET AL. [20] and models the unsplittability constraint of the SCIP with the help of two-edge paths. This means that for every node all possible incoming and outgoing cable combinations are regarded and constrained such that the flow is unsplittable.

In addition, we developed our own ILP formulation called PATH ILP. Here the unsplittability is modeled by restricting each output flow of a turbine ot one distinct source-sink path which cannot be split.

The last ILP formulation is taken from UCHOA ET AL. [37]. In this thesis we have called it FLOW ILP. This ILP is a simple formulation for the MLCMST which works on unitary capacities installing exactly that capacity which is needed by the flow along one edge. Hence, the formulation only uses integer variables indicating the amount of flow on an edge and does not need any further variables for the cable installation itself. In order to apply the FLOW ILP to the SCIP we calculated the cost-minimal cable combinations for every possible flow in the network and hence had an unitary flow-capacity mapping. However, the found solution by FLOW ILP is not necessarily feasible since it depends on the cable type combinations which are consecutive along a path to the sink.

In our experiments we tested these approaches on small instances with 25 turbines and cable type sets of size 3 and 5. None of the formulations could solve all those instances optimally in the given time limit of 30 minutes. While the TWO-PATH MILP still found optimal solutions for the instances with 3 cable types PATH ILP did not find any optimal solution at all. The FLOW ILP could not find its optimal solution only in a few cases. However, some of its solutions were not feasible for the SCIP.

These results show the hardness of the SCIP and demonstrate that there is a need for heuristics in order to calculate good results in reasonable time. Hence, we looked at heuristic approaches as well. We implemented and evaluated the following six approaches.

The first heuristic, MSTHEURISTIC, is a simple approach which calculates the MST on the given input and then installs the optimal cable combination on each edge such that all turbines' weight can be sent to the substation.

Additionally, we adapted an approximation algorithm for the SSBB to the SCIP. This algorithm iteratively aggregates the turbines' weight at a random subset of nodes and installs the necessary cables to support the aggregation and hence was called RANDOMIZEDAGGREGATIONHEURISTIC.

Three further heuristics we implemented are presented by PAPPAS ET AL. [29]. They are greedy heuristics which try to minimize the cost of current solution by iteratively implementing a most profitable capacity upgrade. The heuristics differ in the way they calculate the most profitable upgrade. At each edge only one cable can be installed. Hence, we adapted the input for these heuristics such that they work on the optimal cable combination for all possible flow values.

From one of these heuristics we derived the UnsplittableUpgradeHeuristic. As well as the heuristics of Pappas et al. [29], it iteratively looks for the most profitable capacity upgrade. Though, it only allows upgrades which are in accordance with the unsplittability restriction of the SCIP.

In our experiments these heuristics only needed some seconds to solve SCIP instances with up to 100 turbines. Hence, they outreached the running times of the MILP and ILP approaches by far.

It is notable that the MstHeuristic finds by far the least feasible solutions of all heuristics but, therefore, the found feasible solutions have the lowest gaps to the lower bounds. The UnsplittableUpgradeHeuristic always finds a feasible solution but the feasible solutions of UpgradeHeuristic3 or RandomizedAggregationHeuristic provide lower gaps.

In conclusion, in this thesis we dealt with the SCIP for multiple cables. Before it has been only regarded in the single-cable version by Berzan et al. [2] and the two-cable version by Hertz et al. [20]. Our experiments show that exact approaches fail to provide optimal solutions for even relatively small instances in reasonable time. Therefore, approximate results via different ILP formulations or heuristics are necessary.

For future work it would be interesting to find out if there are any cutting planes for the exact approaches which can speed up the calculation. Additionally, the improvement of the heuristics is appealing. The comparison of UpgradeHeuristic3 and Unsplittable-UpgradeHeuristic shows that the UnsplittableUpgradeHeuristic misses better feasible solutions. Instead of the first fit procedure, other procedures for the detection of unsplittability conformity for a node upgrade might improve the solutions.

Furthermore, finding suitable approaches for the WFCIP is still an open issue in the multi-cable version. One way can be to first allocate the turbines to one substation and then solve each SCIP with one of the approaches presented here.

