Karlsruhe Institute of Technology

Theory and Algorithms of the Solar Farm Cable Layout Problem

Master Thesis of

Dominik Stampa

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	PD Dr. Torsten Ueckerdt
	Juniorprofessor Dr. Thomas Bläsius
Advisors:	Sascha Gritzbach
	Matthias Wolf

Time Period: 1st August 2021 – 1st February 2022

Statement of Authorship

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, February 1, 2022

Abstract

Solar energy is a renewable and sustainable energy, which gets more and more important in times where humanity aims to reduce the usage of fossil fuels. Photovoltaic modules are used to convert sun light into electricity. Often this is done in large solar farms. We model a solar farm as layered graph, where the power generated by the strings (several connected photovoltaic modules) needs to be conducted through the layers of the graph. For the connection of two vertices there are different types of cables with different capacities and costs. The problem is now to find a cable layout with minimal costs, which does not violate cable or vertex capacities. This optimization problem is analyzed and it is shown that the *Solar Farm Cable Layout Problem* is NP-hard for several variants. For the general variant it is even NP hard to find any feasible solution. A heuristic algorithm to solve a special but realistic variant of the problem is proposed. It is compared and evaluated with an MILP formulation of the problem. For large solar farms the heuristic is able to find a solution to the problem where the MILP is not able to find one within 24 hours runtime.

Deutsche Zusammenfassung

In Zeiten, in denen die Menschheit versucht den Gebrauch von fossilen Brennstoffen zu reduzieren, werden erneuerbare Energien und damit insbesondere auch Solarenergie immer wichtiger. Um das Licht der Sonne in Strom umzuwandeln, werden Solarmodule (auch Photovoltaikmodule genannt) verwendet. Häufig geschieht das in großen Solarparks. Wir modellieren solche Solarparks als geschichtete Graphen, in denen der Strom, der in den Strings (Reihenschaltung von mehreren Solarmodulen) produziert wird, durch alle Schichten des Graphen geleitet werden muss. Damit der Strom durch den Graphen geleitet werden kann, müssen die Knoten mit Kabeln verbunden werden, wofür verschiedene Kabel mit unterschiedlichen Kosten und Kapazitäten zur Verfügung stehen. Das Ziel ist nun eine Verkabelung zu finden, die minimale Kosten hat, bei der aber keine der Kabel- oder Knotenkapazitäten überschritten werden. Dieses Optimierungsproblem wird theoretisch untersucht und wir zeigen, dass verschiedene Varianten des Verkabelungsproblems NP-schwer sind. Außerdem wird gezeigt, dass es für die allgemeine Variante des Problems sogar NP-schwer ist irgendeine beliebige gültige Lösung zu finden. Wir stellen zusätzlich einen heuristischen Algorithmus vor, um eine realistische Variante des Verkabelungsproblems zu lösen. Eine Implementierung dieses Algorithmus' wird mit einer MILP Formulierung des Problems verglichen. Für große Solarparks ist der Algorithmus in der Lage, Lösungen für Instanzen zu finden, bei welchen das MILP nach 24 Stunden keine Lösung finden konnte.

Contents

1. Introduction

In times of climate change renewable energy sources are getting more and more important. After hydropower and wind solar power is the third biggest renewable energy source. In 2020 11% of the energy produced worldwide by renewable energy sources was solar energy [RR20]. Additional to that solar energy production is increasing rapidly in the last few years. In 2010 the share of solar power in the renewable energy sources was less than 1% [RR20]. Usually one thinks of photovoltaic systems as solar power and this is indeed the main technology when sunlight is converted into electricity. The annually added capacity of renewable energy in general and photovoltaics (PV) has been increasing over the last few years [Age21a][RR20]. The world wide added capacity of photovoltaic power in 2020 was more than 130 GW and still it is expected that this value increases in the years to come [Age21a]. More than half of the newly installed PV capacity is contributed by utility scale PV systems also called (large scale) solar farms. These types of PV systems are not only a few PV-modules installed on the roof of someones private house, but cover tens of hectares of land and supply several MW of electric energy and often a lot more. For example the solar farm Lauingen Energy Park [AG10] in Germany can supply up to 25 MW of electric energy. That means about 7500 households can be provided with electricity [AG10].

In a solar farm the actual conversion of sunlight to electricity is done by the solar modules or also called PV modules. Usually the power produced by a solar farm is fed into the public electricity grid. But PV modules cannot be connected to the grid directly, so the electricity needs to be directed through some electrical devices before it is possible and allowed to be fed into the grid. An example for these devices are inverters which turn the DC power produced by the PV modules into AC power, which is the power that runs through the grid. For the connection of the PV modules through several intermediate points to the grid a lot of cables are needed. For Lauingen Energy Park as an example, more than 600 km of solar cable trays were installed [AG10]. In total the costs for a solar farm are about one to three million dollars per Megawatt of planned capacity [Lab21] [Yac21]. It is hard to say how high the share of cable costs is exactly as they are usually not listed explicitly as a cost factor, but combined with other hardware [Age21b] [Lab21]. Roughly estimated the share is about 3%, so the costs for cables can easily reach the million dollar mark.

Because of that, optimizing the cable layout of a solar farm in a way that the total cost of all cables is minimized could potentially save a lot of money. So we get an optimization problem where we are given the positions of PV modules and the position of transformers and the modules have to be connected by cables to the transformers via several intermediate

devices. The intermediate devices are divided into several layers, so it is only allowed to connect devices of two consecutive layers. That means the connection of a PV module to a transformer runs through all layers of the solar farm. There can also be further restrictions where to lay and where not to lay cables. As all the electric devices have specified how much electric current they can carry, we need to do the cabling such that these capacities are not exceeded. Additional to that we also have different cable types which each have a cost and a capacity. We need to choose which intermediate devices we want to connect and use for the cabling. If we connect two devices a cable needs to be chosen such that the capacity of the cable is not exceeded. In the end we want to get a cabling with low or minimal costs.

2. Related Work

Solar farms are a widely investigated topic. The investigations range from the question how to find a suitable site to build a solar farm [TSAMA17][PGB19] via studies about the impact of a solar farm on the power network [VRVD16] to analysis of the efficiency of components used in a solar farm [APA18][LYC+19]. Many investigations around solar farms are about the electric properties of the solar farm. For example Lui et al. propose a new PV transformer design [LYC+19]. They implement their new design and show its efficiency by a practical evaluation.

Other works focus on the efficiency of the maximum power point tracking (MPPT). Maximum power point tracking is used to get the maximum possible power out of a solar farm depending on the current and voltage that is produced by the PV modules. There are many different MPPT algorithms. Bollipo et al. give a review of 23 different MPPT methods and show downsides and advantages for these [BMB20].

Another source of inefficiency are the PV modules itself. PV modules produce a lot of heat, which lets them work less efficient. That is why Glick et al. analyze the impact of the tilt angle of PV modules to the heat transfer [GAB+20]. They propose a layout design that improves the air flow around the PV modules and therefore could improve the power production.

All the above mentioned works mainly focus on the efficiency of the solar farm, so they try to increase the power that can be produced by a solar farm. Also when analyzing possible locations for solar farms the possible power output, which depends among other things on the solar irradiation, plays a decisive role [PGB19]. However the cost efficient construction of a solar farm is not a much investigated field, especially regarding a cost efficient cabling. To our best knowledge the only work that also considers the cabling layout of a solar farm is the master thesis of Trofast [Tro20]. He optimizes the layout of a solar farm for a special string inverter of Ampner. The thesis considers an imaginary solar farm site in Portugal and tries to find the most cost efficient layout. For this he does not only consider the cables but also the positions of PV modules and inverters. He proposes how many modules should be connected to each inverter and where the inverters should be placed relative to the modules. As the investigated case is a very special one we want to propose a model which helps to optimize the cable layout of a solar farm in a more general way. We now take a look at problems that are similar to the solar farm cabling problem.

2.1. Wind Farm Cabling

The cabling problem for solar farms was inspired by the wind farm cabling problem. Here wind turbines and substations are given and the turbines need to be connected to the substations. A wind turbine does not need to be connected directly to the substation but can be connected via several other turbines. For the connection of the turbines there are several cable types, which all have a different capacity and cost. The goal is now to find a connection of the turbines to the substation with minimal cost. Where the wind farm cabling problem has only two types of vertices, turbines and substations, the solar farm cabling problem has more vertex types but structured into different layers. The principle of the cabling however is the same as in the solar farm cabling problem. One of the first problem formulations of the wind farm cabling problem was given by Berzan et al. [BVMO11]. They additionally divide the problem into different subproblems and compare these different variants with already existing problems. For example is the wind farm cabling problem a generalization of the capacitated minimum spanning tree problem. Then they propose approaches how to solve these different problems.

Gritzbach et al. propose an algorithm to solve the wind farm cabling problem, which is based on negative cycle canceling [GUW+19]. They compare different variants of their algorithm with solutions of an Mixed Integer Linear Programming (MILP) formulation of the problem and a Simulated Annealing algorithm. Their negative cycle canceling algorithm calculates solutions of similar quality in less time than the MILP and the Simulated Annealing algorithm.

2.2. Facility Location

The structure of the facility location problem comes very close to the solar farm cabling problem, depending on the variant of the facility location problem. In the facility location problem there are given demand points (also called customers) and facility locations. The task is to open facilities at the given locations such that a given demand by the demand points is fulfilled with minimal costs. The costs result from opening facilities and from the distance (or service costs) of the demand points to the opened facilities. Additionally facilities can have restricted capacities and one has to assign the demand points to open facilities such that no capacity is exceeded. This is then called the capacitated facility location problem. There are some more varieties of this problem of which especially the multilevel facility location problem (MFLP) is interesting as it is very similar to the solar farm cabling problem. Here we are given k layers or levels of facilities. Each unit of demand has to be routed to the k-th level through a facility of each level. Usually the facility location problem as well as the MFLP is considered in its metric variant. That means service costs are symmetric and fulfill the triangle inequality. The approximation algorithms below also consider the metric variant of the problem.

Kratica et al. [KDS14] present an MILP formulation for the uncapacitated MFLP. They compare their formulation among other formulations with the LP formulation of Gabor and van Ommeren [GvO10]. Where Gabor and van Ommeren show that their LP rounding scheme is a 3-approximation for the MFLP, Kratica et al. do not approximate the MFLP but show that their MILP uses less variables. They show with experiments that this is an advantage in practice.

Bumb and Kern [BK01] present a primal-dual algorithm with an approximation guarantee of 6 for the uncapacitated MFLP and 12 for the capacitated version. The runtime of their algorithm is $O(n^4 \log n)$, where n is the number of demand points and facility locations together [Bum02].

Where the papers above considered the MFLP with an arbitrary amount of layers Gendron et al. [GKS16] consider the uncapacitated facility location problem with exactly two levels of facilities (With the demand points together there are three levels). The problem is additionally constrained by a single assignment restriction, that means each first-level facility is assigned to at most one second-level facility. This restriction is also part of the solar farm cabling problem. A solution represented as graph would then consist of a forest of trees rooted in second-level facilities. Gendron et al. present a branch-and-bound algorithm and show its efficiency with experimental results.

As the solar farm cabling problem also has lower bounds on some vertices (the inverters) the work of Ahmadian and Chaitanya [AS13] shall be mentioned as they present an approximation algorithm for the facility location problem where facilities have a lower bound. That means if a facility is opened it has to serve a minimum amount of demand points. Although their algorithm has a constant approximation ratio this value is very high with 82.6.

2.3. General Assignment Problem

The general assignment problem is a generalization of the multiple knapsack problem, which is a subproblem of the solar farm cabling problem. Given m bins with capacities and n items which have a different weight and value for each of the bins, the task is to pack items into the bins without exceeding their capacities such that the value of the items in the bins is maximized. Fleischer et al. [FGMS06] present an approximation algorithm for the general assignment problem based on LP relaxation. Their approximation ratio is $1 - \frac{1}{e}$, but for this ratio they need an exact solution of the single knapsack problem. As this is not possible in polynomial time unless P=NP, the approximation guarantee with a polynomial running time is $1 - \frac{1}{e} - \epsilon$ when using an FPTAS to solve the knapsack problem.

2.4. Contribution and Outline

As we have just seen there are on the one hand problems that are very similar or special cases of the solar farm cabling problem. On the other hand topics around solar farms are widely investigated. But to our best knowledge optimizing the cabling layout is not among them.

This thesis proposes a model for the described cabling problem where a solar farm is modeled as graph. It is shown that several variants of this optimization problem are very difficult to solve in a sense that they are NP-hard. If no further assumptions are made even approximating a solution for the cabling problem is NP-hard for any approximation ratio. For an realistic variant of the problem, where the output of a device can be connected to any following device without further restrictions, we propose an heuristic algorithm and evaluate the performance of this algorithm by comparing it to an MILP formulation. To compare the heuristic and the MILP, we created some test instances. The heuristic is able to solve a large portion of the test instances, but struggles with bigger sized instances in terms of solution quality compared to the MILP formulation. However the heuristic is able to find solutions for large instances where the MILP is not able to find one within 24 hours.

In the next section we will give a short introduction into the typical structure of a solar farm and some basics of graph theory and flows. After that we model the solar farm cable layout problem in Chapter 4 and analyze its theoretical complexity in Chapter 5. We show the NP-hardness for several different variants of the cabling problem. In Chapter 6 and Chapter 7 some algorithms to solve the cabling problem exactly and heuristically are shown. Of these we implement and compare a heuristic and an MILP formulation (Chapter 8). In Chapter 9 we discuss possible problems with the model and draw a conclusion also considering the results of our experimental comparison.

3. Preliminaries

Before defining the problem we first explain some basics of graph theory and solar farms.

3.1. Graphs

A *directed graph* G is a pair (V, E) of a set of *vertices* V and a set of *edges* E with $E \subseteq V \times V$. For an edge $(u, v) \in E$ u is the *start* or *initial* vertex and v the *end* or *terminal* vertex. We visualize directed graphs by drawing points for the vertices and arrows for the edges. An edge (u, v) is represented by an arrow pointing from u to v. If an edge $(u, v) \in E$ or $(v, u) \in E$ is connected to a vertex $v \in V$, this edge is *incident* to v. The number of incident edges of a vertex is called the *degree* of this vertex. The degree of a vertex v can be split into the *in-degree*, which is the number of edges ending at v, and the *out-degree*, the number of edges starting at v. Edges (v, v) starting and ending at the same vertex are called *loops*. The graphs we consider in this thesis do not have loops.

A list of vertices $v_0, \ldots, v_n$ is called *path*, if for all $i \in \{0, \ldots, n-1\}$ $(v_i, v_{i+1}) \in E$. If any pair of vertices in a graph is connected by exactly one path, this graph is called a *tree*.

3.1.1. Flow

As the cabling problems we want to consider below can be seen as some special kind of flow problems, we now are going to explain flow networks.

We are given a directed graph $G = (V, E)$ and a capacity for each edge, which is given by a function $c : E \to \mathbb{R}_{\geq 0}$. Additionally there are a *source* $s \in V$ and a *sink* $t \in V$. A *flow* $f : E \to \mathbb{R}_{\geq 0}$ is a function which assigns a flow-value to every edge. Thereby a flow has to fulfill the following constraints:

- capacity constraint: $\forall e \in E : f(e) \leq c(e)$
- flow conservation: $\forall v \in V \backslash \{s, t\} : \sum_{(u,v) \in E} f((u, v)) = \sum_{(v,w) \in E} f((v, w))$

The total flow F between s and t can then be calculated by

$$F = \sum_{(s,v) \in E} f((s, v)) - \sum_{(u,s) \in E} f((u, s)).$$

Given a flow network, that means a tuple (G, c, s, t) as defined above, one could ask for example to maximize F.

The Minimum Cost Flow Problem is an optimization problem very similar to the problems considered in this thesis. Additional to the flow network a cost function $cost : E \to \mathbb{R}$ is given. If $f(e)$ units of flow are sent over edge e, this results in costs $cost(e) \cdot f(e)$. For the Minimum Cost Flow Problem we want to find a flow f, where $\sum_{(s,v)\in E} f((s,v)) - \sum_{(u,s)\in E} f((u,s))$ is given. The goal is to find a flow with minimum costs.

3.2. Components of a solar farm

After some theoretical basics we now take a look at solar farms and their components. The power of a solar farm is produced in photovoltaic (PV) cells and then conducted via several components to transformers and fed into the power grid. Of course a solar farm does not need to be connected to the grid, but we focus on large-scale solar farms that are grid connected. Therefore the typical components and layout of such a solar farm shall be described.

3.2.1. PV string

PV cells are the smallest power generating unit in a photovoltaic system. As the cells produce a very small voltage of less than $1V$ and a small current of a few $\frac{mA}{cm^2}$ [Mer20, p. 121], several cells have to be connected together. The parallel and/or series connection of multiple PV cells is called a PV *module* [ABB19, p. 9][Mer20]. A parallel connection of cells increases the current, a series connection increases the voltage [Mer20, p. 165ff.]. But the voltage and current of a typical PV module are still not sufficient for a large-scale solar farm. That's why modules are connected in series and form a so called *string*. In a solar farm the modules of one string would then be mounted next to each other onto a rack and connected in series [ABB19]. A string produces a voltage U, a current I and a power P , which is basically the product of the voltage and the current. So $P = U \cdot I$.

3.2.2. Inverter

The current produced by PV cells and therefore also by the strings is direct current (DC) [Wag19, p. 120]. To feed the generated power into the power grid, alternating current (AC) is needed. An *inverter* converts the direct current into alternating current. In large-scale solar farms usually two types of inverters are used: *Central inverters* or *(multi) string inverters*. Central inverters have much higher capacity regarding the power that can be connected. String inverters have a lower capacity and the strings are directly connected to them whereas for central inverters additional DC combiner boxes are needed [ABB19, p. 12f.]. Sometimes it is distinguished between string inverters, which invert the current of exactly one string and multi string inverters, which are connected to several strings. However multi string inverter is not a very common term, so we will use string inverter even if we refer to inverters to which multiple strings are connected [ABB19, p. 13].

An inverter for a modern large-scale solar farm is more than just an inverter. Besides monitoring and safety components it contains a maximum power point tracker (MPP tracker) [Mer20, p. 203]. The MPP tracker basically adjusts the voltage to get the maximum possible power out of the connected strings. The inverter and all the other devices built into it bring some constraints with them. Usually this is a maximum power, as already mentioned above, a maximum current and a minimum and maximum voltage. The voltage range is dictated by the MPP tracker and is the range in which the tracker can work properly [SMAb]. This voltage range determines the string length. Additionally the number of connections is limited [SMAb].

3.2.3. Combiner Box

Combiner boxes are used to connect multiple strings in parallel. This can be done on DC side before getting to the inverter or on AC side after the inverters, but is not done on both sides at the same time [ABB19, p. 40]. The voltage of a single string is already near to the maximum possible voltage of the inverter. That means it is not possible to connect two or more strings in series, as this would lift the voltage above the maximum value of the inverter. So strings are only connected in parallel either in a combiner box or directly in the inverter, resulting in an increase of the current only. String inverters do not need DC combiner boxes for the few strings that are connected to them but often are used with AC combiner boxes. As central inverters have much more strings connected to them, the strings need to be combined before in DC combiner boxes [ABB19] [Mer20, p. 202]. It is possible to install more than one layer of combiner boxes, the boxes of the second layer are also called recombiner boxes [ABB19, p. 40].

Combiner boxes bring a total maximum current and a maximum current per connection with them. Of course they have a maximum voltage, which is usually the same as for the inverters and the amount of physical connections is also limited [SMAa]. Most combiner boxes come with only one output channel, but there are boxes with the possibility to use two outputs [Ele20] [SMAa].

Actually there is another method of combining strings next to combiner boxes and inverters, which are Y-connectors or branch connectors. These connectors allow the parallel connection of two strings (sometimes more) and can be used before connecting the cables to the combiner boxes or string inverters [ABB19, p. 51] [Eve16]. Often they are used in combination with thin-film PV modules, which have a lower current than the classic crystalline silicon modules [ABB19, p. 51][Mer20, p. 174ff.][Sol21]. Connecting more than two strings is rarely possible before reaching the combiner boxes or string inverter, because of the maximum current per connection rating, which is usually not high enough for more than two strings. Sometimes even two strings already have a too high current. Y-connectors could be seen as a special type of combiner box, with a very low current capacity that is exactly the current produced by two strings.

3.2.4. Transformer

The voltage of a solar farm is in a low voltage range of typically $1000V$ to $1500V$, which is the voltage a common inverter supports [ABB19][SMAb]. On the AC side the voltage is even a bit less and below $1000V$ [ABB19, p. 46]. As this is far below the voltage of the power grid a *transformer* is needed to step up the voltage, so that the generated power can be fed into the power grid.

The constraints of a transformer are a maximum power and voltage [Cor15, p. 73ff.].

