RESEARCH **Open Access**

Solar farm cable layout optimization as a graph problem

Sascha Gritzbach*, Dominik Stampa and Matthias Wolf

From The 11th DACH+ Conference on Energy Informatics 2022 Freiburg, Germany. 15-16 September 2022

*Correspondence:
sascha.gritzbach@kit.edu
Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract
We introduce the Solar Farm Cable Layout Problem (SoFaCLaP), a novel graph-theoretic optimization problem. SoFaCLaP formalizes the task of finding a cost-optimal cable layout in a solar farm where PV string positions are already determined but the positions of other components such as transformers can be picked from a set of candidate positions. The problem statement incorporates a network flow model in which the flow value of a connection represents the number of strings that are (indirectly) connected to a transformer via this connection. A mixed-integer linear program (MILP) formulation is proposed that uses binary variables to indicate which of several available cable types is chosen for each connection. We propose a framework to randomly generate benchmark instances to evaluate any algorithmic approach to SoFaCLaP. In particular, we generate a set of instances based on real-world solar farm characteristics. With an extensive evaluation of the MILP formulation on those instances we establish mixed-integer linear programming as a baseline for future algorithmic approaches to finding solar farm cable layouts.

Keywords: Solar farm, Cable layout, Mixed-integer linear program, Benchmark generation, Graph

Introduction
The previous decade has seen a boom in the commission of renewable energy power plants, in particular for solar farms. The global installed capacity in photovoltaic (PV) systems has seen a 16-fold increase to over 707 GW between 2010 and 2020 (IRENA 2021). During the same time, the weighted-average cost per newly installed mega-watt has decreased by approximately 93%, mainly—but not solely—due to a decrease in cost of PV modules (IRENA 2021). In 2019, 116.9 GW of new PV capacity were added world-wide, out of which 75.3 GW were installed in utility-scale solar farms (SolarPower Europe 2020).

For the convenience of the reader, we give a general description of the main components of a utility-scale solar farm. For further details, we refer to ABB Ltd (2019), Mertens (2019), which also served as references for the following elaboration. The most visible parts of a solar farm are the solar cells. In those, sunlight is converted to tiny amounts of electric current. The cells are connected forming a PV module. These modules, in turn, are connected in series to form a string which is mounted on a rack. For the remainder of this work, we will consider the PV strings as the smallest building block, since we are mainly interested in optimizing the cable layout. The strings supply electricity in form of Direct Current (DC), which is converted to Alternating Current (AC) in inverters. Strings can have their own inverters (*string inverters*, connected to one or at most to a few strings) or a larger number of strings is connected to only a few *central inverters*. We focus on central inverters but everything in this work can also be adapted to string inverters. In general, inverters have more functions than only conversion. They are used for monitoring and safety purposes and generally also incorporate control elements such as maximum power point trackers, by which the DC voltage in the connected components is adjusted according to environmental conditions to maximize electrical power harvest. Solar farms typically operate at low-voltage levels (the DC side is usually aimed at a maximum voltage of 1 kV or 1.5 kV), so that step-up transformers are needed to feed the generated power into the grid. Since inverters have only a finite number of input circuits, additional devices are installed between strings and inverters. These devices have different names, depending on the monitoring and safety equipment installed in them and on the components they connect. *Y-connectors* are the most simple device. They normally connect just two strings with no additional equipment [except maybe for fuses (Every 2022)]. *Combiner boxes* connect a larger number of strings (or Y-connectors) and have additional safety and monitoring equipment. *Recombiner boxes* have the same equipment as combiner boxes but connect combiner boxes instead of strings. Which kind of components, in particular between strings and inverters, are used ultimately depends on a decision by the solar farm planners. In any case, the components need to be connected by cables. For electrical reasons the cable layout should be balanced to some extent, e. g. to avoid reverse current. In particular, the layered structure should be respected, for example, connecting a recombiner box (to which multiple strings are connected) and a single string to an inverter should be avoided.

With all those restrictions in mind, solar farms appear to be mostly constructed on flat ground following one (of maybe several) pre-specified templates. However, there are exceptions: The Monte Mele photovoltaic plant is situated on the slope of a hill on Sicily, Italy, and has irregular distances between its strings (Alpiq Holding AG 2022). For a solar farm of the size of the Monte Mele plant with its capacity of 718 kW, drawing a cable layout by hand might be feasible. For larger solar farms [the world's currently largest stands at 2245 MW (Sanjay 2022)], algorithmic approaches computing near-to-cost-optimal cable layouts might be the way to go.