A. Further benchmarks of the experiments

A.1. ILP and MILP formulations

	Array distribution									
Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	150	63.49	15.75	84340.91	143	150	46.27	3.45	92079.77	147
Path ILP	150	1084.21	494.82	1124.66	0	150	1048.11	566.10	1093.91	0
Flow ILP	140	48.53	0.00	168.48	7	127	39.40	0.00	144.68	3
Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	150	74.76	3.60	58305.14	148	150	60.16	1.59	161415.79	144
Path ILP	150	1080.20	552.05	1130.19	0	150	1042.95	494.53	1103.37	0
Flow ILP	134	61.80	0.00	182.37	2	122	51.08	0.00	157.73	4
	Random distribution									
Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	50	148.68	109.66	74117.22	48	50	125.23	86.20	103314.52	48
Path ILP	50	1082.30	387.38	1120.38	0	50	1048.21	426.99	1093.79	0
Flow ILP	50	131.68	0.00	165.87	2	49	115.27	0.00	141.86	2
Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
Two-Path MILP	50	179.90	119.63	145345.43	44	50	150.87	7.10	113332.83	50
Path ILP	50	1080.45	271.91	1111.63	0	50	1048.41	523.63	1084.43	0
Flow ILP	50	145.81	0.00	227.79	6	49	128.98	7.10	202.75	0

Table A.1.: **Quality benchmarks for the ILP and MILP formulations on scenarios with 25 turbines and five cable types**

	Array distribution									
Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1849.76	1845.73	1937.59	0	150	1852.95	1845.76	1932.23	0	150
Path ILP	1856.72	1852.91	1948.03	0	150	1856.43	1852.80	1949.00	0	150
Flow ILP	78.93	1.64	907.92	150	0	85.08	1.60	961.99	150	0
Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1853.08	1845.71	1930.07	0	150	1851.91	1845.41	1933.27	0	150
Path ILP	1858.61	1852.83	1950.36	0	150	1857.24	1853.06	1948.82	0	150
Flow ILP	63.31	1.43	1219.96	150	0	46.78	1.84	1471.34	150	0
	Random distribution									
Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1855.08	1845.28	1935.05	0	50	1868.91	1846.29	1933.26	0	50
Path ILP	1868.13	1853.63	1949.59	0	50	1856.12	1853.19	1948.44	0	50
Flow ILP	77.00	16.10	434.48	50	0	103.84	18.14	537.11	50	0
Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Median	Minimum	Maximum	Finished	Time exceeded	Median	Minimum	Maximum	Finished	Time exceeded
Two-Path MILP	1849.56	1845.71	1928.00	0	50	1850.39	1845.42	1921.62	0	50
Path ILP	1856.07	1853.65	1947.91	0	50	1856.07	1853.78	1939.29	0	50
Flow ILP	71.89	8.57	964.17	50	0	82.50	7.40	533.39	50	0

Table A.2.: **Running time benchmarks for the ILP and MILP formulations on scenarios with 25 turbines and the five cable types**

A.2. Heuristics

Algorithms	Sink at border					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	< 0.01
UpgradeHeuristic1	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic2	0.02	< 0.01	0.03	0.02	< 0.01	0.03
UpgradeHeuristic3	0.05	0.01	0.08	0.05	0.01	0.08
UnsplittableUp-gradeHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
RandomizedAggre-gationHeuristic	0.02	0.01	0.06	0.02	0.01	0.07

Algorithms	Sink uniform					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
UpgradeHeuristic1	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic2	0.02	< 0.01	0.02	0.02	< 0.01	0.02
UpgradeHeuristic3	0.04	0.01	0.07	0.04	0.01	0.07
UnsplittableUp-gradeHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
RandomizedAggre-gationHeuristic	0.05	0.01	0.06	0.04	0.01	0.06

Table A.3.: **Running time benchmarks for the heuritics on scenarios with 25 turbines, array distribution and three cable types**

Algorithms	Sink at border					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
UpgradeHeuristic1	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic2	0.02	< 0.01	0.02	0.02	< 0.01	0.02
UpgradeHeuristic3	0.04	0.01	0.06	0.04	0.01	0.05
UnsplittableUp-gradeHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
RandomizedAggre-gationHeuristic	0.03	0.01	0.06	0.02	0.01	0.06

Algorithms	Sink uniform					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	< 0.01
UpgradeHeuristic1	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic2	0.01	< 0.01	0.02	0.01	< 0.01	0.02
UpgradeHeuristic3	0.03	0.01	0.05	0.03	0.01	0.05
UnsplittableUp-gradeHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
RandomizedAggre-gationHeuristic	0.02	0.01	0.06	0.02	0.01	0.07