3.2.5. PV cables

The cables to conduct the power from the strings via combiner boxes and inverters to the transformers have different requirements. First there are DC and AC cables. Especially the DC cables can be installed underground or overground. Cables connecting the strings to the combiner boxes respectively string inverters are fixed to the racks the modules are mounted on. These cables need to withstand extreme weather conditions [ABB19, p. 50]. Between DC combiner boxes and inverters cables are often buried. On the AC side they are mostly buried too. The cables have a maximum voltage and a current carrying capacity, also called ampacity [ABB19, p. 53]. As the voltage is at least on DC side and AC side of the farm constant, the current carrying capacity is the more interesting constraint. If

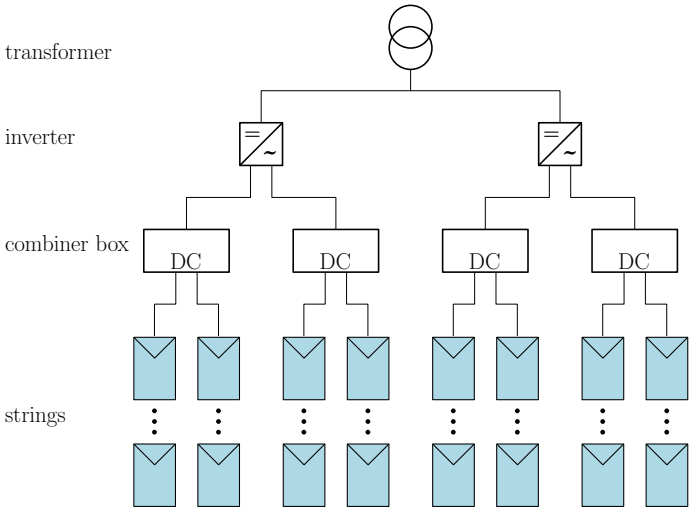


Figure 3.1.: Schematic representation of a solar farm with central inverters and combiner boxes on DC side.

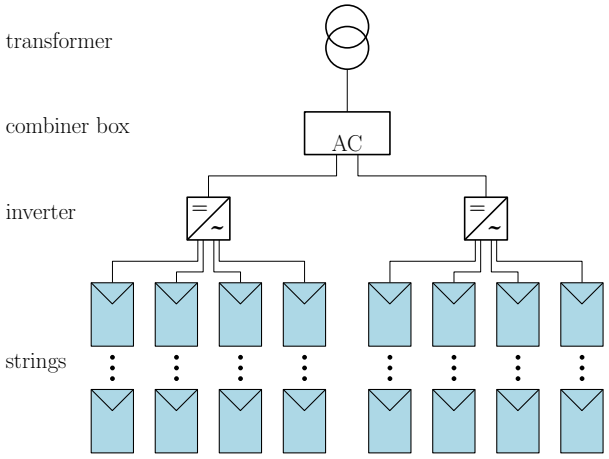


Figure 3.2.: Schematic representation of a solar farm with string inverters and combiner boxes on AC side.

several cables are installed in the same conduit, the ampacity of these cables is decreased because of waste heat [Nat13].

As the design of a solar farm is optimized for efficiency, that means minimizing power losses, there are guiding values how much power may be lost in the cables [ABB19, p. 54][Mer20, p. 184]. The losses depend on the length of the cable, where shorter is better, the material of the conductor, the cross section of the cable, where larger is better and the current flowing through the cable [Mer20, p. 183f.].

4. The solar farm cabling problem

4.1. Defining the problem

Now knowing the typical layout of large-scale solar farm, our goal is now to find a theoretic model of a solar farm with which we can optimize the cabling costs. A solar farm will be represented as a layered directed graph $G = (V, E)$, in which the edges E represent the possibilities to install cables between the vertices V, which represent the solar farm components like strings, combiner boxes and inverters. Each layer of the graph consists of only one type of vertices.

In the following we consider each component and argue, what properties of the components we want to model. Later in Chapter 9 we discuss what might be other properties that could be reasonable.

4.1.1. Strings

We take strings as the basic power generating unit in our model. The set of all strings shall be S. Usually all strings in a solar farm have the same length, which is the case for efficiency reasons. Strings with strongly different electric properties should not be connected to the same MPP tracker, because this would result in power losses [Whi16, p. 54]. And as different string lengths are not typical, we can assume that all strings produce the same power. Of course strings can still have a different orientation, which could cause different power outputs. This however is not as problematic as different string lengths when connecting them in parallel [Whi16, p. 54]. Still one could decide to connect strings with different orientations to the same MPP tracker. The orientation can be represented as a function that assigns an azimuth angle to every string $a : S \to [0, 360]$. A string facing directly north would have a azimuth angle of 0°, facing east would be 90° and so on. With the azimuth angle we can now forbid to connect two strings to the same MPP tracker if the difference of their angles is above a certain maximum $angle_{max}$. However strings could also have a different tilt angle, which determines the angle of the incoming sun rays and thereby the irradiance. In opposition to the azimuth angle, the tilt angle does not vary very much in a large-scale solar farm. There are different methods to get the optimal tilt angle for a specific location and one wants to install the modules with this calculated angle [ABB19, p. 29f.] [JJ18]. Even for the azimuth angle differences in the same solar farm are rarely the case, but for example the solar farms Bruchköbel [MK16] or Cestas [Ken15] have modules with differing azimuth angles. In the case of different orientations string

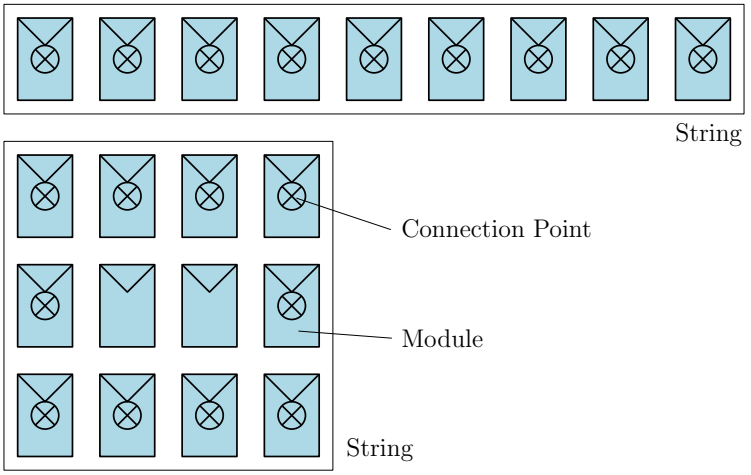


Figure 4.1.: Example of string layouts and possible resulting connection points. For the lower layout it is not reasonable to connect a cable to the middle of the string. That is why there are two modules without a connection point.

inverters are often used instead of central inverters. String inverters can cope better with differing conditions because they divide the PV farm into smaller partitions [ABB19, p. 40]. Within such a partition the conditions for the strings then can be very similar.

Back to the strings, we can assume that they all have the same length and the same tilt angle, which would then result in the same maximum voltage and current for all strings. As the voltage does not change when connecting several strings, the important value is the current or the power which could be seen as equivalent in a model. We will use current in the following or directly speak of flow in graphs.

The whole discussion about different orientations ignores tracking systems, where the PV modules are fixed to single or dual axis trackers that can change the azimuth and the tilt angle. With sinking PV module prices especially dual tracker systems are rarely built anymore [Mer20, p. 187]. But even then tracking systems usually follow the sun, so all strings could be seen as having the same angles.

A string consists of several modules, so each string $s \in S$ has some *connection points* $V_{con}^s \subsetneq V$. These are the vertices, where a cable can be connected to the string and there would be a connection point for every outer module of the string. The connection points of one string then typically are arranged in a rectangle, but the actual arrangement is a question of modeling a particular solar farm and does not need to be rectangular. For two different strings the corresponding connection points are disjoint sets. All connection points of all strings together are V_{con}.

4.1.2. Inverter

The power generated by the strings has to be conducted to the inverters, which are a subset of all vertices $V_I \subsetneq V$. As the voltage is already fixed by the string length, we do not need to worry about voltage constraints of the inverter. Minimum and maximum power respectively maximum current are the more important constraints and result in two values $i_{min}, i_{max} \in \mathbb{N}$ with $0 \leq i_{min} \leq i_{max}$. If the inverters are different this would lead to a function $imax : V_I \to \mathbb{N}$ and $imin$ respectively. Current conducted to one inverter has to be in the range of i_{min} and i_{max} (or of course can be zero). We call i_{max} also the capacity of the inverter. The minimum value, also lower bound, is there for efficiency reasons and not necessarily because of manufacturer constraints. When designing a solar farm one normally decides whether to undersize or oversize an inverter. Undersizing means

connecting more power, which is more strings, to the inverter than the nominal power. Oversizing is the opposite. This is done for efficiency reasons and usually depends on the geographic position of the solar farm, the climatic conditions and the efficiency rating of the inverter [Mer20, p. 215]. The efficiency of an inverter depends on the power that is currently reaching the inverter and if this power is too low the efficiency drops significantly. That is why there is the minimum value i_{min}.

Additionally an inverter has a number of connections $con_I \in \mathbb{N}$. This is the number of cables that can be connected to the inverter. If strings can only be connected with the inverters via combiner boxes, this constraint might not be necessary because the amount of combiner boxes and i_{max} already limit the connections in a way. Nevertheless ignoring this constraint could result in unrealistic cabling solutions.

In an inverter the voltage and current are usually changed, but the voltage is more or less constant [SMAb] [Ele19]. So the current again is the important value. We say that the output of an inverter is the same as the input. This makes it possible to use graph flows for modeling, but is a very questionable method for AC power.

Maximum input currents for the single channels could be modeled by simply restricting the maximum cable capacity, however we decide to not model these.

Additionally one could model the number of MPP trackers in an inverter, which would allow to connect strings with different properties to the same inverter, in the case that the inverter has more than one MPP tracker.

4.1.3. Combiner box

Combiner boxes are vertices $V_C \subsetneq V$ where several cables can be linked together into a new cable. They are possible both on DC and AC side, but normally with string inverters they would appear on AC side and with central inverters on DC side. A combiner box has a maximum current c_{max}, or $c_{max} : V_C \to \mathbb{N}$ if there are different boxes, which is allowed to be connected to it and each single connection also has a maximum allowed current. However this constraint does not have to be considered if we do not allow connections between combiner boxes because then the current coming from one cable is fixed. Either it comes from a string, where we cannot influence the current or it comes from an inverter, where the maximum current depends on i_{max}. As the allowed current of one connection is usually very small [SMAa], not allowing connections between combiner boxes is also reasonable. When adding more layers to the model by adding recombiner boxes V_R and Y-connectors V_Y this could open the possibility to skip layers. As combiner boxes (and recombiner boxes) also contain fuses and other safety devices besides only paralleling strings, this is not really an option for safety and monitoring reasons.

As recombiner boxes and Y-connectors basically have the same constraints as combiner boxes we only speak of combiner boxes, but they shall include recombiners and Y-connectors.

Combiner boxes, like inverters, also have a maximum number of connections. But as current produced by the strings is fixed, this value should be integrated into c_{max} for DC combiner boxes. For the AC combiner boxes we can also argue, that c_{max} should depend on i_{min} and i_{max}. Although there are some combiner boxes which offer the possibility of two outputs, we only allow one output per combiner box

The cabling problem could be complemented by costs for combiner boxes $cost_C \in \mathbb{R}$. If a cable is connected to a combiner box the costs have to be paid.

4.1.4. Transformer

If the cables reach the transformers $V_T \subsetneq V$, the cabling is done. Every transformer has a maximum capacity $t_{max} : V_T \to \mathbb{N}$ or if only one type $t_{max} \in \mathbb{N}$.

There could be two layers of transformers in a solar farm, but we only model the first one.

4.1.5. Cables

Cables are needed to conduct the power produced in the strings to the transformers. We need different cables for AC and DC side. The cables for the connection directly at the strings are fixed, as the power generated by the strings is fixed. So if there are no Y-connectors between strings and string inverter, we do not need to model DC cables at all.

But usually we have a set of AC cables C_{AC} and the two functions $cap_{AC} : C_{AC} \to \mathbb{R}$ and $cost_{AC} : C_{AC} \to \mathbb{R}$, which assign a capacity and cost to every cable and a set of DC cables C_{DC} of which each cable has a capacity $cap_{DC} : C_{DC} \to \mathbb{R}$ and a cost $cost_{DC} : C_{DC} \to \mathbb{R}$.

The goal is to assign a cable to every edge with the constraint that a cable has to have enough capacity for the power that is conducted through it. If an edge shall have no cable assigned to it, we have a special cable c_0 with no cost and capacity.

In a solar farm several cables can be installed in the same physical conduit, which leads to the already mentioned problem of capacity (ampacity) derating. We do not model this property, which is definitely a weakness. To model this, one could insert special vertices in to the graph where several cable could meet but are not combined into a new cable. The cables would then run parallel to each other and a function $derate : \mathbb{N} \to [0, 1]$ could give the derating factor for the number of cables assigned to the same edge.

We also do not consider power losses in the cables. One could check for a cabling produced by this model whether guiding values of how much power may be lost in the cables are violated.

4.1.6. Resulting model(s)

Casting all the constraints above into one model is not easy, especially the difference between string and central inverters is a problem. We also do not model every property, but make more simplifying assumptions which then result in the model explained below.

What is always given is a directed graph $G = (V, E)$ and a set of strings S. The vertices V consist of:

- $V_{con} = \cup_{s \in S} V_{con}^s$: the union of the connection points of every string $s \in S$
- V_Y: the set of Y-connectors. They have a capacity $y_{max} \in \mathbb{N}$
- V_C: the set of combiner boxes. Each combiner box has a capacity $c_{max} \in \mathbb{N}$
- V_R: the set of recombiner boxes. Each recombiner box has a capacity $r_{max} \in \mathbb{N}$.
- V_I: the inverters. Each inverter has a minimum and maximum allowed current $i_{min}, i_{max} \in \mathbb{N}_0$
- V_T: the transformers. Each transformer has a capacity $t_{max} \in \mathbb{N}$

All capacity constraints could of course be modeled as functions, so that every element can have its individual constraints. Especially Y-connectors and recombiner boxes could be integrated into the set of combiner boxes, so we don't have to distinguish between them as they all have the same properties, but different values. However in the following we will not do this. The edges of G have a length given by the function $len : E \to \mathbb{R}$.

To connect the components there are two sets of cables, C_{DC} for the DC side and C_{AC} for the AC side. Each cable has a capacity $cap_{DC} : C_{DC} \to \mathbb{N}$ respectively $cap_{AC} : C_{AC} \to \mathbb{N}$ and a cost $cost_{DC} : C_{DC} \to \mathbb{R}$ respectively $cost_{AC} : C_{AC} \to \mathbb{R}$.

The main difference between string inverters and central inverters is the position of the inverters in the graph.

4.1.6.1. Central inverters

Central inverters are next to the transformers, so the set of edges is $E \subseteq \{(s,y)|s \in V_{con}, y \in V_Y\} \cup \{(y,c)|y \in V_Y, c \in V_C\} \cup \{(c,r)|c \in V_C, r \in V_R\} \cup \{(r,i)|r \in V_R, i \in V_I\} \cup \{(i,t)|i \in V_C, t \in V_T\}$. The graph then is a layered graph with the edges pointing upwards (see Figure 4.2).

We can divide the edges into DC and AC edges. AC edges are the edges between inverter and transformer $E_{AC} = E \cap \{(i,t)|i \in V_C, t \in V_T\}$ and DC edges are then all the other edges between strings and inverters $E_{DC} = E \backslash E_{AC}$.

We want to find a solution consisting of a function $f : E \to \mathbb{R}$ and two functions $h_{DC} : E_{DC} \to C_{DC}$ and $h_{AC} : E_{AC} \to C_{AC}$. For all edges starting at string connection points there are only two possibilities for cables. Either there is no cable or the cable with the lowest capacity as there can be maximal one unit of flow on such an edge and we want to find the cheapest solution. A solution should satisfy the following constraints:

Each string should be connected at exactly one of its connection points.

$$\forall s \in S: \ \exists!\ e = (v,y) \in E \text{ with } f(e) = 1, v \in V_{con}^s, y \in V_Y \tag{4.1}$$

The cable capacity on each edge $e \in E_{DC} \cup E_{AC}$ must not be exceeded.

$$\begin{aligned} \forall e \in E_{DC} : f(e) \leq cap_{DC}(h_{DC}(e)) \\ \forall e \in E_{AC} : f(e) \leq cap_{AC}(h_{AC}(e)) \end{aligned} \tag{4.2}$$

The capacities of combiner boxes, inverters and transformers must not be exceeded.

$$\begin{aligned} \forall c \in V_C : \sum_{(v,c)\in E} f(v,c) \leq c_{max} \\ \forall r \in V_R : \sum_{(c,r)\in E} f(c,r) \leq r_{max} \\ \forall y \in V_Y : \sum_{(v,y)\in E} f(v,y) \leq y_{max} \\ \forall i \in V_I : \sum_{(v,i)\in E} f(v,i) \leq i_{max} \\ \forall t \in V_T : \sum_{(v,t)\in E} f(v,t) \leq t_{max} \end{aligned} \tag{4.3}$$

A minimum amount of current has to be conducted to each inverter or no current at all.

$$\forall i \in V_I : \sum_{(v,i)\in E} f(v,i) \geq i_{min} \ \vee \sum_{(v,i)\in E} f(v,i) = 0 \tag{4.4}$$

Each inverter and combiner box has at most one outgoing cable.

$$\forall i \in V \backslash (V_{con} \cup V_T) : \ \exists!\ e = (i,t) \in E \text{ with } f(e) > 0 \vee \forall e = (i,t) \in E : f(e) = 0 \tag{4.5}$$

No vertex except for strings or transformers shall produce or absorb flow.

$$\forall v \in V \backslash \{V_{con} \cup V_T\} : \sum_{(u,v)\in E} f(u,v) = \sum_{(v,t)\in E} f(v,t) \tag{4.6}$$

The goal is now to find a solution that minimizes the cost function

$$cost_{cables} = \sum_{e\in E_{AC}} cost_{AC}(h_{AC}(e)) \cdot len(e) + \sum_{e\in E_{DC}} cost_{DC}(h_{DC}(e)) \cdot len(e) \tag{4.7}$$

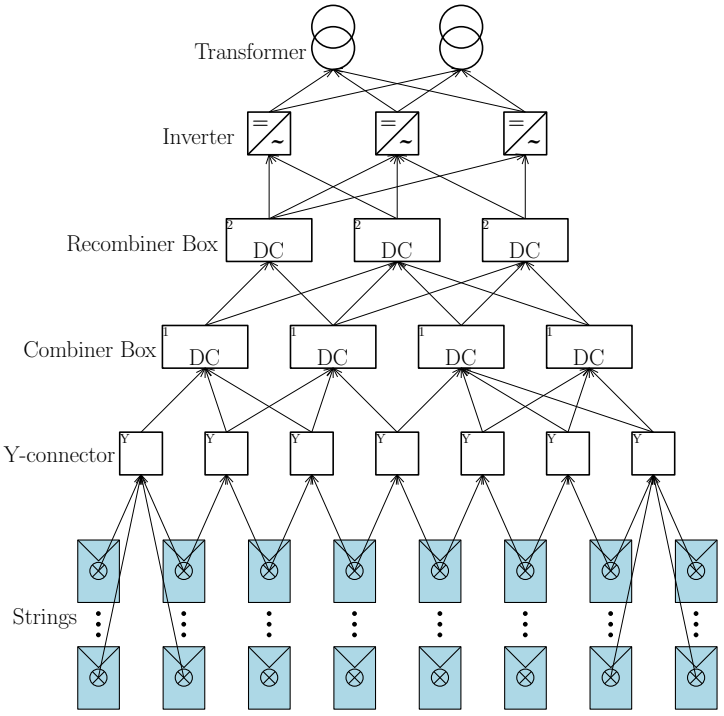


Figure 4.2.: Illustration of the solar farm graph for central inverters.

4.1.6.2. String inverters

The cabling problem with string inverters is almost the same as the one for central inverters. The main difference is the position of the string inverters respectively the set of edges $E \subseteq \{(s,c)|s \in V_{con}, y \in V_Y\} \cup \{(y,i)|y \in V_Y, i \in V_I\} \cup \{(i,c)|c \in V_C, i \in V_I\} \cup \{(c,r)|r \in V_R, c \in V_C\} \cup \{(r,t)|r \in V_R, t \in V_T\}$. An illustration of this layout can be seen in Figure 4.4.

The constraints and the solution are basically the same as with central inverters. When considering subproblems of the solar farm cabling problem the differences between string inverters and central inverters can increase, even with only small adjustments. For example when there are no Y-connectors the cables on DC side are only the ones between strings and inverters. This means one does not have to choose DC cables, as the cable with the lowest cost is obviously the only option for an optimal solution.

4.1.6.3. Decision Problem

With the modeling of the optimization problem above we can formulate the decision problem of the solar farm cabling problem. We do this again with the central inverter model.