Related work

Various aspects of solar farms have been a target for optimization in the literature. Solar cells can be manifactured from different so-called photovoltaic absorber materials that influence the performance of a cell, for a review see (Kirchartz and Rau 2018). The efficiency of transformers can be influenced by using appropriate control methods (Liu et al. 2019) and realized in a prototype. Shifting the focus to the overall electrical system of a solar farm and its operation, a variety of maximum power point tracking techniques can be employed to maximize the power output of the farm (Bollipo et al. 2020). Concerning the early stages of the planning process, a fuzzy Analytic Hierarchy Process has been proposed to find an optimal site for a solar farm (Tavana et al. 2017).

A more holistic view on solar farm design is employed at Siemens Energy: In a multi-criteria decision support system a set of pre-computed designs for a given site can be compared and investigated by planners visually and with respect to different "key performance indicators" (Bischoff et al. 2014). The designs are computed using several (unspecified) heuristics in a three-stage process involving three subproblems: placing service ways, placing strings in the area between ways, and inverter placement. Given the positions of strings and inverters, a cable layout is computed in "single-objective manner, [minimizing] cable cross sections such that specified losses are not exceeded" (Bischoff et al. 2014, p.337). The exact optimization problem is not stated.

Using a different setting, a formalization of the optimization problem of finding a cost-minimal cable layout is given in Luo et al. (2021). Computing cable layouts and the placement of combiner boxes assigned to a single inverter is modelled as a generalized capacitated minimum spanning tree problem and solved by a branch-and-price-and-cut algorithm. In this setting, strings are placed on a grid and edge lengths are given by the ℓ_1 metric. A capacitated spanning tree connecting strings and the inverter yields a cable layout in which a combiner box is placed at each child string of the inverter such that the capacity of the combiner box is not exceeded. The costs arise from a linear-cost flow on edges between any two strings and from a step-cost function between combiner boxes and inverter. The latter models the installation costs of the combiner boxes.

This paper generalizes the task of determining an optimal cable layout and optimal positions of combiner boxes in a solar farm. It is assumed that the positions of the strings are already determined and that candidate positions for all other components, e. g. (re-)combiner boxes or transformers, of a solar farm are given. The goal is to find an optimal subset of the candidate positions that allows all strings to be connected to a transformer such that the total cabling costs are minimized—in Luo et al. (2021) only candidate positions of combiner boxes are considered and they are co-located with strings. As for the cabling, a set of available cable types is given (each with a thermal capacity and a cost per unit of length). For any two components that can be connected according to the aforementioned layered structure one of the cable types should be chosen such that the cable has sufficient capacity and is as cheap as possible. This gives rise to a step-cost function for each connection, not only between combiner boxes and inverters. Such cost functions have been used in papers on optimizing the cable layout of a wind farm (Dutta and Overbye 2011, Lehmann et al. 2017), in which a network flow model with a step-cost function is used). The main difference between our solar farm model and the wind farm models lies in the way in which vertices can be connected. Our model has a layered structure which allows connections only between adjacent layers. In wind farms, there are only two types of vertices (turbines and substations) and turbines are connected to each other.

Contribution

On our endeavour to generalize the task of determining an optimal layout of a solar farm, we introduce a graph problem called the Solar Farm Cable Layout Problem (SoFaCLaP, Problem formulation). The problem statement uses a network flow model to identify the point-to-point connections (*edges*) that are used for the cable layout and to identify how many strings use each connection. The flow on each connection is used to identify which cable type is used for the respective connection. SoFaCLaP bears resemblance to the Steiner Arborescence Problem (Ljubić 2021) and to the Multilevel Facility Location Problem (Ortiz-Astorquiza et al. 2018).

We propose a mixed-integer linear program (MILP) formulation (A mixed-integer linear program) to provide a first solution method for SoFaCLaP. This formulation uses real variables for the flow values and binary variables to indicate which cable type is used for which edge. An extensive evaluation of the performance of the optimizer Gurobi on the MILP formulation is carried out (Results: optimality and (in-)feasibility, Results: running time, and Results: solution quality) to establish mixed-integer linear programming as a baseline for performance evaluations of future approaches to solving SoFaCLaP. For the evaluation of the MILP formulation we propose a framework to randomly generate synthetic benchmark instances (Framework for benchmark generation). We employ this framework using parameters based on real-world solar farms (Generating benchmark instances). The benchmark instances used in the evaluation are publicly available.