Table A.4.: **Running time benchmarks for the heuristics on scenarios with 25 turbines, random distribution and three cable types**

Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	16	4.72	0.00	15.45	2	13	4.57	0.00	14.20	2
UpgradeHeuristic1	90	98.66	55.99	189.49	0	63	106.85	62.16	198.12	0
UpgradeHeuristic2	90	37.12	17.90	66.17	0	63	35.72	20.76	60.40	0
UpgradeHeuristic3	81	22.22	4.13	47.23	0	59	22.88	4.20	45.87	0
UnsplittableUpgradeHeuristic	90	38.85	18.43	79.40	0	63	39.05	21.05	69.91	0
RandomizedAggregationHeuristic	87.1	25.40	4.47	123.38	0.00	61.7	21.82	1.34	83.17	0.00

Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	32	4.23	0.00	20.74	2	25	4.84	0.00	21.46	2
UpgradeHeuristic1	100	101.39	46.61	196.41	0	69	117.09	60.96	213.58	0
UpgradeHeuristic2	99	34.99	15.68	73.75	0	69	36.07	17.07	69.09	0
UpgradeHeuristic3	95	19.30	1.86	45.09	0	63	21.57	2.36	41.72	0
UnsplittableUpgradeHeuristic	100	37.68	6.09	88.18	0	69	39.27	13.72	86.19	0
RandomizedAggregationHeuristic	98.1	22.65	0.00	83.64	0.10	67.9	19.10	0.42	90.76	0.00

Table A.5.: **Quality benchmarks for the heuristics on scenarios with 25 turbines, array distribution and three cable types:** In this table we compare the results of the heuristics with only optimal solutions found by Two-Path MILP.

Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	3	0.37	0.00	1.44	1	2	10.10	2.25	17.95	0
UpgradeHeuristic1	28	91.92	71.75	114.97	0	20	99.47	67.22	117.01	0
UpgradeHeuristic2	28	26.97	18.01	38.57	0	20	28.30	17.38	40.70	0
UpgradeHeuristic3	26	17.70	5.45	29.13	0	16	20.57	6.40	33.45	0
UnsplittableUpgradeHeuristic	28	25.40	15.29	52.76	0	20	26.21	21.14	53.51	0
RandomizedAggregationHeuristic	27.2	24.64	2.11	59.51	0.00	19.2	23.16	4.02	61.25	0.00

Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	7	3.64	0.00	7.60	1	9	6.90	0.00	9.35	1
UpgradeHeuristic1	29	81.83	60.45	103.66	0	28	90.13	75.70	113.82	0
UpgradeHeuristic2	29	20.82	11.74	42.55	0	28	21.53	10.79	43.19	0
UpgradeHeuristic3	25	12.18	4.30	39.41	0	24	15.11	5.54	40.39	0
UnsplittableUpgradeHeuristic	29	19.55	11.77	71.97	0	28	19.80	13.40	71.26	0
RandomizedAggregationHeuristic	28.8	18.28	1.11	60.73	0.00	27.8	18.91	1.47	96.28	0.00

Table A.6.: **Quality benchmarks for the heuristics on scenarios with 25 turbines, random distribution and three cable types:** In this table we compare the results of the heuristics with only optimal solutions found by the Two-Path MILP.

Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	0	–	–	–	0	0	–	–	–	0
UpgradeHeuristic1	136	106.11	55.18	170.89	0	127	115.25	59.92	186.04	0
UpgradeHeuristic2	132	13.06	2.21	32.75	0	135	16.90	3.65	31.40	0
UpgradeHeuristic3	28	5.68	0.00	16.60	5	28	8.95	3.85	20.33	0
UnsplittableUpgradeHeuristic	150	6.55	0.00	63.81	26	150	19.44	0.00	80.85	1
RandomizedAggregationHeuristic	94.5	7.04	0.00	37.73	11.90	112.8	6.19	0.00	63.81	14.90

Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	0	–	–	–	0	0	–	–	–	0
UpgradeHeuristic1	142	123.41	70.10	203.32	0	139	131.01	75.77	189.36	0
UpgradeHeuristic2	139	12.39	0.12	28.63	0	142	16.30	0.00	33.29	1
UpgradeHeuristic3	52	3.77	0.00	16.96	14	51	8.29	0.00	21.87	4
UnsplittableUpgradeHeuristic	150	9.28	0.00	78.95	22	150	15.94	0.00	83.29	2
RandomizedAggregationHeuristic	108.2	7.62	0.00	46.23	11.90	120	6.86	0.00	45.74	14.80