Definition 4.1. *Let $W, y_{max}, c_{max}, r_{max}, i_{min}, i_{max} \in \mathbb{N}_0$. There is a graph $G = (V, E)$ with $V = V_{con} \cup V_Y \cup V_C \cup V_R \cup V_I \cup V_T$ and edges of length $len : E \to \mathbb{R}$. $E_{DC} \subsetneq E$, $E_{AC} \subsetneq E$. Additionally there are two sets of cables C_{AC} and C_{DC} of which each cable has a capacity $cap_{DC} : C_{DC} \to \mathbb{N}$ respectively $cap_{AC} : C_{AC} \to \mathbb{N}$ and a cost $cost_{DC} : C_{DC} \to \mathbb{R}$ respectively $cost_{AC} : C_{AC} \to \mathbb{R}$. Is there a flow $f : E \to \mathbb{R}$ and an assignment of cables to edges $h_{DC} : E_{DC} \to C_{DC}$, $h_{AC} : E_{AC} \to C_{AC}$, such that Equations 4.1 - 4.6 are fulfilled and $cost_{cables}$ (as defined in Equation (4.7)) is at most W?*

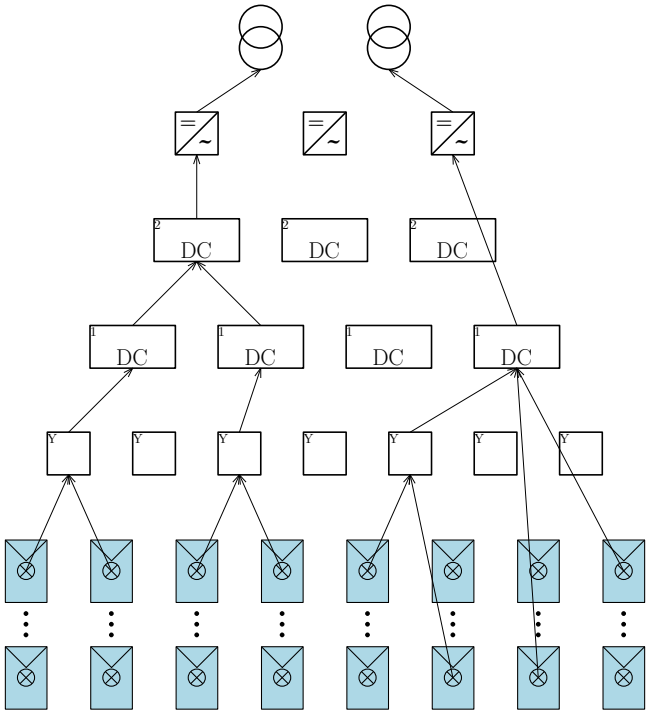


Figure 4.3.: Example cabling solution for central inverters.

4.1.7. Subproblems

We have now defined the complete solar farm cabling problem. But of course not every solar farm has Y-connectors, combiner and recombiner boxes. Often there are only combiner boxes and no Y-connectors or recombiner boxes, which leads to less layers. An illustration of this problem is shown in Figure 4.5 for string inverters. This special case for string inverters is also interesting as we do not need to consider DC-cables separately. The DC-cables are already fixed by the amount of flow coming from the strings. In the same manner we could add or delete other layers of combiner boxes depending on the design of the solar farm.

One problem that is considered in the next section is the problem which consists of a set of Strings S, inverters and transformers. So for our set of vertices this means $V = V_{con} \cup V_I \cup V_T$. All the properties regarding strings, inverters and transformers are the same as in the complete cabling problem. The edges connect the strings to the inverters to the transformers and again have a length given by $len : E \to \mathbb{R}$. We also have a set of cables, but we do not need DC-cables because all cables on the DC side are directly connected to the strings and therefore transmit exactly one unit of flow.

We can also consider proper subproblems of the solar farm cabling problem. That means we take only some of the layers of a solar farm and we can leave out more than only combiner boxes. One subproblem shall be explained more detailed as it is used in the next section.

We will call this subproblem the *two-layer cabling problem*. It consists of a graph $G = (V, E)$ of which the vertices are either combiner boxes V_C or inverters V_I and $E \subseteq V_C \times V_I$. The edges have a length $len : E \to \mathbb{R}$. Each combiner box $c \in V_C$ produces $p(c)$ units of flow, with $p : V_C \to \mathbb{N}$. $p(c)$ would be the number of strings connected to the combiner box c in a whole solar farm. The inverters have a capacity $i_{max} \in \mathbb{N}$ and a minimum allowed current $i_{min} \in \mathbb{N}_0$ as usual. There is one cable with which each combiner box has to be connected to one inverter. The costs for laying a cable on edge e is $len(e)$. For the decision variant of this problem an additional parameter $K \in \mathbb{N}$ is given. Is there a subset of edges

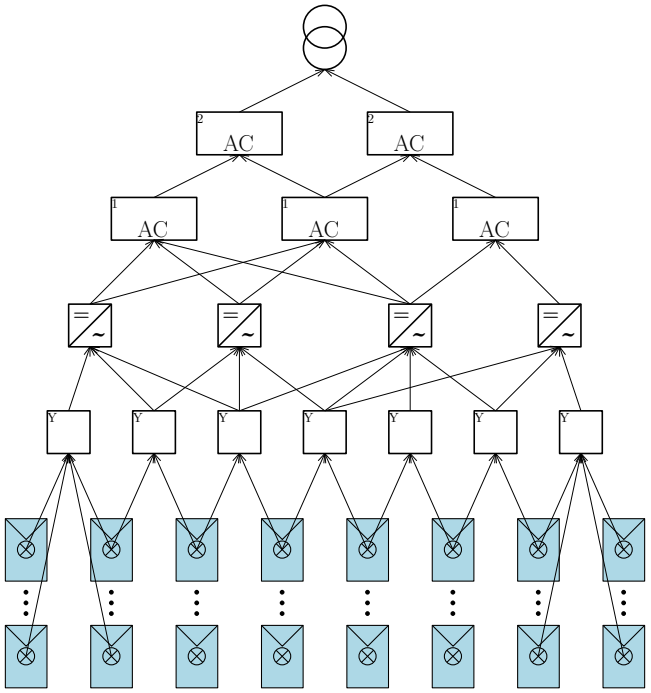


Figure 4.4.: Illustration of the solar farm graph for string inverters.

$E' \subseteq E$ with $\sum_{e \in E'} len(e) \leq K$ such that each combiner box is connected to exactly one inverter without exceeding the inverter capacity i_{max} and without falling below i_{min} for each connected inverter.

Every subproblem can be easily transformed into a complete solar farm cabling problem. For this we just add the missing layers to the graph and add the edges in a way that the cables and the flow for the added layers is clear. We take the two-layer cabling problem for an example. First we insert a transformer and connect every inverter to it with an edge of length 0. Then we add recombiner boxes and Y-connectors. As in the two-layer cabling problem the combiner boxes V_C are directly in front of the inverters, we can also view them as recombiner boxes. So we insert a combiner box and a Y-connector for every $v \in V_C$, connect the combiner box to the recombiner box and the Y-connector to the combiner box by an edge. Now we need to connect $p(v)$ strings to the corresponding Y-connector of each $v \in V_C$. The result is a graph as defined above for the complete solar farm problem, but the cabling problem itself is the same as the two-layer cabling problem.

4.1.8. Edges of String Connection Points

Before we go into the analysis of the cabling problem we first make a simple observation regarding some edges starting at the string connection points. Until now we have not specified any conditions for the edges between two layers, so it is possible for two layers to be fully connected. Some edges between the string connection points and the vertices of the layer above (usually Y-connectors) however can be deleted without changing the problem.

Lemma 4.2. *Let s be a string, $v, v' \in V^s_{con}$ and $c \in V_Y$ a Y-connector such that there are two edges (v, c) and (v', c) with $len((v, c)) \leq len((v', c))$. Then there is an optimal solution which does not use edge (v', c).*

Proof. There are two edges $(v, c), (v', c) \in E$ with $v, v' \in V^s_{con}$ connection points of string s and $len((v, c)) \leq len((v', c))$. Let H be an optimal solution. There are two possible

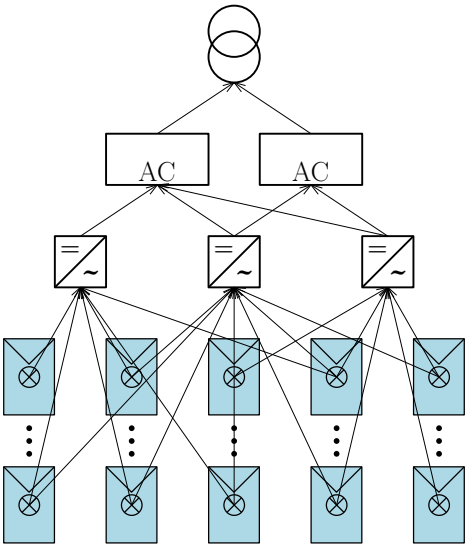


Figure 4.5.: Example for the simple string inverter problem with only one layer of combiner boxes.

cases: H assigns a cable to edge (v', c) or H does not assign a cable to (v', c). If H does not assign a cable to edge (v', c) the lemma is already fulfilled. So we only consider the first case that H assigns a cable to (v', c). We construct a solution H' that assigns a cable to (v, c) instead of (v', c), but otherwise is the same as H. This is possible as both H and H' direct the single unit of flow produced by string s to vertex c. Obviously the only possible cost difference of H' and H is because of using (v, c) instead of (v', c). As $len((v, c)) \leq len((v', c))$ the costs for H' cannot be higher than the costs for H. That means that H' is an optimal solution which does not use edge (v', c) and thus the lemma is proven. □

Lemma 4.2 implicates that we can remove edges from string connection points before applying further algorithms to the graph. If there are two edges (v, c) and (v', c) as described in Lemma 4.2 we can remove edge (v', c) without increasing the costs of an optimal solution. With the following small algorithm we can delete such edges in polynomial time.

1. For all Y-connectors y and for all strings s check if there is more than one edge between connections points of s and y.

2. If yes, delete all edges except for one of the shortest edges.

The naive runtime of the algorithm above is in $O(\sum_{y \in V_Y} (\sum_{s \in S} |V^s_{con}|))$ as we loop over all Y-connectors y and then over all strings s for which we have to check all outgoing edges to the current Y-connector, which are at most $|V^s_{con}|$. We can simplify this term to $|V_Y| \cdot \sum_{s \in S} |V^s_{con}|$. Obviously $|V_Y| \in O(n)$ where n is the number of vertices. As the connection points are also part of the vertices $\sum_{s \in S} |V^s_{con}| \in O(n)$. In total we then can estimate the runtime of the algorithm as $O(n^2)$.

With Lemma 4.2 and our small algorithm we can transform every instance of a solar farm graph into an instance where every string has at most one connection point connected to a Y-connector y. That means if we want to direct flow from a string to a specific Y-connector we have only one edge we can use for this. We then can get rid of the connection points of a string and merge them into one single vertex per string. In this case the edge length would not necessarily be the Euclidean distance between the single connection point of the

string and the Y-connector, but the distance between the original connection point and the Y-connector. One should be aware of this fact especially when drawing a solar farm.

In the following we will only consider solar farm instances which have one connection point per string. That is why we will use string as equivalent term to connection point and we will view the set of strings as part of the vertices $S = V_{con} \subsetneq V$ in these instances.

5. NP-hardness

After defining the solar farm cabling problem we now want to analyze some of its properties especially its complexity. We actually consider different variants of the solar farm cabling problems and analyze all of them.

First we consider the solar farm cabling problem with special conditions which can be enhanced to the general solar farm cabling problem as defined in Section 4.1.6. On the one hand we consider the cabling problem foregoing different cable types and capacities and on the other hand the cabling problem where cable types are the main source of complexity. In both of these cases we do not distinguish between AC and DC cables, but assume that AC and DC cable types have the same capacity and costs. The same applies to other upcoming proofs if it is not stated otherwise. Second we consider the two-layer cabling problem as mentioned in Section 4.1.7 and then we show that it is hard to find any feasible solution for the general cabling problem. Third we go over to a special more realistic cabling problem where the positions of the vertices are in $\mathbb{R}^2$ and all layers are fully connected.

But now we want to show the NP-completeness of the general solar farm cabling problem respectively a special variant which implicates the NP-completeness of the general problem. For this we reduce the 3-satisfiability problem (3SAT) to a part of the solar farm cabling problem.

3SAT: There are a set of boolean variables $X = \{x_1, \ldots, x_n\}$ and a set of clauses $C = \{c_1, \ldots, c_m\}$. Each clause is a disjunction of exactly three literals, the negation of a variable or the variable itself. Is there an assignment of *true* and *false* to the variables such that the conjunction of all clauses evaluates to *true*?

As one of Karp's 21 NP-complete problems 3SAT is NP-complete [Kar72].

Theorem 5.1. *The solar farm cabling problem with a set of strings, a set of inverters, one transformer and a single type of cable, but no vertex or cable capacity restrictions is NP-complete.*

Proof. We take an instance of the 3SAT problem and transform it into a cabling problem.

We need a graph $G = (V, E)$ with V consisting of strings (with one connection point each), inverters and one transformer. For every variable $x \in X$ there are two inverters i_x and $i_{\neg x}$ and one string (with one connection point) $s_x \in V$. There is also one string $s_c \in V$ for every clause $c \in C$ and in total one transformer $t \in V$. The inverters and

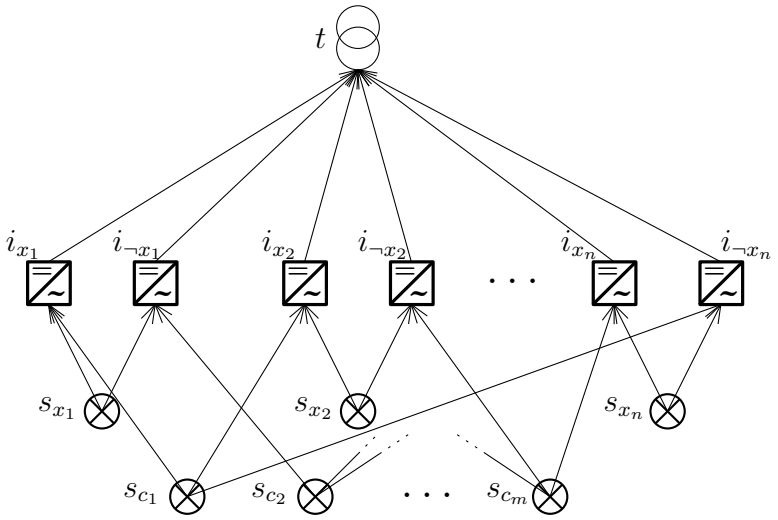


Figure 5.1.: Visualization of the construction in the proof of Theorem 5. An instance of 3SAT is transformed into a solar farm cabling problem. Clause c_1 of the corresponding 3SAT instance is $x_1 \vee x_2 \vee \neg x_n$, that is why string s_{c_1} is connected to the inverters i_{x_1}, i_{x_2} and $i_{\neg x_n}$.

the transformer have no capacity restrictions. Every inverter i_x or $i_{\neg x}$ is connected by an edge to the transformer and the corresponding string s_x. Every string s_c is connected to the three inverters representing the literals in clause c. So $E = \{(s_x, i_x), (s_x, i_{\neg x}) | x \in X\} \cup \{(s_c, i_x) | x \in c\} \cup \{(s_c, i_{\neg x}) | \neg x \in c\} \cup \{(i_x, t), (i_{\neg x}, t) | x \in X\}$ and every edge has length 1. A visualization of this construction is shown in Figure 5.1. Additionally we have one cable with unrestricted capacity and cost 1. This transformation is possible in polynomial time as the graph consists of $3n + m + 1$ vertices and at most $4n + 3m$ edges.

We now show that there is a solution of the 3SAT instance if and only if there is a solution of the cabling problem with costs of at most $2n + m$.

Given a solution of the 3SAT instance, we construct a solution of the cabling problem. If the value *true* is assigned to a variable x, we lay a cable between s_x and i_x and do not use $i_{\neg x}$. If x is *false*, we do it the other way round. For a feasible solution of 3SAT every clause has at least one *true* literal. Because of our construction this means that every string s_c is connected by an edge to an inverter of which the corresponding literal is *true*. We can then lay a cable along this edge. As for every variable either x or $\neg x$ is *true*, we only use half of the inverters. Each inverter is connected to t by an cable, which are then n edges with cables. Additionally each string is connected to one inverter by a cable. There are $n + m$ strings and so we have to use $n + m$ cables between the strings and inverters. Altogether we get a feasible solution for the cabling problem with costs of $2n + m$.

If we now take a solution of the solar farm cabling problem with at most $2n + m$ cost, we first observe that in this solution (and every other feasible solution) $n + m$ cables are needed to connect all the strings to the inverters, one cable for each of the $n + m$ strings. As each cable has a cost of 1, these cables together have costs of $n + m$. This leaves at most a cost of n for the connection between inverters and the transformer and means at most n edges between the inverters and the transformer may be used for cables. Of each pair of inverters i_x and $i_{\neg x}$ one inverter has to be used, as the corresponding string s_x can only be connected to one of these two inverters. So at least half of the inverters have to be connected to the transformer, which already results in n cables between inverters and transformer and therefore in costs of n. This means of each pair i_x and $i_{\neg x}$ exactly one inverter is connected with a cable because more would result in costs of more than $2n + m$.

If i_x is used, we set the variable x to *true*, if $i_{\neg x}$ is used we set x to *false*. Every string s_c is now connected to one inverter of which the corresponding literal is *true*. Otherwise the string would only be connected to inverters which are not connected to the transformer, but this would not be a feasible solution for the cabling problem. So every clause c contains a *true* literal and we have a solution for the 3SAT problem.

This shows that the solar farm cabling problem is NP-hard. Obviously a solution of the solar farm cabling problem can be checked for feasibility in polynomial time, so the problem is NP-complete. □

In the proof above inverters and transformers could of course be interchanged with combiner boxes without changing the proof. So it is not only this special subproblem with three layers, that is NP-complete. If you take three layers of strings, combiner boxes and inverters the problem is also NP-complete.

However in the proof above we do not really use cables as there is only one cable which is not even capacity restricted. We want to do another NP-hardness proof which only uses cables, but no vertex capacities and unit edge lengths.

For this proof we use the Multiple Subset Sum Problem (MSSP) [CKP00, p. 309]

Multiple Subset Sum : Let $K, c \in \mathbb{N}$. There is a set of items $N = \{1, \ldots, n\}$ of which each item i has a weight $w_i \in \mathbb{N}$ and a set of bins $B = \{b_1, \ldots, b_m\}$ which all have the capacity c. Can a subset $S \subseteq N$ of items be packed into the bins with S_j being the set of items in packed into bin b_j, such that for all bins $b_j : (\sum_{i \in S_j} w_i) \leq c$ and $\sum_{i \in S} w_i \geq K$?

MSSP is strongly NP-hard [CKP00]. There are some additional assumptions we can make for the MSSP. For every MSSP instance we can assume that $w_i \leq c$ for all $i \in N$. Otherwise there were items that could not be assigned to any bin. And we can assume $K > m$ because in the other case $K \leq m$ solving the MSSP would be simple. For every bin we pick an arbitrary item (if still one left) and assign it to the bin. If the sum of the weights of the assigned items is K or more, we have found a solution, otherwise no solution exists.

Theorem 5.2. *The solar farm cabling problem with unit edge lengths, no capacity restrictions at the vertices, but an arbitrary amount of cable types is NP-complete.*

Proof. We make a reduction from the MSSP to this special solar farm cabling problem. So we take an MSSP instance and construct a solar farm graph and corresponding cable types. In the graph we construct we do not model strings explicitly, but directly start with combiner boxes which produce units of flow. These combiner boxes are implicitly connected with as many strings as units of flow they produce (as described in Section 4.1.7).

For each item $i \in N$ the solar farm graph has one combiner box c_i with a production of w_i units of flow and one combiner box $\bar{c_i}$ with $c + 1$ units of flow. There is one inverter v_b for each bin $b \in B$ and one inverter v_i for each item $i \in N$. The combiner boxes c_i are connected to all inverters v_b. Additionally each c_i is connected to its corresponding inverter v_i. A combiner $\bar{c_i}$ is only connected to its corresponding inverter v_i and to no other inverter. All inverters are connected by an edge to a single transformer t. As stated in the theorem all edges have length 1 and neither the transformer nor the inverters have a capacity. The solar farm graph obtained by this construction can be seen exemplarily in Figure 5.2. Now we need to specify the cable types. Let $G = \sum_{i \in N} w_i$. There is one cable k_1 with capacity c and cost 1, one cable k_c with capacity $c + 1$ and cost G and for each w_i for $i \in N$ there is a cable k_{w_i} with capacity $c + 1 + w_i$ and cost $G + w_i$. We obtain a graph with $3n + m + 1$ vertices and at most $n + 1$ cable types, so the construction is possible in

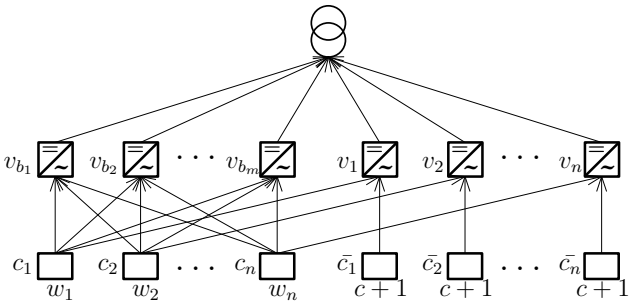


Figure 5.2.: Visualization of the construction of a solar farm graph out of a MSSP instance with n items and m bins like done in the proof of theorem 5.2. All edges have unit length.

polynomial time. Even if we added the strings the construction would still be polynomial as MSSP is strongly NP-hard.

The question for the constructed cabling problem instance is: Is there a cabling solution with costs of at most $2nG + n + m + G - K$? We show that this solution exists if and only if there is a solution of the MSSP instance with a weight of at least K.