The problem statement and the MILP formulation can be adapted to cover further aspects of solar farm layouting (cf. Variants of the problem and SoFaCLaP from a practical perspective). The process for generating benchmark instances can incorporate such aspects. It can also be customized to address further requirements of solar farm planners and to use different characteristics of various electrical devices.

To summarize, this paper has four main purposes: to introduce SoFaCLaP as a graph optimization problem, to establish an MILP formulation as a standalone solution method that can also be used as a baseline for future algorithms, to introduce a framework for generating synthetic input instances, and to provide an example set of instances based on real-world characteristics.

Problem formulation

The Solar Farm Cable Layout Problem (SoFaCLaP) is a minimum-cost flow problem on a directed layered graph $G = (V, E)$. The vertex set consists of the strings V_S that need to be connected, as well as of the potential Y-connectors V_Y, combiner boxes V_C, recombiner boxes V_R, inverters V_I, and transformers V_T. Edges have a length $\mathrm{len} : E \to \mathbb{R}$ and only exist between one layer and the next, i.e.,

$$V = V_S \cup V_Y \cup V_C \cup V_R \cup V_I \cup V_T, \tag{1}$$

$$E \subseteq V_S \times V_Y \cup V_Y \times V_C \cup V_C \times V_R \cup V_R \times V_I \cup V_I \times V_T. \tag{2}$$

For easier reference, the layers are enumerated from $V_1 = V_S$ to $V_6 = V_T$. Vertices from layers $V_2, \ldots, V_6$ have an upper capacity $\mathrm{u}_i \in \mathbb{N}$ for $i = 2, \ldots, 6$ on the amount of strings that can be routed via these vertices. Inverters also have a lower capacity ℓ_5. We may refer to the capacity bounds of a layer by a vertex of that layer, e. g. $\mathrm{u}_2 = \mathrm{u}(v)$ for any $v \in V_2$. The left-hand side of Fig. 1 shows an example of layered graph with capacities as given by a SoFaCLaP instance.

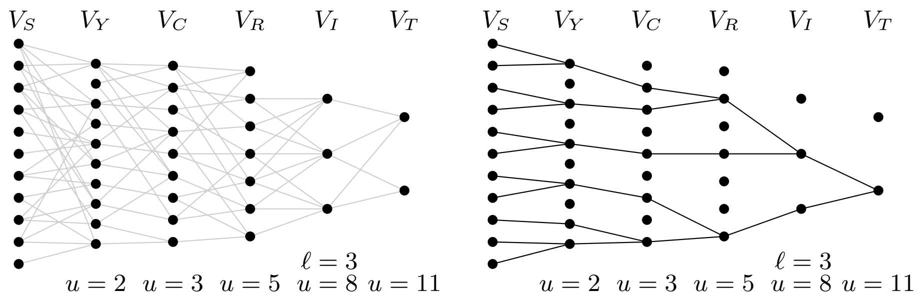
V_S V_Y V_C V_R V_I V_T
$\ell = 3$
$u = 2$ $u = 3$ $u = 5$ $u = 8$ $u = 11$
V_S V_Y V_C V_R V_I V_T
$\ell = 3$
$u = 2$ $u = 3$ $u = 5$ $u = 8$ $u = 11$

Fig. 1 An example instance of SoFaCLaP showing the layered graph and the capacities of all layers *(left)* and—assuming that there is a cable type with sufficient capacity—a feasible cable layout *(right)*. The flow values on the edges are omitted in this visualization. They are, however, uniquely determined, since the cable layout is cycle-free, and can be computed by counting the strings in the respective subtrees of the layout

The inverters split the instance into an AC-side with the transformers and a DC-side with the strings. Denote the edges on the AC-side by$E_{\text{AC}} = \{(i,j) \in E : i \in V_I, j \in V_T\}$ and by $E_{\text{DC}} = E \setminus E_{\text{AC}}$ the edges on the DC-side. For either side, there is a set of possible cable types K_{AC} and K_{DC}. Each cable type k has a thermal capacity $\text{cap}(k)$ and a cost per unit of length $c(k)$. For easier notation, the subscripts AC and DC are omitted. The cable types include a dummy cable type of capacity and cost 0.