Table A.7.: **Quality benchmarks for the heuristics on scenarios with 50 turbines, array distribution and five cable types**

Algorithms	Sink at border									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	0	–	–	–	0	0	–	–	–	0
UpgradeHeuristic1	50	102.04	74.32	122.60	0	49	110.21	90.32	134.62	0
UpgradeHeuristic2	46	10.79	4.05	22.37	0	49	14.06	3.35	26.10	0
UpgradeHeuristic3	12	2.23	0.00	18.16	4	7	3.08	0.00	11.41	1
UnsplittableUp-gradeHeuristic	50	3.09	0.00	31.22	14	50	7.69	0.00	44.99	5
RandomizedAggre-gationHeuristic	34.1	8.40	0.00	35.03	3.20	38	6.00	0.00	34.08	4.40

Algorithms	Sink uniform									
	Economies of scale					No economies of scale				
	Feasible	Median	Minimum	Maximum	LB	Feasible	Median	Minimum	Maximum	LB
MstHeuristic	0	–	–	–	0	0	–	–	–	0
UpgradeHeuristic1	50	109.12	80.84	128.58	0	50	117.16	94.26	142.42	0
UpgradeHeuristic2	48	8.48	1.18	14.57	0	46	10.76	3.74	22.95	0
UpgradeHeuristic3	21	1.40	0.00	6.90	8	20	4.14	0.00	17.16	5
UnsplittableUp-gradeHeuristic	50	2.36	0.00	16.39	14	50	5.83	0.00	16.13	4
RandomizedAggre-gationHeuristic	39.9	7.37	0.00	37.55	2.80	39.5	7.12	0.00	32.04	4.60

Table A.8.: **Quality benchmarks for the heuristics on scenarios with 50 turbines, random distribution and five cable types**

Algorithms	Sink at border					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic1	0.05	0.03	0.06	0.05	0.03	0.07
UpgradeHeuristic2	0.61	0.52	0.76	0.62	0.52	0.73
UpgradeHeuristic3	2.08	1.45	2.76	2.09	1.42	2.96
UnsplittableUpgradeHeuristic	0.22	0.07	0.30	0.22	0.08	0.31
RandomizedAggregationHeuristic	3.00	2.76	7.53	12.20	11.17	28.26

Algorithms	Sink uniform					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic1	0.04	0.03	0.06	0.05	0.03	0.13
UpgradeHeuristic2	0.55	0.45	0.76	0.57	0.47	1.73
UpgradeHeuristic3	1.73	1.35	2.75	1.79	1.32	6.32
UnsplittableUpgradeHeuristic	0.18	0.08	0.27	0.18	0.07	0.28
RandomizedAggregationHeuristic	3.01	2.74	7.41	12.19	11.23	28.01

Table A.9.: **Running time benchmarks of the heuristics on scenarios with 100 turbines, array distribution and five cable types**

Algorithms	Sink at border					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic1	0.05	0.04	0.13	0.05	0.04	0.14
UpgradeHeuristic2	0.56	0.51	1.63	0.57	0.51	1.59
UpgradeHeuristic3	1.83	1.50	5.26	1.90	1.58	4.79
UnsplittableUp-gradeHeuristic	0.19	0.09	0.23	0.19	0.09	0.24
RandomizedAggre-gationHeuristic	3.11	2.78	7.52	12.20	11.24	27.82

Algorithms	Sink uniform					
	Economies of scale			No economies of scale		
	Median	Minimum	Maximum	Median	Minimum	Maximum
MstHeuristic	< 0.01	< 0.01	0.01	< 0.01	< 0.01	0.01
UpgradeHeuristic1	0.05	0.04	0.13	0.05	0.04	0.13
UpgradeHeuristic2	0.49	0.43	1.44	0.49	0.44	1.44
UpgradeHeuristic3	1.47	1.26	4.81	1.45	1.27	4.81
UnsplittableUp-gradeHeuristic	0.15	0.10	0.22	0.15	0.12	0.24
RandomizedAggre-gationHeuristic	3.07	2.75	7.62	12.23	11.21	27.35

Table A.10.: **Running time benchmarks of the heuristics on scenarios with 100 turbines, random distribution and five cable types**

List of Tables

List of Algorithms

Bibliography

[1] D. Berger, B. Gendron, J.-Y. Potvin, S. Raghavan, and P. Soriano. Tabu Search for a Network Loading Problem with Multiple Facilities. *Journal of Heuristics*, 6:253–267, 2000.