Given a solution of the MSSP instance, we construct a solution of the cabling problem. If an item i is assigned to bin b, combiner box c_i is connected with cable k_1 to inverter v_b. If item i is not assigned to any bin c_i is connected to v_i with cable k_1. Each combiner box $\bar{c_i}$ can only be connected to its corresponding inverter v_i. Now each inverter is connected to the single transformer with an appropriate cable, that is the cheapest cable with enough capacity.

What are the costs of a so constructed solution? The connection of all combiner boxes to an inverter costs $n + n \cdot G$ as there are n combiner boxes with less than $c + 1$ units of flow for which cable k_1 is sufficient and n boxes for which we need cable k_c. Each inverter v_b has at most c incoming units of flow because c is the bin capacity and this must not be exceeded in a solution of the MSSP. Connecting the inverters v_b to t therefore results in costs of m. The interesting part is the connection of the inverters v_i to t. Each v_i has at least $c + 1$ incoming units of flow and if the corresponding item i is not assigned to any bin, it has additional w_i incoming units of flow. In the first case the cheapest cable to connect v_i to t would be k_c and in the second case k_{w_i}. The total costs for all v_i then are $n \cdot G + \sum_{i \in N \setminus S} w_i$ which results in total cabling costs of:

$$n + n \cdot G + m + n \cdot G + \sum_{i \in N \setminus S} w_i \leq 2nG + n + m + G - K$$

Now we start with a cabling solution with costs of not more than $2nG + n + m + G - K$ and construct a MSSP solution with a weight of at least K. Each item i whose corresponding combiner box is connected to inverter v_b is assigned to bin b. This assignment results in a solution of the MSSP instance and that is now shown.

We first show that a cabling solution with costs of at most $2nG + n + m + G - K$ does not assign more than c units of flow to an inverter v_b. Assume that an inverter v_b has more than c incoming units of flow. The cost of connecting the combiner boxes to the inverters always is $n + nG$. Connecting the inverters v_i costs at least nG as all of them need at least cable k_c. For v_b we also need at least cable k_c, which has costs of G as we assumed that v_b has at least $c + 1$ incoming units of flow. In total the costs would be

$$2nG + n + G > 2nG + n + m + G - K$$

because $K > m$. We now know that no inverter v_b has more than c incoming units of flow. That means that the assignment of items to bins we made above does not exceed bin capacities.

What remains to be shown is that this constructed solution has a weight of at least K. For this we again take a look at the cabling costs. Let $S \in N$ be the set of items assigned to bins. The cabling costs are n for the combiner boxes c_i, nG for the combiner boxes $\bar{c_i}$, at most m for the inverters v_b and $nG + \sum_{i \in N \setminus S} w_i$ for the inverters v_i. In total we get

$$
\begin{aligned}
& 2nG + n + m + \sum_{i \in N \setminus S} w_i \leq 2nG + n + m + G - K \\
\Leftrightarrow & \sum_{i \in N \setminus S} w_i \leq G - K \\
\Leftrightarrow & \sum_{i \in S} w_i \geq K
\end{aligned}
$$

which means we constructed a solution for the MSSP instance with a weight of at least K.

As we can check for a given solution of the cabling problem in polynomial time whether cable capacities are exceeded and the cost is at most a given value, the considered solar farm cabling problem is NP-complete. □

We have now two different proofs more or less both showing the same statement. The interesting thing is that in the first proof we showed that the solar farm cabling problem is NP-hard without considering different cable types. In the second proof we only used different cable types to show NP-hardness, but no capacities at the vertices. In the general solar farm cabling problem therefore two NP-hard problems overlap.

5.1. NP-hardness of other subproblems

We can go even a step further and consider only two layers of the solar farm cabling problem, the last layer of combiner boxes and the inverters. One could imagine that the power generated by the strings is already at the combiner boxes, so the cabling between strings and combiner boxes has already been done and now the cheapest cabling between combiner boxes and inverters has to be found. We called this problem the two-layer cabling problem (see Section 4.1.7). One can show that even this subproblem is NP-hard by a reduction of the Multiple Subset Sum Problem.

Theorem 5.3. *The two-layer cabling problem consisting only of one layer of combiner boxes and one layer of inverters is strongly NP-hard.*

Proof. We transform an instance of MSSP, for which we make the same assumptions as in Theorem 5.2, into an instance of the two-layer cabling problem as described above.

We build a graph $G = (V, E)$ with two layers. The first layer are the combiner boxes. For each item $i \in N$ we have a combiner box $c_i \in V$ with a production $p(c_i) = w_i$. The second layer are inverters. For each bin $b_i \in B$ we have one inverter $v_{b_i} \in V$ and one inverter $v_i \in V$ for each item $i \in N$. All inverters have the capacity $i_{max} = c$ and no restriction for a minimum current, so $i_{min} = 0$. Every combiner box is connected by an edge to all inverters $v_{b_i} \in V$ with $b_i \in B$. All these edges have length 1. Additionally each combiner box c_i is connected to the inverter v_i for $i \in N$. These edges have length $w_i + 1$. To connect combiner boxes with the inverters, there is one cable with unrestricted capacity and cost 1.

A visualization of this construction can be seen in Figure 5.3. As the number of vertices in the constructed graph is $2n + m$, the construction is possible in polynomial time.

We now fix $K \in \mathbb{N}$ and show that a solution for MSSP exists if and only if there is a solution for the cabling problem constructed above with cabling costs of not more than $G - K + n$, where $G = \sum_{i \in N} w_i$.

A solution of MSSP assigns a subset $S \subseteq N$ of items to the bins with $\sum_{i \in S} w_i \geq K$. For the corresponding instance of the cabling problem we connect every combiner box c_i to the inverter v_{b_j} by a cable if item $i \in S$ is assigned to bin b_j. As no bin capacity is exceeded in the solution of MSSP, no inverter capacity is exceeded. All other combiner boxes c_i are connected to their corresponding inverter v_i via the edge with costs of $w_i + 1$. Again no capacity is exceeded as $w_i \leq c$ for all $i \in N$. The connections costs in total then are

$$\sum_{i \in S} 1 + \sum_{i \in N \setminus S} (w_i + 1) = n + \sum_{i \in N \setminus S} w_i \leq G - K + n$$

which is a feasible solution for the cabling problem.

Now we take a solution for the cabling problem with maximal costs of $G - K + n$ and construct a solution for MSSP. We assign item $i \in N$ to bin b_j if combiner box c_i is connected to inverter v_{b_j} by a cable. These items together form the subset S. Items i of which the corresponding combiner box is connected to v_i are not part of S. As each combiner box i has w_i units of flow which need to be conducted to the inverters and no inverter capacity is exceeded, no bin capacity is exceeded either by this assignment. Otherwise the cabling solution would not be correct. We still need to show $\sum_{i \in S} w_i \geq K$. The cost for the cabling is calculated as above:

$$\sum_{i \in S} 1 + \sum_{i \in N \setminus S} (w_i + 1) = n + \sum_{i \in N \setminus S} w_i$$

Because the cost for the cabling solution is at most $G - K + n$, it follows that

$$\begin{aligned} & n + \sum_{i \in N \setminus S} w_i \leq G - K + n \\ \Leftrightarrow & \sum_{i \in N \setminus S} w_i \leq G - K \\ \Leftrightarrow & \sum_{i \in S} w_i \geq K. \end{aligned}$$

Hence our constructed solution for MSSP is feasible.

Altogether this shows that the subproblem consisting only of two layers of vertices is strongly NP-hard. □

Observation 5.4. *In the proof of theorem 5.3 we did not make any restrictions regarding the units of flow produced by the combiner boxes. If we take the complete solar farm cabling problem the values w_i would be restricted by the number of strings, that means restricted by a polynomial in the input length. But even with this restriction the two-layer cabling problem stays NP-hard as MSSP is a strongly NP-hard problem.*

In the proof above we could interchange the layer of inverters with a layer of recombiner boxes without any difficulty and still get the same result. So the layers of the two-layer problem do not need to be combiner boxes and inverters to make the problem NP-hard.

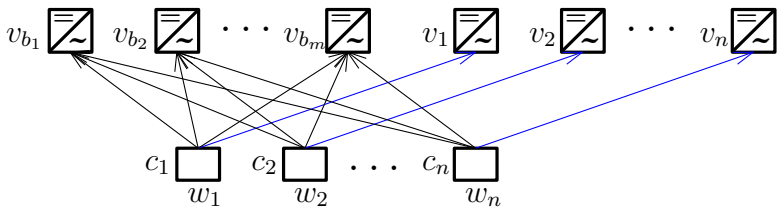


Figure 5.3.: Visualization of the construction in the proof of theorem 5.3. An MSSP instance with n items and m bins is transformed into a instance of the two-layer cabling problem. The blue edges have length $w_i + 1$, all other edges length 1.

5.2. Finding any solution

The preceding proofs show that it is hard to find a solution of the solar farm cabling problem that has maximum costs of some given value. But actually one can show that for the general case of the solar farm cabling problem it is NP-hard to find any feasible solution without caring about the costs of the solution. That means for the following proof we neither consider edge lengths nor cable costs.

First we need the NP-hard problem 3-Partition.

3-Partition: Let $S = \{s_1, \ldots, s_{3m}\}$ be a (multi)set with $s_i \in \mathbb{N}$ and $T \in \mathbb{N}$ with $\sum_{s \in S} s = mT$. Additionaly for each $s \in S$: $\frac{T}{4} < s < \frac{T}{2}$. Can S be partitioned into m triplets $S_1, \ldots S_m$ such that for $1 \leq i \leq m$: $\sum_{s \in S_i} s = T$?

3-Partition is strongly NP-hard [BRG89].

Theorem 5.5. *It is NP-hard to find any feasible cabling for the two-layer cabling problem.*

Proof. Starting with an instance of 3-Partition we construct an instance of the two-layer cabling problem. In this proof we are only interested in finding any solution. As we do not need to consider costs for that, we do not specify the edge lengths and the cable costs.

For each item $s \in S$ there is one combiner box c_s with $p(c_s) = s$. Remember that $\frac{T}{4} < s = p(c_s) < \frac{T}{2}$. There are m inverters which each have a maximal capacity of T. All combiner boxes are connected with all inverters by an edge. Additionally there is one cable with unrestricted capacity. We are only interested in finding any cabling that connects all combiner boxes to the inverters without exceeding inverter capacities. The question for our cabling problem therefore is: Is there a cabling (with arbitrary costs) that assigns combiner boxes to inverters without exceeding inverter capacities? As the constructed two-layer graph has $4m$ vertices, the construction is possible in polynomial time.

We start with a solution of the 3-Partition instance. For each triplet S_i we connect the three corresponding combiner boxes to one inverter. As there are m inverters each triplet of combiner boxes is connected to a different inverter. The sum of each triplet is T so the amount of flow reaching every inverter is T as well and therefore no inverter capacities are exceeded. Altogether we constructed a feasible cabling for the two-layer cabling problem.

Now we construct a solution of the 3-Partition problem with the help of a given solution of the two-layer cabling problem. A feasible solution of the two-layer instance constructed above always assigns exact T units of flow to each inverter. That is because the total capacity of all inverters is mT which is the same as the sum of the units of flow of all combiner boxes. T units of flow cannot be supplied by only two combiner boxes as $2 \cdot p(c) < 2\frac{T}{2} = T$. That means that each inverter needs three combiner boxes connected to it to reach T units of flow. But if we are now given a solution we can simply build one triplet S_i for each inverter i which consists of the items corresponding to the combiner

boxes connected to inverter i. As there are m inverters we get exactly m triplets and we already argued that the sum of flow at each inverter is T, so the sum of each triplet is also T. The so constructed triplets S_i are a solution of the 3-Partition instance.

In total we then get that it is NP-hard to find a feasible cabling for the two-layer cabling problem. □

By further analyzing the proof we we can see that finding a feasible cabling (with arbitrary costs) for the general solar farm cabling problem is NP-hard.

Corollary 5.6. *Finding a feasible cabling with arbitrary costs for the general solar farm cabling problem is NP-hard.*

The construction in the proof assigns each combiner c_s box some units of flow $p(c_s) = s$. In the general cabling problem these units of flow originate at the strings. That means we need one string for each unit of flow. More precisely we need s strings that are connected to combiner box c_s by an edge and not connected to any other combiner box. We already described this construction in Section 4.1.7. For the proof above we actually need 3-Partition to be strongly NP-hard, so 3-Partition is still NP-hard if T is bounded by a polynomial in the input size. Then each item $s \in S$ is bounded by a polynomial too. The construction including the strings then is still possible in polynomial time.

Theorem 5.5 also leads to another observation.

Corollary 5.7. *There is no polynomial-time approximation algorithm with any approximation guarantee for the general solar farm cabling problem (unless P=NP).*

A polynomial-time approximation algorithm would always find a feasible cabling for the solar farm cabling problem if such a cabling existed. As we have shown, finding a feasible cabling is NP-hard and such an algorithm would therefore show P=NP. Therefore we will now consider the cabling problem with an additional constraint.

5.3. Vertices in $\mathbb{R}^2$

A real life solar farm is usually built on a plane or at least an area that comes close to a plane. Because of this and because this might make the problem approximable in polynomial time we want to make the further assumption that all vertices of a solar farm graph are placed on the $\mathbb{R}^2$ plane. The question then is: Does this make the solar farm cabling problem easier?

The answer to this question is "no" in a sense that the problem stays NP-hard even when restricting vertex positions to $\mathbb{R}^2$ and the distances between vertices to the Euclidean distance. In this problem each vertex of a layer can be connected by a cable with any vertex of the next layer, the layers are fully connected.

For proving this statement we again need the 3-Partition problem (5.2).

Theorem 5.8. *The solar farm cabling problem where vertex positions are in $\mathbb{R}^2$, all layers are fully connected and the edge lengths are the Euclidean distance is NP-hard.*

Proof. To prove this Theorem we try to adapt the proof of Theorem 5.5. For this we make the same construction, but as the layers are now fully connected we need to make sure

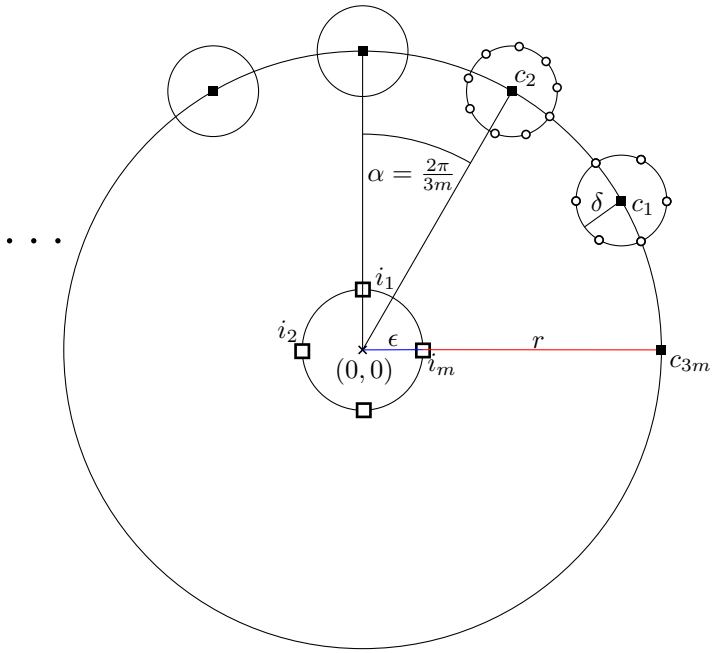


Figure 5.4.: The figure shows an exemplary sketch of the construction made in the proof of Theorem 5.8. The filled black squares are combiner boxes, the other squares in the middle are inverters and the small circles strings. Implicitly each string is connected to each combiner box and each combiner box to each inverter by an edge with the Euclidean distance as edge length. The so constructed problem of the two-layer cabling problem is NP-hard.

that the strings have to be connected to certain combiner boxes by positioning strings and combiner boxes very near to each other.

We start with a 3-Partition instance and construct a solar farm graph consisting of three layers: strings, combiner boxes and inverters. We arrange the inverters in a circle with small radius and the combiner boxes in a circle with larger radius around the center $(0,0) \in \mathbb{R}^2$. Then we place as many strings around one combiner box as units of flow we want to have at this box. The following construction is visualized in Figure 5.4. In detail we place m inverters $i_1 \dots i_m$ on a circle with radius $\epsilon > 0$ around $(0,0)$. The inverters all have a capacity of T. The second circle of combiner boxes has a radius $r + \epsilon$ and we place one combiner box c_j for each item $s_j \in S$ such that the combiner boxes are equally distributed on the circle. ϵ and r will be specified later. All combiner boxes have capacity $\left\lfloor \frac{T-1}{2} \right\rfloor$ and for each combiner box c_j s_j strings are placed around the box in a circle with radius $\delta > 0$. Remember that $s < \frac{T}{2}$, so it is always possible to connect s_j strings to a combiner box. Each string is connected with each combiner box and each combiner box with each inverter by an edge. As already stated in the Theorem, the edge lengths are the Euclidean distances between the vertices. Additionally there is one cable with unrestricted capacity and costs 1.

The constructed graph has $4m + mT$ vertices. As 3-Partition is a strongly polynomial problem T is bounded by a polynomial and so is $4m+mT$. Therefore the graph construction is possible in polynomial time. Now we still need to specify r, ϵ and δ. We choose $\epsilon < \frac{1}{6m}$ so we get $3m \cdot 2\epsilon < 1$. For r and δ we choose

$$r = \frac{1}{2 \cdot \sin(\frac{\pi}{3m})} \quad \text{and} \quad \delta = (r+\epsilon) \cdot \sin\left(\frac{\pi}{3m}\right) - \frac{1}{2} = \epsilon \sin\left(\frac{\pi}{3m}\right) > 0$$

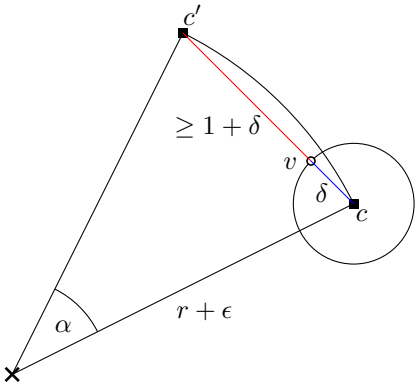


Figure 5.5.: In the proof of Theorem 5.8 we choose r and δ such that the distance of string v to combiner box c' is at least $1+\delta$ (red line). The red and the blue line together are the chord of a circular segment with angle α and radius $r+\epsilon$.

basically because we want to clearly separate the string circles from each other. Later this will get clearer when we actually use r and δ. Now we want to show that there is a solution for the 3-Partition instance if and only if there is a solution for the constructed cabling problem with costs lower than $3mr + 1 + mT\delta$.

First we take a solution of the 3-Partition problem and construct a solution of the cabling problem. Independently of the 3-Partition solution we connect the strings around a combiner box c_j to this combiner box, so that there are s_j units of flow at combiner box c_j. In total this results in costs of $mT\delta$ as there are mT strings. This is also possible because each combiner box has enough capacity as already argued above. For each triple S_i we then connect the corresponding three combiner boxes to one inverter. Each triple of combiner boxes has exactly T units of flow so no inverter capacity is exceeded and obviously there are enough inverters as there are $3m$ combiner boxes and m inverters. The distance of a combiner box to an inverter is at most $r+2\epsilon$. Connecting all combiner boxes then results in costs of at most $3m(r+2\epsilon)$. All in all the costs are at most $3m(r+2\epsilon)+mT\delta < 3mr+1+mT\delta$ and we obtain a solution of the cabling problem.

Now we solve the 3-Partition instance by using a given solution of the cabling problem. The cabling solution has costs lower than $3mr + 1 + mT\delta$.

First of all we argue that in such a solution each combiner box has to be used. That means each combiner box is connected by a cable to an inverter. Why is that the case? Each inverter has a capacity of T and there are m inverters. As there are mT strings, each inverter has to take exactly T units of flow. The maximum amount of flow one combiner box can provide is $\left\lfloor \frac{T-1}{2} \right\rfloor < \frac{T}{2}$ because of their capacity constraints. It follows from this that at least three combiner boxes are needed to provide T units of flow to one inverter. There are m inverters and $3m$ combiner boxes, so each combiner box is actually used. The cabling of combiner boxes to inverters has costs of at least $3mr$ as r is the distance of the inverter circle to the combiner box circle.

With this in mind we now argue that each string is connected to its nearest combiner box. So each circle of strings is connected to the combiner box in their middle. For this argument we need the lower bound for the distance of a string v in the circle around a combiner box c to another combiner box c'. This lower bound is $1+\delta$ and we show that now.

All combiner boxes are placed on a circle with radius $r+\epsilon$ around the center. They are all equally distributed along the circle so the distance of two combiner boxes next to each other is given by the length of the chord of the circular segment with radius $r+\epsilon$ and angle

$\frac{2\pi}{3m}$. You can see that exemplary in Figure 5.5. In general the length of the chord with radius t and angle α is given by $2t \cdot \sin\left(\frac{\alpha}{2}\right)$. The distance d of two combiner boxes next to each other then is $d = 2(r+\epsilon) \cdot \sin\left(\frac{\pi}{3m}\right)$. This is also a lower bound of the distance of any two combiner boxes c and c'. The distance of string v to combiner box c is δ, so connecting v to c' results in costs of at least

$$
\begin{aligned}
&2(r+\epsilon) \cdot \sin\left(\frac{\pi}{3m}\right) - \delta \\
&= 2r \sin\left(\frac{\pi}{3m}\right) + 2\epsilon \sin\left(\frac{\pi}{3m}\right) - \epsilon \sin\left(\frac{\pi}{3m}\right) \\
&= 2r \sin\left(\frac{\pi}{3m}\right) + \epsilon \sin\left(\frac{\pi}{3m}\right) \\
&= 2\left(\frac{1}{2 \cdot \sin(\frac{\pi}{3m})}\right) \cdot \sin\left(\frac{\pi}{3m}\right) + \delta \\
&= 1 + \delta.
\end{aligned}
$$

With the lower bound of $1+\delta$ now shown, we now show that connecting string v to a combiner box c' instead of c is too expensive for a solution of the constructed cabling problem.

We already determined that connecting the combiner boxes to the inverters at least costs $3mr$. Connecting all other $mT-1$ strings besides v to a combiner box has costs of at least $(mT-1) \cdot \delta$ and as we just calculated, the cost for connecting v to c' is at least $1+\delta$. So if in any solution of the cabling problem a string is not connected to its nearest combiner box the resulting costs would be at least

$$3mr + (mT-1) \cdot \delta + 1 + \delta = 3mr + mT\delta + 1.$$

But this would not be a solution for the constructed cabling problem as the costs have to be strictly lower than $3mr + mT\delta + 1$. We then can conclude that in a solution which fulfills the cost condition all strings are always connected to their associated combiner box. That means the combiner box whose distance to the string is δ.

As a short recapitulation we now know that in a solution of our constructed cabling problem each combiner box has to be used and the strings in a circle around each combiner box actually have to be connected to this box. A combiner box c_j therefore provides exactly s_j units of flow. With this information we now construct the solution of the 3-Partition instance.

For each inverter $i = 1 \dots m$ we build a triple S_i. Each inverter is connected to three combiner boxes, as already argued above. The three corresponding items $s \in S$ form one triple S_i as their sum is exactly T (also shown above). With m inverters we get m triples and a correct solution for the 3-Partition instance. □

6. Exact Solving

After defining the problem and showing the NP-hardness for several variants we want to propose some algorithms which solve the solar farm cabling problem exactly or approximately. First we show a dynamic program and an mixed integer linear program which return optimal solutions of the cabling problem. In the next chapter we will then consider a heuristic algorithm.

6.1. Dynamic Program

We first present a dynamic program which returns an optimal solution for the solar farm cabling problem. For this we need the terms *configuration* and *layer* as we use them in the following. The layers of a solar farm graph are S, V_Y, V_C, V_R, V_I and V_T, where S is the bottom or first layer and V_T is the top or last layer. The dynamic program could handle an arbitrary amount of layers and therefore we now use generic layer names V_i.

Definition 6.1. *A* configuration *of a layer V is a tuple $(p_1, ..., p_{|V|}), p_i \in \mathbb{N}_0$, where p_i is the amount of flow at vertex $v_i \in V$. The total flow in one layer equals the number of strings: $\sum_{i=1}^{|V|} p_i = s$, where s is the number of strings.*

A configuration is feasible *if no p_i violates the maximum or minimum flow values for its vertex.*

The minimal cost *for a configuration of layer V is the minimal cost of all feasible cablings until layer V such that as many units flow are directed to each vertex in V as given by the configuration.*

Observation 6.2. *If we take a configuration p of a layer V_i and a cabling c between V_i and the next layer V_j, p and c together define a configuration of layer V_j. We say that the configuration p and the cabling c* result *in a configuration of the next layer V_j. For an arbitrary p and c the resulting configuration might not be feasible of course.*

Now we come to our dynamic program. Algorithm 6.1 basically takes configurations of a layer V_i and cablings to the next layer V_j to produce a partial solution of the cabling problem up to layer V_j. More precisely the dynamic program considers every layer of vertices, except for the first layer and every cabling between all configurations of the previous layer and the current layer (Line 5 - Line 7). We then check whether the current

combination of configuration and cabling results in feasible configuration of the current layer i (Line 8). In the case that the costs of the previous configuration and the cabling is lower than the costs of the configuration of layer i the costs of the configuration of layer i are updated (Line 12). So for each configuration the combination of cabling and predecessor configuration with the lowest cost is stored. To get the solution we find the configuration of the last layer with minimal costs. As we stored a predecessor configuration for every configuration of each layer we can then put together the solution (Line 13 - 17).

Algorithm 6.1 uses generic layers names V_i. For a solar farm graph we would then have $V_1 = S$, $V_2 = V_Y$ and so on. So we can use the dynamic program for solar farm graphs without problems.

Algorithm 6.1: Dynamic Program

Input: solar farm $G = (V, E)$, with layers $V_1, \ldots, V_k$
Output: assignment of cables to edges with minimal costs

```
1  costs_i = array of costs for configurations of layer i, initialized with ∞ for each
     configuration
2  cabling_i = array of cablings for configurations of layer i
3  if there is only one layer V_1 then
4      return ∅
5  for i ∈ {2, ..., k} do
6      forall feasible configurations pre_config of layer V_{i-1} do
7          forall cablings s between V_{i-1} and V_i do
8              if pre_config and s result in a feasible configuration c of V_i then
9                  if cost(s)+ costs_{i-1} [pre_config] < costs_i [c] then
10                     predecessor (c) ← pre_config
11                     cabling_i [c] ← s
12                     costs_i [c] ← cost(s)+ costs_{i-1} [pre_config]
13 config ← configuration of the lowest costs in costs_k
14 do
15     solution ← solution ∪ cabling_i [config ]
16     config ← predecessor (config)
17 while config is not the configuration of layer 1
18 return solution
```

Theorem 6.3. *Algorithm 6.1 returns an optimal solution of the solar farm cabling problem.*

Proof. We first show that the algorithm actually returns a feasible solution in a sense that no cable or vertex capacities are exceeded. And second we show by induction the more general statement that the dynamic program calculates the lowest costs for every possible configuration of the last layer. If we then take the cheapest among these configurations, we get the optimal solution for the whole solar farm.

The algorithm only considers feasible configurations which means no vertex capacities are exceeded. The same applies to the cablings between to layers as the cheapest cable for a given amount of flow (given by the configuration) is uniquely determined. So no cablings which connect cables with too low capacity to vertices are considered. At the end we get a solution which does not violate any capacity constraints and is therefore correct.

Second we show by induction that the algorithm calculates the minimal costs for every possible configuration of the last layer. The base case of our induction is one layer. The

first layer of a solar farm is the layer of the strings, so we have exactly one configuration for this layer. As there is no cabling for a single layer, the minimal cost for the configuration is obviously 0 and the dynamic program indeed does not return a cabling.

Our induction hypothesis is that the algorithm calculates the minimal costs and the corresponding cabling for every configuration of layer $n-1$. Now we want to show that it then also calculates the minimal costs for each configuration of layer n. So the algorithm considers the configurations of layer $n-1$. For each configuration all cablings which result in a feasible configuration c of layer n are taken into account (Line 8). As all combinations of configurations of layer $n-1$ and cablings between layer $n-1$ and n are tried out, we will reach all possible feasible configuration of layer n. Other feasible configurations of layer n cannot be reached by any legal cabling. These configurations if they exist then have infinite costs. But for the configurations c of layer n which are possible, all combinations of configuration of layer $n-1$ and cabling that result in c have been tried out and we only store the cheapest one. So the algorithm calculates a minimum cost cabling for all configurations of the layer n.

An optimal solution of the solar farm cabling problem includes one of all the possible configurations of the last layer. The algorithm has calculated the solution with minimal costs for each configuration of the last layer. Hence, if we pick the configuration of the last layer with the lowest costs we get an optimal solution for the solar farm cabling problem. □

Although it is nice to have an algorithm that solves the solar farm cabling problem exactly, we have not spoken about the runtime of the algorithm yet. We have already shown that the solar farm cabling problem is NP-hard so it would be pretty surprising if the runtime was polynomial.

Lemma 6.4. *The runtime of Algorithm 6.1 is in* $O\left(\lceil \frac{n}{2} \rceil^{\lfloor \frac{n}{2} \rfloor} \cdot \binom{n-1}{\lceil \frac{n-1}{2} \rceil} \cdot T\right)$*, where* T *is the time to update costs and predecessor of a configuration.*

Proof. First we calculate the number of configurations of a layer. Of course there is only one configuration for the layer of strings. Let n_i be the number of vertices of a layer i and s be the number of strings and $n_1 = s$. If we do not take capacities into account, there are $\binom{n_i+s-1}{s}$ configurations for layer i. This is the number of possibilities to distribute s units of flow to n_i vertices.

The number of cablings between a layer i and a layer $i-1$ with n_i and n_{i-1} vertices respectively is at most $n_i^{n_{i-1}}$. This is the case when the two layers are fully connected.

The algorithm now iterates over all layers. For each layer every configuration is considered and for these each cabling to the next layer. The number of combinations of configurations and cablings then is $\binom{n_{i-1}+s-1}{s} \cdot n_i^{n_{i-1}}$. Each of these combinations result in a configuration of layer i. Let T be the time to update a configuration (potentially new costs and predecessor). Hence we get the runtime $O\left(\binom{n_{i-1}+s-1}{s} \cdot n_i^{n_{i-1}} \cdot T\right)$ for one layer i. For all layers together we then get

$$\sum_{i=2}^{k} O\left(\binom{n_{i-1}+s-1}{s} \cdot n_i^{n_{i-1}} \cdot T\right).$$

The first layer in a solar farm are the strings and as every string produces exactly one unit of flow there is only one configuration for the first layer and the runtime reduces to

$$O(n_2^s \cdot T) + \sum_{i=3}^{k} O\left(\binom{n_{i-1}+s-1}{s} \cdot n_i^{n_{i-1}} \cdot T\right).$$

Of course the number of vertices of each layer together are n, so $\sum_{i=1}^{k} n_i = n$

In a solar farm graph like we defined in Section 4.1.6 there are at most $k = 6$ layers so we basically have six times the term $\binom{n_{i-1}+s-1}{s} \cdot n_i^{n_{i-1}} \cdot T$. We now want to find an upper bound estimation for this term. For $i > 1$ we can roughly estimate

$$\binom{n_{i-1} + s - 1}{s} \leq \binom{n-1}{s} \leq \binom{n-1}{\lceil \frac{n-1}{2} \rceil}$$

as $n_{i-1} + s$ is at most n and a binomial coefficient $\binom{n}{k}$ gets maximal for a fixed n if $k = \lceil \frac{n}{2} \rceil$ or $k = \lfloor \frac{n}{2} \rfloor$. Further we can estimate $n_i^{n_{i-1}} \leq \lceil \frac{n}{2} \rceil^{\lfloor \frac{n}{2} \rfloor}$ to get a very rough runtime estimation of

$$O\left(\left\lceil \frac{n}{2} \right\rceil^{\lfloor \frac{n}{2} \rfloor} \cdot \binom{n-1}{\lceil \frac{n-1}{2} \rceil} \cdot T \right).$$

In this estimation we now assumed the upper bound $\frac{n}{2}$ for s, n_i and n_{i-1}, which is of course impossible in case of three different layers $s, i-1$ and i. □

The runtime of Algorithm 6.1 is prohibitively slow. Even for small instances of solar farms the algorithm would take years to calculate a solution, which can be seen at the following example. A small solar farm could have about 50 strings and three combiner boxes for the second layer. That means we have 3^{50} possible cablings between the two layers. If we make the unrealistic assumption that we can list one cabling per nanosecond, listing the cablings would last $3^{50}\ ns \approx 7 \cdot 10^{23}\ ns \approx 21$ years. That is why we now consider a standard method to solve optimization problems: Linear Programming.

6.2. Linear Programming

In the case for solar farms we can formulate a mixed integer linear program (MILP). The MILP formulation does not make the cabling problem less NP-hard. In general MILPs are NP-hard as they are a generalization of integer linear programming which was shown to be NP-hard by Karp [Kar72]. But in practice there are very optimized solvers for MILPs, so we also give an MILP formulation of the solar farm cabling problem.

6.2.1. MILP formulation

We have variable $x_{ij} \in \mathbb{R}_{\geq 0}$ which states how much flow is directed from vertex i of layer V_l to vertex j of layer V_{l+1}. Variable $e_{ijc} \in \{0, 1\}$ states whether the edge connecting vertex i of layer V_l and vertex j of layer V_{l+1} is used for cable c. The MILP formulation below does not distinguish between AC and DC cables to make the formulation a bit clearer. The idea could then be adapted later. Let k be the number of layers of the solar farm graph and let V_R be layer before the inverter layer V_I.

$$\min \sum_{l=2}^{k} \sum_{i \in V_l} \sum_{j \in V_{l+1}} \sum_{c \in C} e_{ijc} \cdot cost(c) \cdot len(i, j) \tag{6.1}$$

$$s.t. \sum_{j \in V_2} x_{ij} = 1 \qquad \forall i \in V_1 \tag{6.2}$$

$$\sum_{i \in V_l} x_{ij} = \sum_{m \in V_{l+2}} x_{jm} \qquad \forall l \in \{1, \ldots, k-2\} \;\; \forall j \in V_{l+1} \tag{6.3}$$

$$\sum_{i \in V_l} x_{ij} \leq \max(j) \qquad \forall l \in \{1, \ldots, k-1\} \;\; \forall j \in V_{l+1} \tag{6.4}$$

$$\sum_{i \in V_R} x_{ij} \geq \min(j) \cdot \sum_{i \in V_l} \sum_{c \in C} e_{ijc} \qquad \forall j \in V_I \tag{6.5}$$

$$\sum_{j \in V_l} \sum_{c \in C} e_{ijc} \leq 1 \qquad \forall l \in \{1, \ldots, k-1\} \;\; \forall j \in V_{l+1} \tag{6.6}$$

$$\left(\sum_{c \in C} e_{ijc} \cdot cap(c) \right) \geq x_{ij} \qquad \forall i \in V_l \;\; \forall j \in V_{l+1}, \; l \in \{1, \ldots, k-1\} \tag{6.7}$$

$$x_{ij} \in \mathbb{R}_{\geq 0}, e_{ijc} \in \{0, 1\} \tag{6.8}$$

Equation (6.2) makes sure each string has an outflow of exactly one. Flow preservation is ensured by Equation (6.3). Equations 6.4 and 6.5 make sure that no vertex capacities are exceeded respectively no inverter lower bounds are undercut. With Equation (6.6) no vertex has more than one outgoing cable. Equation (6.7) has two different functions. First it makes sure no cable capacity is exceeded. Second we need to make sure that the flow exiting a vertex does this along the one outgoing cable. If for a vertex i $e_{ijc} = 1$ for any cable c, but $x_{iv} > 0$ for another vertex $v \neq j$, then the left side of Equation (6.7) would be 0 for i and v and therefore less than x_{iv}. Note that the variables x_{ij} are actually numbers from $\mathbb{N}_0$. That is as side effect of Equation (6.6).

7. Heuristic Solving

We now take a step back and first try to develop an algorithm that finds any solution without exceeding vertex or cable capacities. The problem here is that we already showed in Theorem 5.5 that for the general solar farm cabling problem it is NP-hard to find any feasible solution. Therefore we only consider solar farm graphs with fully connected layers like in Section 5.3 and at first we will ignore the lower bounds of inverters. However the algorithm we are going to develop is not limited to any kind of metric edge lengths. Again as in the previous chapter for the dynamic program, we use generic layer names where V_1 is the layer of strings.

Before we come to our algorithm we take a look at cables layouts. As already stated a cable layout is a forest of trees rooted in the transformers. We can make the observation that each cable layout consists of a number of paths starting at layer V_1 and ending in some layer V_i for $i \in 2, \ldots, k$. Each given layout can be decomposed into all these paths by recursively cutting out the longest path of each tree of the layout. If we then successively add the cables defined by the paths to the fully connected solar farm graph, we get the cable layout as result. Figure 7.1 shows an example of a decomposition of a tree into several paths.

The height of a path is the number of unconnected vertices at the time when the path is added to the cable layout. So e.g the first path added to an empty cable layout always has height k. It follows that a path with height i consists of a path of i unconnected vertices in the layers 1 to i and is connected to an already existing path in layer $i + 1$.

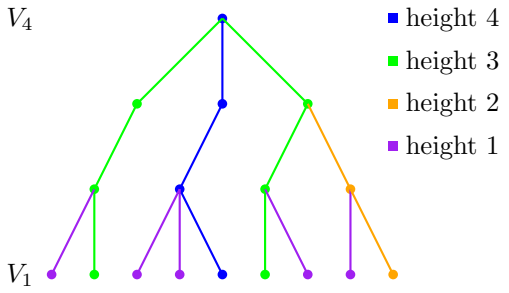


Figure 7.1.: The tree is decomposed into paths of different heights marked by the colors.

A *partial* cable layout is a layout where not all strings are connected to a transformer. That means only a part of the connections of the layout are already fixed. Examples for partial cable layouts can be seen in Figure 7.2.

Now we need one last definition before we explain our algorithm: *remaining capacities.*

Definition 7.1. *The* remaining capacity ρ *of a vertex* v *as part of a partial layout is defined depending on a reference layer* i*. The layer of vertex* v *has to be higher than* i*.*

The remaining capacity of a transformer t *is* $\rho(t) = \max(cap(t) - x_t \cdot c, 0)$*, where* $cap(t)$ *is the capacity of* t*,* x_t *is the number of vertices of layer* i *that are connected to* t *and* c *is the capacity of the vertices of layer* i*.*

If v *is not a transformer, the remaining capacity is the minimum of the remaining capacity of its parent vertex in the layout and* $\max(cap(v) - x_v \cdot c, 0)$*.* x_v *is the number of vertices of layer* i *that are connected to* v*.*

The remaining capacity of a vertex basically states how much flow can be directed through a vertex if all its previously connected child vertices and vertices of the same layer as the child vertices are filled up to their capacity. Now we can finally get to Algorithm 7.1.

The basic idea of Algorithm 7.1 is to connect the strings one by one to a transformer such that in each step as few opportunities for the strings still to be connected as possible are destroyed. For each string we try to connect the string to as many unconnected vertices as possible (Line 4). That means we first connect higher paths resulting in the cable layout being built from top to bottom in a sense that all vertices of a higher layer are already connected when we connect the last vertex of a lower layer. In the case that there are no more empty vertices in a layer, the strings are connected to the vertex with the most remaining capacity. If there is a tie of several vertices the amount of flow is considered as a tiebreaker.

Algorithm 7.1: String connection

```
   Input: solar farm graph G = (V, E) with layers V_1, ... V_k
   Output: cable layout for the given graph
1  forall strings s ∈ V_1 do
2      forall layers 2, ..., k do
3          if there is an empty vertex in this layer then
4              choose empty vertex
5          else
6              break
7      if last chosen vertex ∉ V_k then
8          with the following priorities choose the vertex with:
9          1. the most remaining capacity
10         2. the least amount of flow (if equal do this recursively for the parent
             vertices)
11         3. due to a fixed order
12     connect string to path of chosen vertices
13     if vertex capacities are exceeded then
14         return false
```

So there are basically three criteria how to connect a string to the top of the graph. The first criterion is to always choose an empty vertex if there are empty vertices left in a layer.

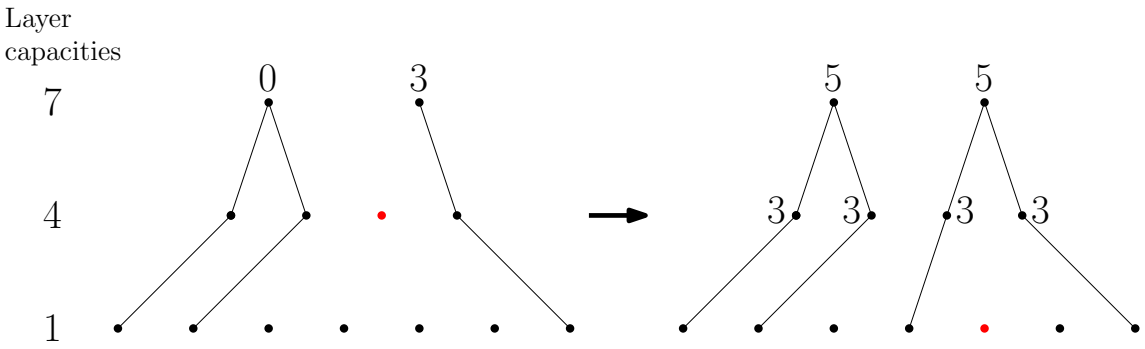


Figure 7.2.: The figure shows the remaining capacities of a partial cable layout. In the left graph the remaining capacities depend on the amount of child vertices of the middle layer, because the red vertex is not connected yet. In the right graph all vertices of the middle layer are already connected and the remaining capacity now depends on the bottom layer.

The second is to choose the vertex with the most remaining capacity. The third criterion is to choose the vertex with the least amount of flow. In the case that this still does not eliminate all but one vertex, the amount of flow of the corresponding parent vertices is compared.