[2] C. Berzan, K. Veeramachaneni, J. McDermott, and U.-M. O'Reilly. Algorithms for cable network design on large-scale wind farms. *Massachusetts Institute of Technology*, pages 1–24, 2011.

[3] Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. Hintergrundinformationen zum Ausbau der Erneuerbaren Energien in Deutschland bis 2020, 2011. URL `http://www.erneuerbare-energien.de/fileadmin/ee-import/files/pdfs/allgemein/application/pdf/hintergrund\_ausbau\_ee\_bf.pdf`. [Online; accessed 22/06/2014].

[4] T. Burton, N. Jenkins, D. Sharpe, and E. Bossanyi. *Wind Energy Handbook*. John Wiley & Sons Ltd., Chichester, second edition, 2011.

[5] E.H. Camm, M.R. Behnke, O. Bolado, M. Bollen, M. Bradt, C. Brooks, W. Dilling, M. Edds, W. J. Hejdak, D. Houseman, S. Klein, F. Li, J. Li, P. Maibach, T. Nicolai, J. Patino, S.V. Pasupulati, N. Samaan, S. Saylors, T. Siebert, T. Smith, M. Starke, and R. Walling. Wind power plant collector system design considerations: IEEE PES wind plant collector system design working group. In *Power Energy Society General Meeting, 2009. PES '09. IEEE*, pages 1–7, July 2009.

[6] C. Chekuri, Sanjeev K., and Joseph Seffi Naor. A Deterministic Algorithm for the COST- DISTANCE Problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 232–233, 2001.

[7] European Commission and Directorate-General for Communication. Europe 2020 – Europe's growth strategy: growing to a sustainable and job-rich future, 2012. [Online; accessed 22/06/2014].

[8] A. Czumaj, J. Czyzowicz, L. Gąsieniec, J. Jansson, A. Lingas, and P. Zylinski. Approximation Algorithms for Buy-At-Bulk Geometric Network Design. *International Journal of Foundations of Computer Science*, 22(8):1949–1969, 2011.

[9] S. Dutta. *Data Mining and Graph Theory Focused Solutions to Smart Grid Challenges*. PhD thesis, University of Illinois, 2012.

[10] P. Fagerfjäll. *Optimizing wind farm layout: more bang for the buck using mixed integer linear programming*. PhD thesis, Chalmers University of Technology and Gothenburg University, 2010.

[11] I. Gamvros, S. Raghavan, and B. Golden. An Evolutionary Approach to the Multi-Level Capacitated Minimum Spanning Tree Problem. In G. Anandalingam and S. Raghavan, editors, *Telecommunications Network Design and Management*, pages 99–124. Springer Science+Business Media, New York, reprinted edition, 2003.

[12] I. Gamvros, B. Golden, and S. Raghavan. The Multilevel Capacitated Minimum Spanning Tree Problem. *INFORMS Journal on Computing*, 18(3):348–365, 2006.

[13] N. Garg, R. Khandekar, G. Konjevod, F. S. Salman, and A. Sinha. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem. In K. Aradal and B. Gerards, editors, *Proceeding of the 8th International Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science 2081, pages 170–184, Berlin Heidelberg, 2001. Springer-Verlag.

[14] B. Gendron, J.-Y. Potvin, and P. Soriano. Diversification strategies in local search for a nonbifurcated network loading problem. *European Journal of Operational Research*, 142(2):231–241, 2002.

[15] F. Grandoni and G.F. Italiano. Improved approximation for single-sink buy-at-bulk. In T. Asano, editor, *Algorithms and Computation, 17th International Symposium, ISAAC 2006*, Lecture Notes in Computer Science 4288, pages 111–120, Berlin Heidelberg, 2006. Springer-Verlag.

[16] F. Grandoni, T. Rothvoss, and L. Sanita. From Uncertainty to Nonlinearity: Solving Virtual Private Network via Single-Sink Buy-at-Bulk. *Mathematics of Operations Research*, 36(2):185–204, 2011.