Determining the amount of flow at a vertex is pretty simple. For the remaining capacity this is a bit more complicated, so we give an example of how the remaining capacity is calculated in Algorithm 7.1. In the left graph of Figure 7.2 there is one vertex of the middle layer unconnected, the red one. The remaining capacities of the left top vertex is calculated as $7 - 2 \cdot 4$ as it is connected to two vertices with capacity 4 of the middle layer. Because the remaining capacity cannot be negative, it is then set to 0. The right top vertex is only connected to one vertex with capacity 4 and so its remaining capacity is $7 - 4 = 3$. Now in the right graph there are no unconnected vertices of the middle layer anymore, so the remaining capacity of the vertices of top and middle vertices needs to be recalculated. Each of the middle vertices is connected to one bottom vertex. That means the remaining capacity for them is calculated as $4 - 1 = 3$. The top vertices are connected to two bottom vertices, so their remaining capacity is calculated as $7 - 2 \cdot 1 = 5$. As this is more than the remaining capacity of their child vertices, the remaining capacity of those does not need to be changed and stays 3.

Before we continue to work with Algorithm 7.1 we need to state that we are unfortunately not able to prove the correctness of the algorithm. That means if Algorithm 7.1 does not return a cabling we cannot show that there actually is no cabling that does not exceed vertex capacities (lower bounds are ignored). We believe that it works correctly though.

Conjecture 7.2. *Algorithm 7.1 calculates a cabling for a given solar farm graph that does not violate the vertex capacities iff such a cabling exists.*

We will now give some reasons and examples why this seems logical and to show what thoughts are behind the algorithm. Later when the algorithm is evaluated in practice we will come back to this. First we consider the first criterion in the algorithm, which always chooses an empty vertex if possible.

Algorithm 7.1 adds paths in an order of descending height to the cable layout, as we always choose unconnected (empty) vertices for a path if they exist in a layer.

Observation 7.3. *The amount of paths of height i in the output of Algorithm 7.1 is determined by $|V_i| - |V_{i+1}|$, with V_i being the vertices of layer i and V_{i+1} of layer $i+1$ respectively.*

For each path that uses an vertex of layer $i+1$ a vertex of layer i is needed. So for paths of height i we can only use the remaining vertices of layer i.

We now show that it is not worse to use longer paths in a cable layout if possible. This is also what Algorithm 7.1 does, as it always connects empty vertices of a layer if there are any left. So for each string a path of maximal height is connected to the existing partial layout.

Lemma 7.4. *A cable layout F can be transformed into a cable layout F' that consists of a maximal amount of paths of height i or more for $i = \{1, \ldots, k\}$ without reducing the capacity of F.*

Proof. We have already stated above that the maximal amount of paths of at least height i depends on the number of vertices in layer i. So if F does not consist of a maximal amount of paths of at least length i, that means that F does not use all vertices of layer i. Let v_{i-1} be the highest vertex of a path of height $i-1$ (if that does not exist, we do the whole process in layer $i-1$). In F v_{i-1} is connected to a vertex v_i which is again connected to a vertex v_{i+1}. We break the connection of v_{i-1} and v_i and connect v_{i-1} to the empty vertex v_e instead. That means the whole subtree of v_{i-1} is now attached to v_e and with v_e we now have a path of height i. v_e is then connected to v_{i+1}. Both new connections are possible because v_e and v_i have the same capacity, so if the subtree of v_{i-1} could be connected to v_i, it can also be connected to v_e. And the amount of flow that is directed through v_{i+1} does not change.

Using longer paths in a cable layout thus does not reduce the capacity. □

Algorithm 7.1 tries to connect strings in a way that as few capacity and possibilities for subsequent strings as possible are wasted. The first criterion to this, is to always choose an unconnected vertex if such a vertex exists in a layer. We have just shown that this alone does not destroy the possibility to get a feasible layout for a solar farm graph.

The second criterion is the remaining capacity. It shall prevent that an vertex v with high capacity is connected to a vertex v', where this high capacity of v cannot be fully used because the capacity of v' is already filled by other vertices than v. Figure 7.3 shows an example for this. The green vertex is supposed to be connected. There are two relatively obvious choices, marked by the red and blue dashed line. Which of these two is better depends on the actual capacities of the vertices. We first consider the blue capacities on the left. For these capacities the blue connection is better. Just imagine that each of the bottom layer vertices can be completely filled because there are 12 strings which each produce one unit of flow. With the red connection it would not be possible to get this flow to the top as one transformer can only take 7 units of flow. If all bottom layer vertices which are connected to the right transformer were completely filled the right transformer would have only one unit of capacity left, but the green vertex has potential for two units of flow. So with the red connection and the blue capacities, the potential of the green vertex might be wasted. The blue connection is the connection that the algorithm would choose because of remaining capacities. For the blue vertex the remaining capacity is 2, where it is only 1 for the red vertex.

Now we consider the red capacities. In a first idea of Algorithm 7.1 the criterion to decide to which vertex a connection is made was the amount of flow of a transformer subtree. So

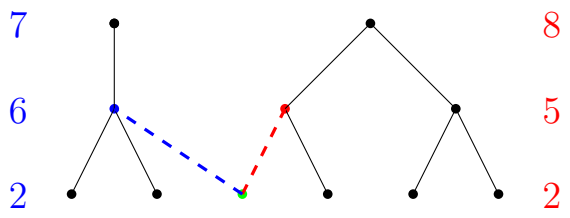


Figure 7.3.: The green vertex needs to be connected. For the blue capacities on the left, the blue connection would be better because with that connection all bottom layer vertices can be completely filled. For the red capacities the red connection is the better choice.

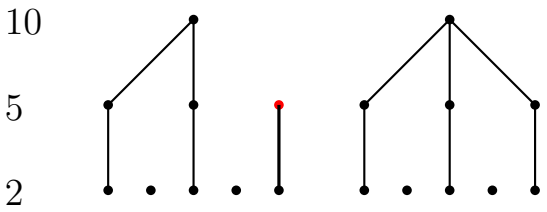


Figure 7.4.: The red vertex shall be connected to one of the two top vertices. Algorithm 7.1 would choose the left one because it has less units of flow than the right one.

first the transformer with the lowest amount of flow was chosen, then the child vertex with the lowest amount of flow and so on. But this method would lead to the blue connection even if we use the red capacities. It is again easy to see that the red connection is the better choice. The left vertex of the middle layer can only take 5 units of flow in total. In the case that its two already connected child vertices are completely filled, one unit of capacity is left for the green vertex. On the other side the red connection would always allow the green vertex to be filled with two units of flow. That is again what is described by the remaining capacity of the red vertex.

But what happens if the remaining capacity is equal for several vertices? We then choose the vertex with the least amount of flow. If here again are several vertices with an equal amount we compare their parents and do this recursively until either the transformers are reached or there is only one candidate left. For this criterion we use that we build the layout path by path. That means that for each path that is added the flow value is increased by one. Figure 7.4 shows an example. The red vertex is already connected to the lower layer. Algorithm 7.1 would now connect the red vertex to the left transformer because both transformers have remaining capacity 0, but the left one has one unit of flow less than the right one. This is the better decision. If we take a look only at the two upper layers one could think that connecting two vertices of the middle layer to a vertex of the top layer is enough to fill the capacity. The problem is that we need three vertices of the bottom layer to fill one vertex of the middle layer. Every time we do that we waste one unit of capacity. That is why it is better to connect more than two vertices of the middle layer to a top layer vertex. However connecting the red vertex to the right side would not increase the amount of flow that can reach the right top vertex. If the remaining capacity of several vertices is equal we want to equally distribute the vertices among them and that is what is done with the flow criterion.

If both remaining capacity and the flow criterion are equal for several vertices it does not matter which vertex we choose. All vertices that were compared with the flow criterion then have the same amount of vertices of the current layer (the highest layer that still has empty vertices) connected to it.

Now we take a look at the runtime of Algorithm 7.1.

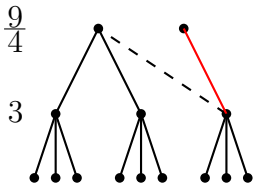


Figure 7.5.: The top vertices are inverters with capacity 9 and lower bound 4. The continuous lines show a cabling that would be returned by Algorithm 7.1. This cabling does not hold the lower bound of 4 for the right inverter. If the red cable would be replaced by the dashed one the resulting solution would be valid.

Theorem 7.5. *The runtime of Algorithm 7.1 is in* $O(n^3)$.

Proof. Each string is considered once and for each string we search a path to a transformer which consists of as many unconnected vertices as possible. Finding this path is possible in linear time. When the path is connected to the existing layout the vertex with the maximum remaining capacity is searched. This is again possible in linear time. If there are vertices with equal remaining capacity, the flow of these vertices needs to be compared. The comparison is again a linear time operation because it might be necessary to recursively check the parent vertices. In total we then get a quadratic time for connecting path to the existing layout. After connecting the remaining capacities have to be updated. The main operation here, updating children of vertices when their remaining capacity changes, is possible with a depth first search which takes linear time in a tree. In total the runtime then is $O(n^3)$ □

7.1. Including Lower Bounds

For now we assume that Algorithm 7.1 can find a cabling for a solar farm graph, which does not violate any vertex capacities. But we have not spoken about lower bounds of inverters. Still we can run the algorithm several times on slightly changed input graphs to try to find a cabling which does not violate lower bounds. But what can actually happen when we run the algorithm only once? We might get a solution where the amount of flow at an inverter is below the minimum allowed value. An example is shown in Figure 7.5. The algorithm assigns two combiner boxes to the left inverter and one to the right which then does not hold enough flow to meet the lower bound. But obviously this example graph is solvable as we can assign all combiner boxes to only one inverter. If we input the same graph but with only one inverter, Algorithm 7.1 would find this solution. To solve this problem we run the algorithm several times, but with a different amount of inverters each time. The minimum amount of inverters needed to get a solution is $\lceil \frac{s}{i_{max}} \rceil$, the maximum amount is $\lfloor \frac{s}{i_{min}} \rfloor$. A solution uses an amount of inverters in between these two values. So we just try out each possible number of inverters.

Although this method surely improves the amount of solar farm graphs on which a solution can be found, there are still graphs where a feasible cabling is obviously possible, but not found by the algorithm. An example is given in Figure 7.6. The trick to calculate a layout using only two inverters does not work here as there are 9 strings, but the capacity of two inverters is only 8. So we always need all of the inverters for a feasible cabling. To get around this problem we simulate a lower capacity for the inverters. If we calculate a layout for the graph of Figure 7.6 with inverter capacities of 3, Algorithm 7.1 would find a correct solution. That is why we do not only try a range of different amounts of inverters, but also a range of different inverter capacities. With these two tricks we can increase the amount of solar farms where a feasible solution can be found.

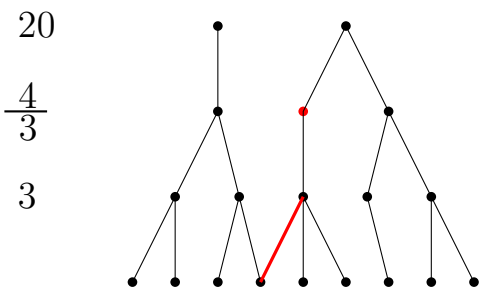


Figure 7.6.: The numbers on the left describe the vertex capacities of the corresponding layer, the numbers separated by the horizontal line describe capacity and lower bound of the inverters. The shown layout (excluding the red connection) would be found by Algorithm 7.1, but it does not fulfill the lower bound of the red vertex. With the red connection the layout would be feasible.

For the runtime of the algorithm we get an additional factor of $|V_I|$ when we check a range of inverter numbers. When we also apply the method to check all possible inverter capacities, we get another factor of $i_{max} - i_{min}$. Both factors are at most n and so the runtime would increase to $O(n^5)$. With a better analysis one can probably show that the runtime is actually lower. In practice the number of inverters is much smaller than the number of strings (of course depending whether central or string inverters are used) and the difference between i_{max} and i_{min} probably as well.

7.2. Local Improvements

With Algorithm 7.1 (and the additional methods for lower bounds) we have a way to find a solution for a solar farm with fully connected layers, although we do not know whether a solution exists in the case when no solution is found. If a solution is found, the only property of this solution is that vertex capacities and inverter lower bounds are not violated. We do not know anything about the costs of the calculated solution. Our goal is now to design a simple heuristic algorithm that makes improvements to a given solution.

Algorithm 7.2 calculates for each vertex with an outgoing cable the cost reduction if the cable is switched to another outgoing edge. For each vertex the best alternative outgoing edge is stored. If an alternative edge would lead to a violation of capacities or inverter lower bounds, this edge is not considered as possible improvement. The global best cable switch is made. This procedure is done until no further improvement is possible. An interesting part is when there are empty vertices in the layout, as they have many possible outgoing edges. For such an vertex the best outgoing edge is calculated with an shortest path algorithm (Line 10). Let x be the amount of flow of the currently investigated vertex and let w be the empty vertex. For each path from w to the top we store how much the costs increase when x units of flow are added to the edges. Then a shortest path algorithm is executed starting at w. Actually the shortest path algorithm can be a breadth first search because of the structure of the layered graph.

The cost reduction calculated for a vertex does not need to be recalculated each time after an improvement was made. If the vertex lies higher in graph as where the changes took place, the possible improvement does not change. In Figure 7.7 an example is shown. After doing the improvement for the red vertex, the possible improvements for the the green vertices do not need to be recalculated.

We now want to evaluate Algorithm 7.1 (including the additional methods for inverter lower bounds) and Algorithm 7.2. That means we calculate a solution with Algorithm 7.1 and then try to improve its cost with Algorithm 7.2.

Algorithm 7.2: Switching upwards paths

Input: solar farm graph $G = (V, E)$ with cable layout
Output: returns a cable layout for G

```
1  costReduction = array of costs reductions, one entry for each vertex
2  alternativeEdge = array of alternative edges for each vertex
3  do
4      initialize costReduction with 0
5      set alternativeEdge to the currently used edge ∀v ∈ V
6      forall v ∈ V\V_T with outgoing cable do
7          x = flow at vertex v
8          forall outgoing edges e of v do
9              c = current costs
10             if end vertex w of e is empty then
11                 calculate costs of connecting w by shortest path algorithm
12             c' = costs when using e as outgoing cable edge of v
13             costReduction (v) = max(c − c', costReduction(v))
14             if new reduced costs then
15                 alternativeEdge (v) = e
16     Switch the cables for the vertex v where costReduction is maximal
17 while max(costReduction) > 0
```

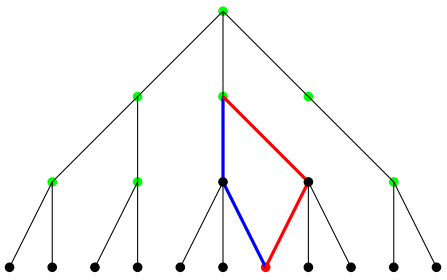


Figure 7.7.: If an improvement is done for the red vertex where the outgoing cable is switched from the blue edge to the red one, the only edges where the flow value changes are the red and blue ones. After that improvement the possible improvements for the green vertices do not change.

8. Experiments

After analyzing a lot of theoretic properties of the solar farm cabling problem and algorithms to solve the problem, we want to analyze the performance of our heuristic algorithm in practice. That is taking a solution found by Algorithm 7.1 (if one is found) and improving it with Algorithm 7.2. The optimization mentioned in Section 7.2 is also implemented. For the evaluation, solar farm instances of three different sizes where generated.

8.1. Solar Farm Generation

The solar farm instances used for the experiments are randomly generated. We decided to only use solar farms with central inverters for better comparability and because for large utility scale solar farms central inverters are the most used solution [ABB19][PK20]. Each solar farm has six layers as described in Section 4.1.6. We choose six layers as this is the most complicated case.

The instances are generated in three sizes *small, medium* and *large*. For each solar farm a rectangle with random width and height is chosen, where the ranges for width and height were determined by rule of thumb. That means a few different values were tried until the space of the rectangle was more or less filled by the strings. After choosing the size a number of strings depending on the size of the solar farm is uniformly random distributed in the rectangle. The ranges of the string numbers are given in Table 8.1.

In general the example solar farm given by ABB [ABB19][p. 120 ff.] was used as an orientation for the instance generation, especially for the following ratio parameters. After placing the strings - how this is done will be explained later - the amount of other vertices (which are Y-connectors, combiner boxes, recombiner boxes, inverters, transformers) is chosen depending on the number of strings and a random ratio within an upper and a lower bound. The exact parameters can be seen in Table 8.1 together with references of e.g. datasheets of PV-components. For example the ratio for strings to combiner boxes is a number between 10 and 20. In the example solar farm of ABB a combiner box combines 13 or 14 strings [ABB19]. This is a rather low value as there are combiner boxes that can handle more input [SMAa], but as the instances also have recombiner boxes, there is no need for such high string to combiner ratios. Additional to that the capacity of the generated components is not the randomly chosen ratio, but a random value that can be up to 1.5 times the ratio (respectively the number of strings divided by the number of components rounded up). A combiner box from LS Electric for example is available with

12, 16 or 20 inputs and up to 500 Ampere capacity [Ele20]. The case with 20 inputs is probably too large to be covered by the generated instances, but as already said, these high values are not necessary with recombiner boxes.

	small	medium	large	references
number of strings	120-180	500-750	1200-1500	
strings to Y-connectors	1.5-3			[Eve16]
strings to combiner	10-20			[ABB19][Ele20][SMAa]
combiner to recombiner	3 - 8	3-10*		[ABB19] [Sol]
strings to inverter	1 in total	200-300		[ABB19]
inverter to transformer	1 in total	1-3		[ABB19][Gaj16]
capacity for $v \in V \backslash V_I$	(min, 1.5 · min)			
capacity for inverters	-	(min, 1.2 · min)		[Mer20]
lower bound for inverter	-	(0.5, 0.8) · capacity		[Mer20]

Table 8.1.: Parameters of the solar farm generation. The upper bounds for the combiner to recombiner ratio for medium and large solar farms is reduced when the string to combiner ratio is already high. The min stands for the minimum value with which there is enough capacity for all strings.

For small solar farms the upper bound of the ratio of combiner to recombiner boxes is a bit smaller. As the number of strings here is a bit lower, this is done to reduce the occurrences of graphs where there is only one recombiner box in the graph. This makes the problem a lot easier and the task of the recombiner box of combining cables could then be done by the inverter directly. With the same argument the upper bound of the combiner to recombiner ratio is reduced for medium and large instances when the number of combiner boxes is already high. When the combining is potentially already done in the combiner boxes, we do not need so many recombiner boxes anymore. The upper bound is $\min(10, 26 - \text{string to combiner ratio})$.

How are the vertices actually placed? The first vertices to be placed are the strings respectively the string connection points. A single string is placed by choosing a random position in the rectangle and then placing three connection points per string on a line with a randomly picked angle, where the angle is picked once globally for all strings. In general the length of this line can be an arbitrary number, but of course equal for all strings. It represents the length of a string. The generated instance have a string length of 40 units. If a random position picked for a string is too close to another already placed string, a new position is chosen. Too close means that the position is less than the string length away from another string. The new position is again checked. If placing a string fails too many times in a row or if the limit for the number of strings is reached, the placement of the strings is stopped. Per string we chose a number of three connection points, they shall represent the possibility to connect a cable to a string at each of both ends or in the middle. More connection points are possible, but seem unnecessary as the distances to other connection points of the same string gets smaller the more connection points are used. Therefore more connection points do not make much of a difference.

The amount of vertices for the other vertices than strings is chosen as described above. Y-connectors and combiner boxes are then placed next to randomly picked strings. They are very small (in the case of Y-connectors only a small electric module) and can be placed directly next to strings. The same applies to recombiner boxes. However they are placed next to randomly picked combiner boxes. The inverters and transformers are again placed like the strings. As these are taking much more space, they also need to be placed with the minimum distance to the strings and other inverters and transformers.

The capacity choice is the same for all vertices except inverters and for small instances the capacity of the single inverter and transformer is simply set to the number of strings. The inverter capacity is only up to 1.2 times the ratio of strings to inverters. Because of efficiency reasons the number of inverters is much more adapted to the number of strings. One would not want to place inverters which have a capacity that is much higher than the number of strings [Mer20]. Inverters additionally have a lower bound also for efficiency reasons as the efficiency varies depending on the input power [Mer20]. The lower bound is chosen randomly between 0.5 and 0.8 times the capacity.

Of course all the capacity values strongly depend on the current rating of the chosen PV modules, that is why among other reasons the instances are generated with the randomness describes above. An example of a generated small solar farm can be seen in Appendix Section A.

8.1.1. Cables

All experiments were executed with the same set of cables, which is shown in Table 8.2. We decided to not distinguish between DC and AC cables, but argue that the cables with high capacity are usually AC cables an the cables with lower capacity for the DC side. As template for the cable list, PV cables by Helukabel [Hel] and faber [fab22] were used. Unfortunately the capacity (rated current) of the cables with the highest capacity that are sold by these two companies is not high enough for large solar farms. Due to a lack of information regarding cables with high enough capacity the information from Helukabel and faber was extrapolated to get cable types with a higher capacity. This applies to the last two cables of Table 8.2. As price information is not easily available either, the costs in Table 8.2 are proportional to the amount of copper used in the cable.

cost	capacity
4	5
34	22
120	50
230	80
750	180
2300	400

Table 8.2.: Cabletypes

In detail the capacity of the cables given by Helukabel and faber was divided by 10 and rounded. The capacity is given in Ampere and usually a string roughly produces a current of 10 Ampere [ABB19, p. 121]. Of course this value can vary a lot depending on the PV modules that are used in a solar farm, but 10 Ampere seems a reasonable value. For the cost of a cable the amount of copper per kilometer cable was divided by 10 too and then rounded. This division was just to make the cost values a bit smaller. If we take the second cable of Table 8.2 as an example, the capacity of the original cable is 218A and 336kg of copper are used per kilometer cable. For the two additional cable types the cost and capacity were estimated per rule of thumb with the principle: Doubling the capacity triples the price.