[17] S. Guha, A. Meyerson, and K. Munagala. A Constant Factor Approximation for the Single Sink Edge Installation Problem. *SIAM Journal on Computing*, 38(6):2426–2442, 2009.

[18] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th ACM Symposium on Theory of computing*, pages 365–372, New York, USA, 2003. ACM Press.

[19] E. Hau. *Windkraftanlagen. Grundlagen, Technik, Einsatz, Wirtschaftlichkeit.*, volume 4th. Springer-Verlag, Berlin Heidelberg, 2008.

[20] A. Hertz, O. Marcotte, A. Mdimagh, M. Carreau, and F. Welt. Optimizing the Design of a Wind Farm Collection Network. *INFOR*, 50(2):95–104, 2012.

[21] R. Jothi and B. Raghavachari. Improved approximation algorithms for the single-sink buy-at-bulk network design problems. In T. Hagerup and J. Katajainen, editors, *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science 3111, pages 336–348, Berlin Heidelberg, 2004. Springer-Verlag.

[22] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer-Verlag, Berlin Heidelberg, 2004.

[23] I. Ljubić, P. Putz, and J.-J. Salazar-González. Exact approaches to the single-source network loading problem. *Networks*, 59(1):89–106, 2012.

[24] P. A. Lynn. *Onshore and Offshore Wind Energy: An Introduction.* John Wiley & Sons Ltd., Chichester, 2012.

[25] T. L. Magnanti, P. Mirchandani, and R. Vachani. Modeling and Solving the Two-Facility Capacitated Network Loading Problem. *Operations Research*, 43(1):142–157, 1995.

[26] J. F. Manwell, J. G. McGowan, and A. L. Rogers. *Wind Energy Explained: Theory, Design and Application.* John Wiley & Sons Ltd., Chichester, second edition, 2009.

[27] A. X. Martins, M. C. Souza, M. J. F. Souza, and T. A. M. Toffolo. GRASP with hybrid heuristic-subproblem optimization for the multi-level capacitated minimum spanning tree problem. *Journal of Heuristics*, 15(2):133–151, 2008.

[28] A. Meyerson, K. Munagala, and S. Plotkin. Cost-Distance: Two Metric Network Design. *SIAM Journal on Computing*, 38(4):1648–1659, 2008.

[29] C. A. Pappas, A.-C. G. Anadiotis, C. A. Papagianni, and I. S. Venieris. Heuristics for the multi-level capacitated minimum spanning tree problem. *Optimization Letters*, 8 (2):435–446, 2013.

[30] L. Roskoden. WikiEnergy – Energieanlagenkarte für regenerative und konventionelle Energien, June 2014. URL `http://www.wikienergy.de/`. [Online; accessed 22/06/2014].

[31] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-Bulk Network Design: Approximating the Single-Sink Edge Installation Problem. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1997.

[32] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Approximating the Single-Sink Link-Installation Problem in Network Design. *SIAM Journal on Optimization*, 11(3):595–610, 2000.

[33] F. S. Salman, R. Ravi, and J. N. Hooker. Solving the Capacitated Local Access Network Design Problem. *INFORMS Journal on Computing*, 20(2):243–254, 2008.

[34] M. Samorani. The wind farm layout optimization problem. In P. M. Pardalos, S. Rebennack, M. V. F. Pereira, N. A. Iliadis, and V. Pappu, editors, *Handbook of Wind Power Systems*, Energy Systems, pages 21–38. Springer-Verlag, Berlin Heidelberg, 2013.

[35] K. Talwar. The Single-Sink Buy-at-Bulk LP Has Constant Integrality Gap. In W. J. Cook and A. S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, number 1 in Lecture Notes in Computer Science 2337, pages 475–486, Berlin Heidelberg, 2002. Springer-Verlag.

[36] O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. URL `http://www.gnu.org/s/parallel`.

[37] E. Uchoa, T. A. M. Toffolo, M. C. de Souza, A. X. Martins, and R. Fukasawa. Branch-and-cut and hybrid local search for the multi-level capacitated minimum spanning tree problem. *Networks*, 59(1):148–160, 2012.

[38] A. Zuylen. Deterministic Sampling Algorithms for Network Design. *Algorithmica*, 60 (1):110–151, 2011.