8.2. Running the Experiments

The heuristic Algorithm 7.2 together with an initial solution produced by Algorithm 7.1 shall be compared with the solution of the MILP formulation that was given in Section 6.2.1. The MILP is solved with Gurobi version 9.5.0 [gur]. As the amount of variables is very

high for large and medium sized solar farm instances and therefore the problem difficult to solve for the MILP solver, the runtime for the solver was restricted to 24 hours. For the heuristic we chose a maximal runtime of one hour.

The code was written in C++14 and compiled with GCC 10.3.0 with the `-Ofast` and the `-march=native` flag. All experiments were run on a 64-bit architecture with four times 12-core AMD CPUs each with 2.1GHz, but both the heuristic and the MILP solver were only run on a single core. In total there are 256GB RAM. The installed operating system is openSUSE Leap 15.3.

As a small side note: The Open Graph Drawing Framework (OGDF) [CGJ+14] was used to represent the solar farm graph in the code. For a cable layout two arrays were used, of which one stores the parent vertex in the layout an the other a pointer to a list of child vertices for each vertex.

8.2.1. Test sets

The experiments are run on three test sets, one with small, one with medium and one with large sized solar farm instances. The instances are generated as described above. Each set consists of 60 solar farms. None of the solar farms of the test sets were proven to be infeasible by the Gurobi solver within 24 hours.

The set of small solar farms consists of solar farms with a string number between 120 and 180 strings and an average of 145 strings. The medium solar farms have between 518 and 750 strings with 640 strings average and the large set consists of solar farms with 1200 to 1500 strings and an average of 1266. All experiments were done with the set of six cables that was shown above. We now take a look at the performance of the MILP and our heuristic for all three test sets.

8.2.2. Small Solarfarms

For the small solar farms the MILP formulation is of reasonable size. The maximum number of edges in the small instance is 20515 and the average about 10000. With six cables being used there are 7 variables for each edge in the MILP formulation. This leads to 70000 variables in average of which a lot can be directly set to 0. For example the outgoing edges of strings will never use a cable type other than the cheapest type. That is why the MILP solver is able to solve these instances optimally in the given time. Optimally in this case means that if the cost value calculated by the MILP solver is less than 0.01% higher than the lower bound calculated by the MILP, the solution is regarded as optimal. Of the 60 small instances all but a single one could be solved optimally by the MILP. This single instance was still solved very close to optimal, as the gap between the lower bound and the calculated solution was 0.016%.

For the optimal solved instances the MILP runtime ranges from 1.6 seconds to almost 10 minutes with an average of about 43 seconds. About 80% of the instances could be solved optimally within one minute. This makes an MILP formulation a relatively powerful option to find a solution for small instances, as we are almost guaranteed to find an optimal solution within a few minutes and most of the times less than a minute.

The heuristic algorithm has much shorter runtimes than the MILP, but a worse solution quality. An example for a heuristic and a MILP solution of a small solar farm can be seen in Appendix Section A. On average the solutions produced by the heuristic have costs of about almost 12% more than the solution of the MILP. The runtimes range from 0.18 to almost 12 seconds with an average of about 3 seconds. This makes the heuristic much faster than the MILP, but it still lags behind in terms of solution quality. Most of the runtime

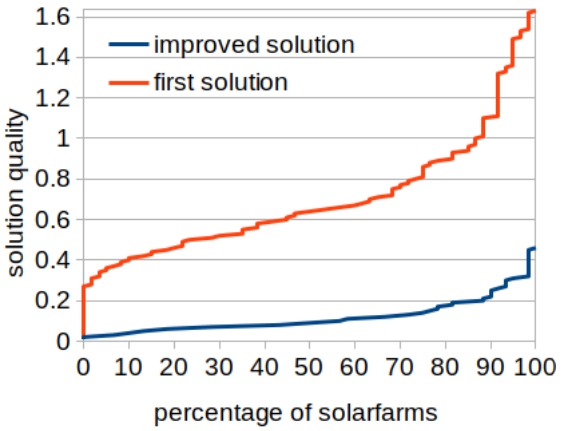
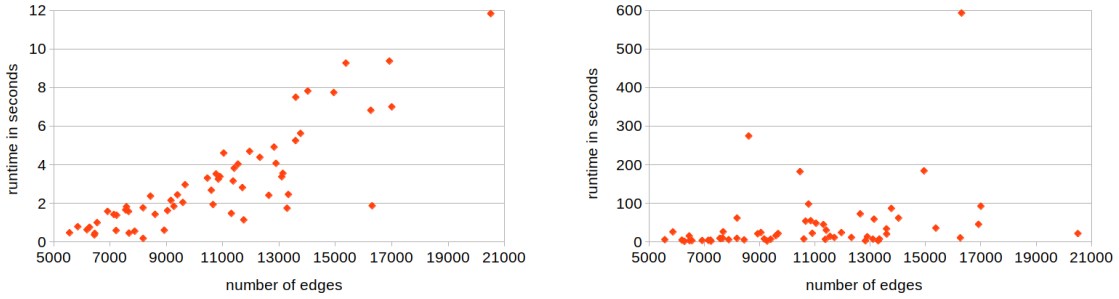


Figure 8.1.: The y-axis shows the additional costs compared to the MILP solution costs for small solar farms. The x-axis shows the percentage of solar farms that were solved with at most the cost given by the y-axis.



(a) Runtime of the heuristic algorithm in relation to the number of edges of the small solar farms.

(b) Runtime of the MILP in relation to the number of edges of all small solar farms that were solved optimally.

Figure 8.2.

of the heuristic is needed to improve the solution. The initial solution for small instances is found within a few milliseconds, the average here is 2 milliseconds. However the cost of the initial solution is on average 70% higher instead of 12%. Figure 8.1 compares the solution qualities of the initial and the improved solution. One can see that where most of the improved solutions have costs of maximal 20% higher than the MILP, there is no initial solution with that solution quality.

There is a clear relation between number of edges and the runtime of the heuristic as can be seen in Figure 8.2. This was to expected as Algorithm 7.2 improves the cost of a solution by iterating over all edges of the graph. Although the number of edges directly determines the number of variables in the MILP, the increase of the runtime depending on the number of edges is not that obvious. Especially the instance with the most amount of edges has a runtime of just above 20 seconds, but there are instances with far less edges and a much higher runtime.

If we compare the capacities of the instance with the most edges and the instance with the longest runtime, the capacities of the instance with the longest runtime are much tighter. That means we take the minimum combined capacity of a layer of the solar farm and divide it by the number of strings. The instance with the most edges for example has 180 strings and the layer with the lowest combined capacity has a capacity of 208. So the capacity tightness of this instance is 1.15. In opposition to that this value is 1.01 for the instance

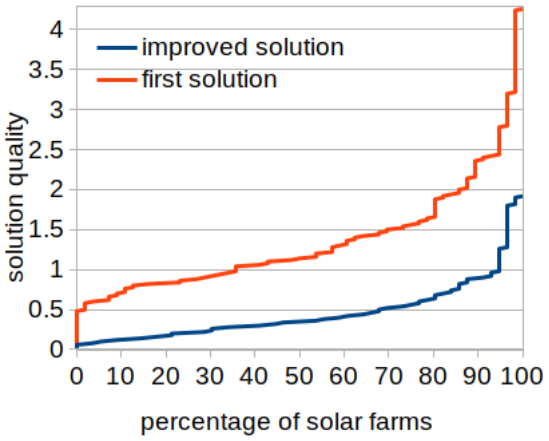


Figure 8.3.: The y-axis shows the additional costs compared to the MILP solution costs for medium solar farms. The x-axis shows the percentage of solar farms that were solved with at most the cost given by the y-axis.

with the longest runtime among all instances that were solved optimally. For the only instance that was not optimally solved the capacity tightness is 1.03. This might lead to the assumption that a tighter capacity leads to a longer runtime for the MILP. However if we compare the average tightness of the lower quartile instances regarding the runtime to the upper quartile instances, the average tightness of the lower quartile is actually smaller. So there is no clear relation between the capacity tightness and the MILP runtime either.

8.2.3. Medium Solarfarms

The medium solar farms consist of much more strings than the small ones. This increased amount of strings compared to the small solar farms causes a huge increase in the number of edges compared to the small instances and therefore an increase of the number of variables for the MILP. Of the 60 instances only 14 could be solved optimally by the MILP in 24 hours and for 4 instances the MILP was not even able to either calculate any solution or prove that the instance has no feasible solution. However it is not clear to say that the number of edges (respectively the number of variables) is the (main) cause that some instances can be solved optimally and some not. The number of edges for all medium solar farms is between 108000 and 346000 edges, the number of edges for the optimally solved solar farms is between 120000 and 304000. So the optimally solved solar farms cover a quite large range of numbers of edges. The number of edges for the four unsolved instances is also not significantly high. Although all of the unsolved instances have edge numbers above the average of about 200000, none of the four instances has more than 270000 edges. But four instances are also not enough to make well-founded statements.

The instances with a feasible but not optimal solution still have a cost value which is on average not even 1% higher than the lower bound for the cost that was calculated by the MILP solver. To reach this solution quality though the MILP solver ran 24 hours. The runtime for the optimally solved instances ranges from slightly more than one hour to more than 17 hours. For solar farms of the medium test set the MILP is not anymore guaranteed to find a optimal solution (within 24 hours), but still after 24 hours the solution quality is very near to the optimum.

The heuristic approach is not able to solve all instances either. These were the same instances that could not be solved by the MILP. All other instances were solved by the heuristic. This might indicate that these four instances are actually not solvable. For the instances that could be solved, the solution costs are between 6% and 200% higher than the

costs calculated by the MILP, which can be seen in Figure 8.3. The figure shows how many solar farms could be solved with a certain quality compared to the MILP solution both for the first and the improved solution. On average the costs for the improved solution are about 50% more than the MILP costs. As with the MILP the runtime for medium sized instances is much higher. For 12 of the 60 solar farms the improvement heuristic did not finish its calculation. The lowest runtime of the solar farms with solution was just short under two minutes. The time to find any feasible solution (if a solution is found) is now about 120 milliseconds. Reason behind this high value is next to size of the solar farm that there are three solar farms where the lower bounds of the inverters are not immediately fulfilled. That means that Algorithm 7.1 is run several times. Without these three solar farms the average runtime for finding a solution (including only solar farms where a solution is found) is only 76 milliseconds.

Comparing these initial solutions to the MILP solutions, we see that the costs are at most about 4 times higher than the costs of the MILP solution. This is a surprisingly low value considered that no sort of optimization is done in Algorithm 7.1 and the runtime to get these solutions bear no comparison with the MILP runtimes.

For medium solar farms the solution quality of the heuristic clearly decreases compared to the solution quality of the small solar farms. The times to find a feasible solution however are still away from the one second mark.

8.2.4. Large Solarfarms

The MILP had big problems with finding a solution for the large solar farms. 22 instances could not be solved or proven to be infeasible within 24 hours, that is more than one third of the test set. The number of edges of the large solar farms is accordingly high. For some instances the number of variables in the MILP comes close to the 10 million mark. But the number of variables is not the sole reason that instances could not be solved by the MILP, as for example the instance with the most amount of edges (1.3 million) could be solved whereas instances with less than half this number of edges could not be solved. And neither could it be proven that they are infeasible.

None of the solved instances was solved optimally. The gap between the calculated lower bound of the MILP and the cost of the solution ranges between 0.1 and 28 percent. On average the difference between lower bound and solution is 6%, which means that the solution quality indeed dropped compared to the medium instances, but still is in an acceptable range.

The heuristic could solve a lot more instances than the MILP. 12 of the 22 instances that were not solved by the MILP could be solved by the heuristic. For the other 10 the heuristic did not find a solution either. So again the heuristic could find a solution for each instance where the MILP found a solution.

No calculation of the heuristic finished within one hour, except for the 10 instances where no solution was found. The solution costs were between 0.4 and 2.5 times higher than the solution cost of the corresponding MILP instance in case that the instance could be solved. This can also be seen in Figure 8.4. For the first found solution of the heuristic, the cost values are actually not much higher. The maximum here is a bit less than five times the MILP solution cost, with the difference that this solution was found in less than a second. The average time for finding the first solution is about one second. However we can observe the same as with the medium instances, that the main factor for an high runtime is the case where Algorithm 7.1 does not find a solution which fulfills the lower bounds in the first try.

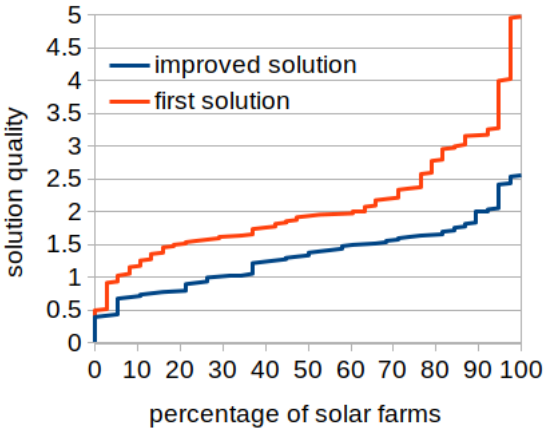


Figure 8.4.: The y-axis shows the additional costs compared to the MILP solution costs for large solar farms. The x-axis shows the percentage of solar farms that were solved with at most the cost given by the y-axis.

For the large solar farms the MILP solver reaches its limits, as roughly one third of the instances could not be solved nor could be proven that these instances are not solvable. The heuristic performs much better in that regard as it can solve 80% of the large solar farm instances.

8.2.5. Summary

Throughout all three test sets the quality of the MILP solutions is better than the heuristic solutions, but the heuristic could find more solutions for the large solar farms. Both MILP and heuristic struggle more to find a feasible solution when the instances get larger. An additional factor which makes medium and large solar farms harder to solve, is that they have more than one inverter. That means that the lower bound of the inverters actually is important when finding a solution. As there is only one inverter for small solar farms, the lower bound is automatically fulfilled for each solution that does not violate the capacities.

The solution quality of the heuristic also decreases drastically when compared to the MILP solution. Where for small solar farms the solution costs are some 10% higher it is up to 250% higher for large solar farms. Figure 8.5 compares the solution quality of all tree test sets. The reduction in quality from small to large is clearly to see. What surprises is that the cost of the initially found solution is not that far away from the solution cost calculated by the MILP, as the maximum value is only 5 times the MILP costs. A reason for that could be fact that the vertex positions for the solar farm graphs are all in the Euclidean plane in a rectangle. So there are no outlier vertices. As a consequence it might be worth it to try to improve Algorithm 7.1 directly instead of using a heuristic that improves a calculated solution. A simple idea could be to not choose an arbitrary empty vertex, but choose the vertex with the shortest distance.

That the heuristic could find a solution each time when the MILP found one, is an indicator that Algorithm 7.1 actually is correct. But it is not more than that as only 180 instances were tested and these additionally have special properties. There is for example never the case that the capacity of a recombiner box is only one unit more than the capacity of a combiner box. To investigate this topic further it would be useful to design a special MILP formulation that just finds any feasible solution. If Algorithm 7.1 is not correct one might find an example where the algorithm does not find a solution although a solution exists.

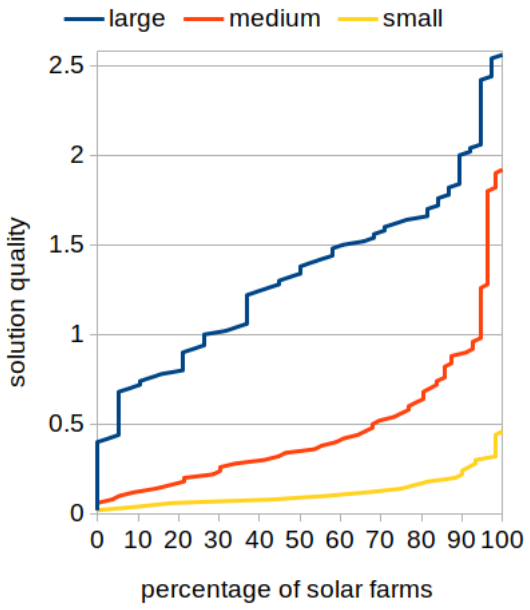


Figure 8.5.: The graph shows the percentage of solar farms that were solved by the heuristic with a solution cost relative to the MILP solution given by the y-axis for all three test sets.

9. Discussion and Conclusion

9.1. Discussion of the Model

We have defined and worked with our solar farm model (Section 4.1.6), but now we want to take a look at the model again to discuss possible downsides and adaptations that could be made. The first addition could be costs for the components of a solar farm besides the cables. In our model it is not rewarded to use less combiner boxes for example, but as combiner boxes are not free of costs it might be good to model their costs as well. That means we want a function $cost_{devices} : V \setminus S \to \mathbb{R}$ that returns the cost for using any device from Y-connector to transformer. The cost function then is

$$cost = cost_{cables} + \sum_{v \in V \setminus S} cost_{devices}(v) \cdot b(v)$$

where $b(v) = \begin{cases} 1 \text{ if there is an edge } e = (v, c) \text{ with } f(e) > 0 \\ 0 \text{ else} \end{cases}$.

With that we basically mixed a facility location problem into the cabling problem. However neither the dynamic program (Algorithm 6.1) nor the Algorithm 7.1 to find any feasible solution need to be changed much, as this addition does only change the cost function, but not the structure of the problem.

One of the probably biggest problems of our model is efficiency. Solar farms are usually designed in a way to maximize the power output of the solar farm. There are several properties that affect the efficiency, some more and some less. One of these properties is the lower bound for inverters which is already part of the model. Another one are power losses in cables. These are not taken into account. The problem here is that the longer a cable is the higher is its resistance and this leads to higher power losses [Mer20]. Usually there are guiding values how high these losses should be at maximum [ABB19]. In our model they are not considered, which is surely a downside.

Another efficiency related topic are possible different azimuth angles of the strings, as already described in Section 4.1. Sometimes there are strings with relatively strong differing angles, which means that the sun rays hit these strings with a different angle and this might lead to different current outputs. A single maximum power point tracker, usually located in an inverter, is then not able to get the maximum power out of the connected strings. If azimuth angles of the strings would be taken into account, $a : S \to [0°, 360°]$

assigns an angle to every string. If the angle between two strings $s_1, s_2 \in S$ is too large, $|a(s_1) - a(s_2)| > angle_{max}$, they have to be connected to different inverters. However this is more of an special case, as most large scale solar farms have the same azimuth angle for all their strings. And if not string inverters can help with this problem as then there are much more inverters and maximum power point trackers. Sometimes inverters even have more than one maximum power point tracker.

There is another problem including the cables, which already has been mentioned in Section 4.1. If several cables are laid in the same conduit, the waste heat of the cables may reduce the ampacity of the cables. The actual positions where cables are laid are not part of our model. One might argue that the cables can be installed in different conduits to avoid this problem.

Now to the probably most problematic design decision. We model electric flows as graph flows. Especially for the conversion of DC current to AC current, this is at least questionable. For graph flows we simply apply the flow preservation property, but inverters are much more complicated devices. Power is lost during the conversion, the voltage is changed and reactive power might be added in the inverter. Can these properties still be modeled with classic graph flows?

Many of the problems discussed above come from the goal to design a rather simple model, but despite these problems we believe that the model is a basis one can work with.

9.2. Summary and Outlook

Now we want to take a look back at what has been done during this thesis. We introduced the solar farm cable layout problem and designed a model for it. This model simplifies some properties of a real world solar farm and we discussed possible problems with that. Although there might be weaknesses in the model it is a basis to work with. The cable layout problem was analyzed in different variants and even if many of the parameters are simplified, e.g. unit edge lengths and one unrestricted cable type, the problem is still NP-hard. This peaks in the finding that it is NP-hard to even find any feasible solution for the general solar farm cabling problem. And from that it follows that no polynomial time approximation algorithm with any approximation guarantee can exist for the solar farm cabling problem unless P=NP.

Because of that we restricted the search of an algorithm that can solve the cabling problem (in a sense of finding any solution) in polynomial time to a special variant where the layers of the solar farm graph are fully connected. This problem even with vertices in $\mathbb{R}^2$ is still NP-hard. It might be interesting to investigate what approximations are possible for this special variant. We presented an algorithm to calculate a solution for this problem, but we were not able to prove that this algorithm always calculates a solution if a solution exists. Additionally the algorithm only calculated solutions which fulfill the capacities, but not necessarily the lower bounds. Proving the correctness of the algorithm or finding a counterexample is surely one of the next steps to be taken. If the correctness can be proven, it would be nice to also enhance the algorithm to lower bounds.

For existing solutions of the solar farm cabling problem we designed a simple heuristic (Algorithm 7.2) that can improve a given solution. However when testing the heuristic on some benchmark instances, it could be seen that it was very slow. In opposite to that a first solution could be found very fast in most of the cases. Surely the implementation of Algorithm 7.2 can be improved, however it could also be worth to improve Algorithm 7.1 so that the solution found here is already more optimized. Besides that the heuristic approach could solve solar farm instances that could not be solved by an MILP. This was the case for large instances.

One of the biggest questions for future work surely is the design and proof of an algorithm that can find a solution for the variant of the cabling problem where the layers are fully connected. Maybe there are some additional assumptions that can be made to simplify the solving the optimizing of the cabling problem, for example are the strings of a solar farm usually embedded in the Euclidean plane (or at least very near to that). Another task would be to improve the runtime and the solution quality of an algorithm which tries to solve the solar farm cabling problem. Our tested implementation was both slow and the solution quality was at least for medium and large instances far away from a quality that could be used in practice. Maybe it is even possible to improve the MILP formulation, as many variables are 0 right from the start. So there are still some big question marks, but we hope that this thesis gives as basis to solve them.

Bibliography

[ABB19] ABB. Photovoltaic plants - Cutting edge technology. From sun to socket. May 2019. URL https://search.abb.com/library/Download.aspx?DocumentID=9AKK107492A3277&LanguageCode=en&DocumentPartId=&Action=Launch. last visited 2022-02-01.

[AG10] Gehrlicher Solar AG. Lauingen Energy Park, 2010. URL http://www.gehrlicher.com/fileadmin/content/pdfs/Factsheets/Factsheet_Lauingen_A4_en.pdf. last visited 2022-01-03.

[Age21a] International Energy Agency. Renewable Energy Market Update - Outlook for 2020 and 2021, May 2021. URL https://iea.blob.core.windows.net/assets/18a6041d-bf13-4667-a4c2-8fc008974008/RenewableEnergyMarketUpdate-Outlookfor2021and2022.pdf. last visited 2022-01-03.

[Age21b] International Renewable Energy Agency. *Renewable Power Generation Costs in 2020*. Internationl Renewable Energy Agency, July 2021. isbn:978-92-9260-348-9. URL https://www.irena.org/publications/2021/Jun/Renewable-Power-Costs-in-2020.

[APA18] Amar Prakash Azad, Manikandan Padmanaban, and Vijay Arya. A data lens into MPPT efficiency and PV power prediction. In *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, Washington DC, DC, USA, February 2018. IEEE. doi:10.1109/ISGT.2018.8403327.

[AS13] Sara Ahmadian and Chaitanya Swamy. Improved Approximation Guarantees for Lower-Bounded Facility Location. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Thomas Erlebach, and Giuseppe Persiano, editors, *Approximation and Online Algorithms*, volume 7846, pages 257–271. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-38016-7_21. Series Title: Lecture Notes in Computer Science.

[BK01] Adriana F. Bumb and Walter Kern. A simple dual ascent algorithm for the multilevel facility location problem. *Electronic Notes in Discrete Mathematics*, 8:14–17, May 2001. doi:10.1016/S1571-0653(05)80067-8.

[BMB20] Ratnakar Babu Bollipo, Suresh Mikkili, and Praveen Kumar Bonthagorla. Critical Review on PV MPPT Techniques: Classical, Intelligent and Optimisation. *IET Renewable Power Generation*, 14(9):1433–1452, July 2020. doi:10.1049/iet-rpg.2019.1163.

[BRG89] David Bernstein, Michael Rodeh, and Izidor Gertner. On the complexity of scheduling problems for parallel/pipelined machines. *IEEE Transactions on Computers*, 38(9):1308–1313, September 1989. doi:10.1109/12.29469.

[Bum02] Adriana F. Bumb. *Approximation algorithms for facility location problems*. PhD thesis, Twente University Press, Enschede, 2002. isbn: 9789036517874 OCLC: 67015241.

[BVMO11] Constantin Berzan, Kalyan Veeramachaneni, James McDermott, and Una-May O'Reilly. lgorithms for Cable Network Design on Large-scale Wind Farms. Technical report, Massachusetts Institute of Technology, 2011.

[CGJ+14] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). In R. Tamassia (ed.), editor, *Handbook of Graph Drawing and Visualization*, chapter 17. CRC Press, 2014. URL `https://ogdf.uos.de`.

[CKP00] Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. The Multiple Subset Sum Problem. *SIAM Journal on Optimization*, 11(2):308–319, January 2000. doi:10.1137/S1052623498348481.

[Cor15] International Finance Corporation. Utility-Scale Solar Photovoltaic Power Plants: A Project Developer's Guide, June 2015. URL `https://www.ifc.org/wps/wcm/connect/topics_ext_content/ifc_external_corporate_site/sustainability-at-ifc/publications/publications_utility-scale+solar+photovoltaic+power+plants`. last visited 2022-02-01.

[Ele19] General Electric. LV5+ Solar Inverter Data Sheet, September 2019. URL `https://www.ge.com/renewableenergy/sites/default/files/related_documents/lv5plus-solar-inverter-datasheet.pdf`. last visited 2022-02-01.

[Ele20] LS Electric. PV Combiner Box, April 2020. URL `https://www.vmc.es/es/system/files/archivos/pv-combiner-box_catalog_en_202004.pdf`. last visited 2021-07-25.

[Eve16] Eric Every. Using Y-connectors in String Inverter Systems. *SolarPro*, (9.4):16–20, July 2016.

[fab22] faber. Solarleitung FABER SOLAR H1Z2Z2-K, 2022. URL `https://shop.faberkabel.de/out/downloads/uploads/DE/dbl_h1z2z2-k.pdf`. last visited 2022-01-10.

[FGMS06] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm - SODA '06*, pages 611–620, Miami, Florida, 2006. ACM Press. doi:10.1145/1109557.1109624.

[GAB+20] Andrew Glick, Naseem Ali, Juliaan Bossuyt, Marc Calaf, and Raúl Bayoán Cal. Utility-scale solar PV performance enhancements through system-level modifications. *Scientific Reports*, 10(1):10505, December 2020. doi:10.1038/s41598-020-66347-5.

[Gaj16] John W. Gajda. Solar Farms: design & construction - Impacts to utility distribution systems, 2016. URL `https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/presentations/nesc_`

workshop__solar_farms-design_construction.pdf. last visited 2022-01-10.

[GKS16] Bernard Gendron, Paul-Virak Khuong, and Frédéric Semet. A Lagrangian-Based Branch-and-Bound Algorithm for the Two-Level Uncapacitated Facility Location Problem with Single-Assignment Constraints. *Transportation Science*, 50(4):1286–1299, June 2016. doi:10.1287/trsc.2016.0692.

[gur] Gurobi. URL https://www.gurobi.com/. last visited 2022-02-01.

[GUW+19] Sascha Gritzbach, Torsten Ueckerdt, Dorothea Wagner, Franziska Wegner, and Matthias Wolf. Engineering Negative Cycle Canceling for Wind Farm Cabling. *arXiv:1908.02129 [cs]*, August 2019. URL http://arxiv.org/abs/1908.02129. arXiv: 1908.02129.

[GvO10] Adriana F. Gabor and Jan-Kees C.W. van Ommeren. A new approximation algorithm for the multilevel facility location problem. *Discrete Applied Mathematics*, 158(5):453–460, March 2010. doi:10.1016/j.dam.2009.11.007.

[Hel] Helukabel. Cables and cable systems for photovoltaic installations. URL https://www.helukabel.com/media/publication/de/brochures/pv_20/PV_BROCHURE_Photovoltaik_EN~1.pdf. last visited 2022-01-10.

[JJ18] Mark Z. Jacobson and Vijaysinh Jadhav. World estimates of PV optimal tilt angles and ratios of sunlight incident upon tilted and tracked PV panels relative to horizontal panels. *Solar Energy*, 169:55–66, July 2018. doi:10.1016/j.solener.2018.04.030.

[Kar72] Richard Karp. Reducibility among combinatorial problems. volume 40, pages 85–103, 01 1972. doi:10.1007/978-3-540-68279-0_8.

[KDS14] Jozef Kratica, Djordje Dugošija, and Aleksandar Savić. A new mixed integer linear programming model for the multi level uncapacitated facility location problem. *Applied Mathematical Modelling*, 38(7-8):2118–2129, April 2014. doi:10.1016/j.apm.2013.10.012.

[Ken15] Tom Kenning. Europe's largest solar park: Squeezing out maximum energy from the minimum land space - PV Tech, September 2015. URL https://www.pv-tech.org/europes_largest_solar_park_squeezing_out_maximum_energy_from_the_minimum_la/. last visited 2022-02-01.

[Lab21] National Renewable Enerfy Laboratory. U.S. Solar Photovoltaic System and Energy Storage Cost Benchmark: Q1 2020. Technical report, National Renewable Enerfy Laboratory, January 2021. URL https://www.nrel.gov/docs/fy21osti/77324.pdf. last visited 2022-01-26.

[LYC+19] Tao Liu, Xu Yang, Wenjie Chen, Yang Li, Yang Xuan, Lang Huang, and Xiang Hao. Design and Implementation of High Efficiency Control Scheme of Dual Active Bridge Based 10 kV/1 MW Solid State Transformer for PV Application. *IEEE Transactions on Power Electronics*, 34(5):4223–4238, May 2019. doi:10.1109/TPEL.2018.2864657.

[Mer20] Konrad Mertens. *Photovoltaik: Lehrbuch zu Grundlagen, Technologie und Praxis*. Hanser, München, 5., aktualisierte auflage edition, 2020. isbn:978-3-446-46404-9.

[MK16] Versorgungswerke Mainz-Kinzig. Kreiswerke Main-Kinzig: Solarpark Bruchköbel, 2016. URL `https://www.versorgungsservice-main-kinzig.de/Solarpark-Bruchkoebel.1991.0.html`. last visited 2022-02-01.

[Nat13] National Fire Protection Association. *National electrical code handbook, 2017*. National Fire Protection Association, Quincy, Mass., 2013. isbn:978-1-4559-0672-7 978-1-4559-1277-3 978-1-4559-0673-4.

[PGB19] Diane Palmer, Ralph Gottschalg, and Tom Betts. The future scope of large-scale solar in the UK: Site suitability and target analysis. *Renewable Energy*, 133:1136–1146, April 2019. doi:10.1016/j.renene.2018.08.109.

[PK20] Jeremy Powell and Panos Kamperidis. Central vs. String Inverters: Myth & Reality, July 2020. URL `https://www.pv-magazine.com/webinars/central-vs-string-inverters-myth-reality/`. last visited 2022-01-09.

[RR20] Hannah Ritchie and Max Roser. Renewable energy, 2020. URL `https://ourworldindata.org/renewable-energy`. last visited 2022-02-01.

[SMAa] SMA. SMA String-Combiner - Datasheet. URL `https://files.sma.de/downloads/DC-CMB-U-DEN1834-V16web.pdf`. last visited 2022-02-01.

[SMAb] SMA. Sunny Central Up - Datasheet. URL `https://files.sma.de/downloads/SC4xxxUP-DS-en-26.pdf`. last visited 2022-01-23.

[Sol] SolarBOS. Standard Recombiners. URL `http://www.solarbos.com/Standard-Recombiners`. last visited 2022-01-10.

[Sol21] First Solar. First Solar Series 6 - Datasheet, May 2021. URL `https://www.firstsolar.com/en-Emea/-/media/First-Solar/Technical-Documents/Series-6-Datasheets/Series-6-Datasheet.ashx`. last visited 2021-07-28.

[Tro20] Kalle Trofast. *Optimization of utility-scale PV plant design for three-phase string inverters*. PhD thesis, Aalto University, April 2020. URL `https://aaltodoc.aalto.fi/bitstream/handle/123456789/44320/master_Trofast_Kalle_2020.pdf?sequence=1&isAllowed=y`. last visited 2022-01-20.

[TSAMA17] Madjid Tavana, Francisco J. Santos Arteaga, Somayeh Mohammadi, and Moslem Alimohammadi. A fuzzy multi-criteria spatial decision support system for solar farm location planning. *Energy Strategy Reviews*, 18:93–105, December 2017. doi:10.1016/j.esr.2017.09.003.

[VRVD16] Rajiv K. Varma, Shah Arifur Rahman, Tim Vanderheide, and Michael D. N. Dang. Harmonic Impact of a 20-MW PV Solar Farm on a Utility Distribution Network. *IEEE Power and Energy Technology Systems Journal*, 3(3):89–98, September 2016. doi:10.1109/JPETS.2016.2550601.

[Wag19] Andreas Wagner. *Photovoltaik Engineering: Handbuch für Planung, Entwicklung und Anwendung*. VDI-Buch. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019. isbn:978-3-662-58454-5 978-3-662-58455-2, doi:10.1007/978-3-662-58455-2.

[Whi16] Sean White. *PV technical sales: preparation for the NABCEP technical sales certification*. Routledge, Abingdon, Oxon ; New York, NY, 2016. isbn:978-0-415-71334-4 978-1-315-77008-6.

[Yac21] Jeanne Yacoubou. A Guide To Solar Farm Land Requirements - Green Coast, December 2021. URL `https://greencoast.org/solar-farm-land-requirements/`. last visited 2022-01-07.

10. Appendix

A. Generated and Solved Solar Farm

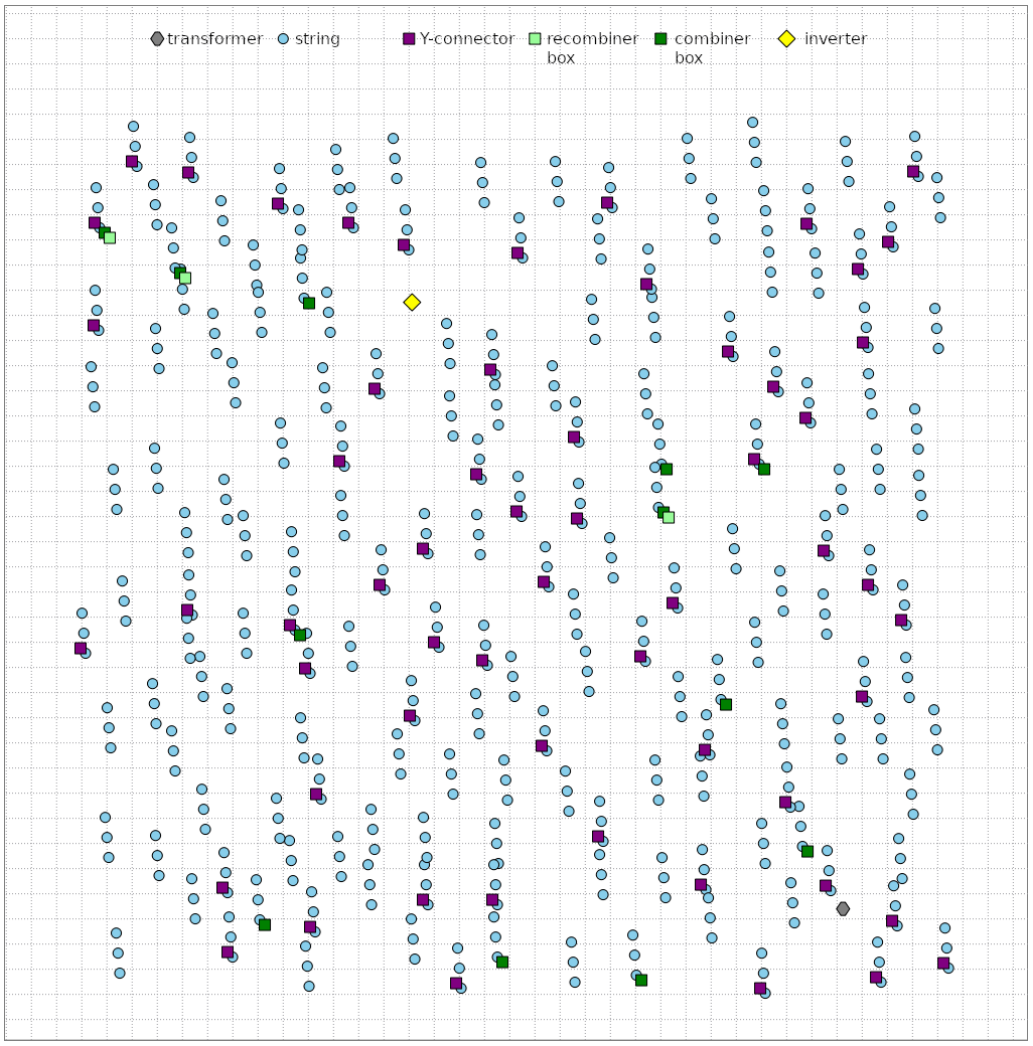
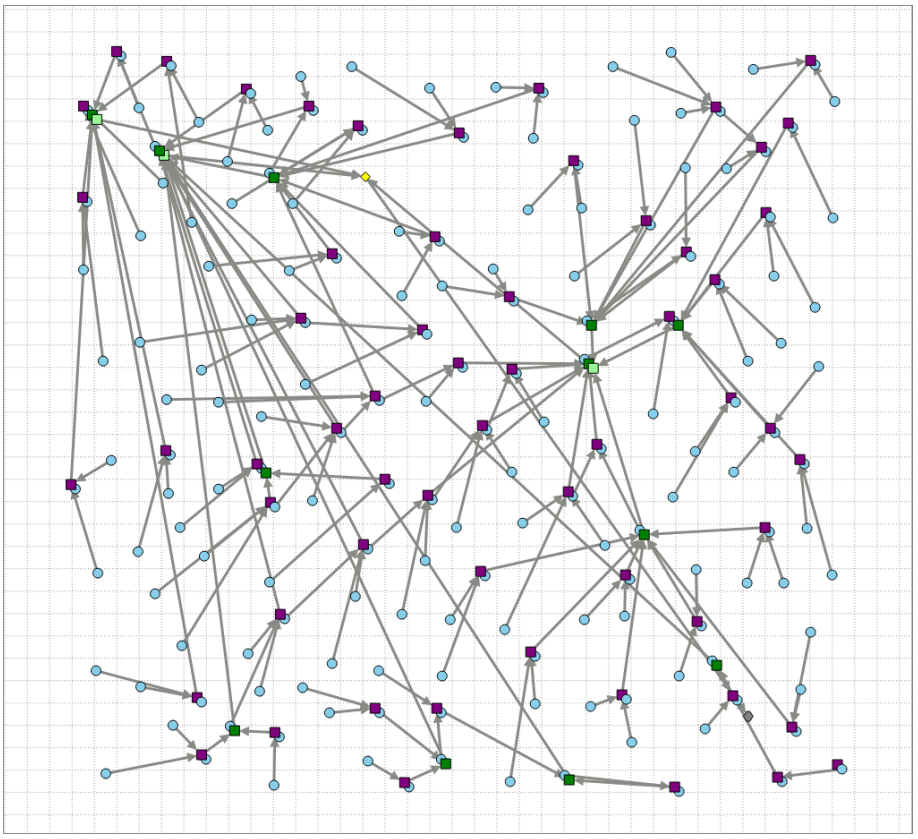
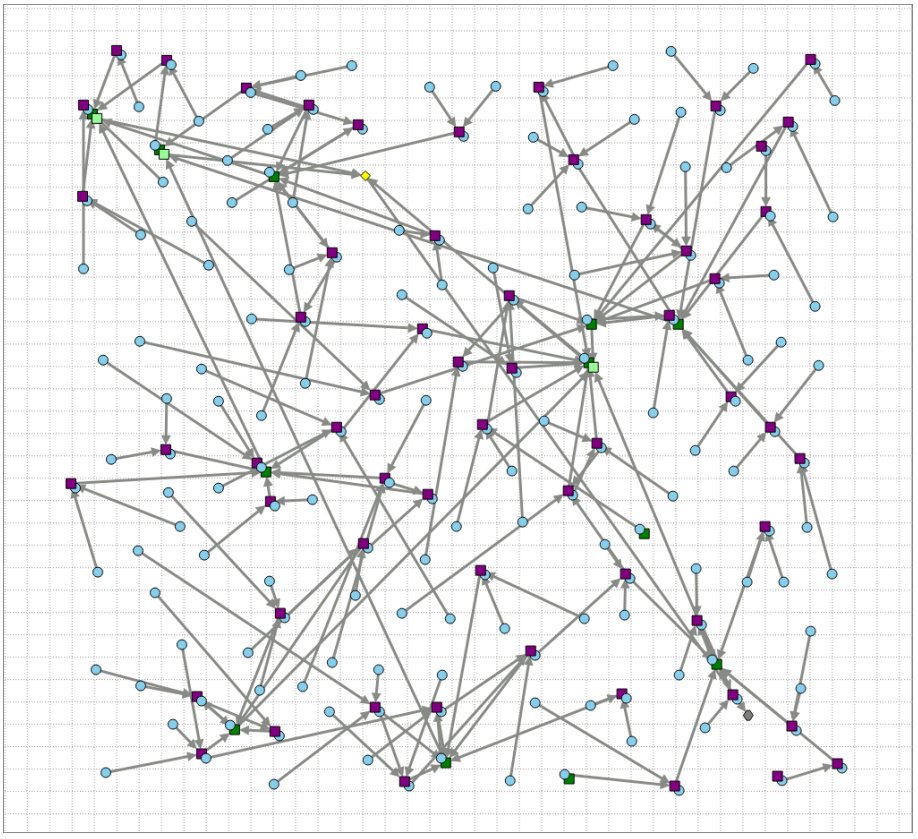


Figure A.1.: Example of a small solar farm with three connection points per string. Both an solution by the MILP and a solution by the heuristic can be found below.



(a) Optimal solution of the cabling problem calculated by the MILP



(b) Solution of the cabling problem calculated by the heuristic.

Figure A.2.: Solutions of the solar farm above by MILP and heuristic. As explained in Section 4.1.8 the connection points of each string are merged into one vertex before a solution is calculated.