The goal of SoFaCLaP is to find a minimum-cost flow on G, i. e. a function $f : E \to \mathbb{R}_{\geq 0}$ of minimal cost. The flow must transport one unit of flow from each string to a transformer, conserve the flow at intermediate vertices, respect the vertex capacities, and have at most one outgoing edge from each vertex with positive flow. The latter enforces a cycle-free cable layout. The right-hand side of Fig. 1 shows an example layout. The cost of a flow on an edge is given by $c(k^\star) \cdot \text{len}(e)$ where $k^\star$ is the cheapest cable type with sufficient capacity to carry the flow. The total cost of a flow is the sum of costs over all edges. A flow that exceeds the maximum cable capacity on one or more edges is not considered a feasible solution.

SoFaCLaP is a computationally difficult problem.

Theorem 1 *It is strongly NP-complete to decide if an instance of the Solar Farm Cable Layout Problem has a feasible solution.*

Proof A candidate for a feasible solution to SoFaCLaP can be provided by specifying the flow values on the edges. To verify feasibility, it must be checked that flow conservation holds and that all vertex and cable capacities are respected. This is possible in polynomial time. Thus, membership in NP is shown. Note that it is not necessary to compute the costs, which could involve computations with real numbers.

The hardness proof uses a reduction from the strongly NP-hard problem 3-Partition (Garey and Johnson 1979, SP15): Let $m, T \in \mathbb{N}$ and let $S := \{s_1, \ldots, s_{3m}\}$ be a multiset of natural numbers such that $T/4 < s < T/2$ for all $s \in S$ and $\sum_{s \in S} s = mT$. Can S be partitioned into triplets $S_1, \ldots, S_m$ such that $\sum_{s \in S_i} s = T$ for all $i = 1, \ldots, m$?

In the reduction, the Y-connectors represent the elements of the multiset and the combiner boxes the triplets. The edges between strings and Y-connectors force an outflow from the Y-connector equal to the respective element of the multiset. So given an instance of 3-Partition, we construct an instance of SoFaCLaP with $3m$ Y-connectors $y_1, \ldots, y_{3m}$, each with capacity $T/2$, and mT strings such that the s_i strings have an edge only to y_i for all $i = 1, \ldots, 3m$. The instance has m combiner boxes $c_1, \ldots, c_m$, each with capacity T. Recombiner boxes, inverters and transformers are not needed for this reduction, and neither are cable types, so there one of each, with capacity mT. All layers are fully connected except for the string layer as mentioned above. This graph has $mT + 3m + T + 3$ vertices and $mT + 3mT + m + 2$ edges. Thus, the construction is possible in polynomial time since we may assume that T is polynomial in the size of the input.

We show that the instance of 3-Partition is a yes-instance if and only if the SoFaCLaP instance has a feasible solution. If the instance of 3-Partition is a yes-instance, then we connect a Y-connector y_i to a combiner box c_j if and only if $s_i \in S_j$. The edges between the strings and the Y-connectors yield that y_i has an outflow of exactly $s_i < T/2 = \mathrm{u}(y_i)$. Since $\sum_{s \in S_j} s = T$, the capacity of c_j is not exceeded either. Thus, we obtain a feasible SoFaCLaP instance.

On the other hand, let the constructed instance of SoFaCLaP be feasible. By means of $T/4 < s$, it follows that each combiner box has at most three Y-connectors connected to it. If one combiner box had only two Y-connectors, another one would have four. Thus, each combiner box is connected to exactly three Y-connectors. Since the total capacity of all combiner boxes is exactly mT, the inflow at each combiner box equals T. Thus, the assignment of Y-connectors to combiner boxes gives an assignment of the elements of S to m triplets with the desired property. $\square$

We have shown that already finding a feasible solution is a difficult task, at least in the most general setting of SoFaCLaP. One might hope that the whole problem becomes a lot easier with additional simplifications. On the positive side, we suspect that finding a feasible solution is easy (Conjecture 1). On the negative side, it can be shown the optimization remains difficult (Theorem 2, proof omitted).

Theorem 2 *It is strongly NP-hard to decide if an instance admits a solution with a value below a given threshold, even if the instance respects all of the following*:

- *Vertex positions are given by points in* $\mathbb{Q}^2$ *and no two vertices have the same positions.*
- *Edge lengths are given by the Euclidean distance of the endvertices.*
- *All layers are fully connected.*
- *There is only one cable type, which has unlimited capacity.*
- *Lower vertex capacities do not apply.*

Conjecture 1 *In the setting of Theorem 2, a feasible solution can be computed in polynomial time.*

These simplifications have multiple motivations: Euclidean coordinates and edge lengths are in line with real-world applications where vertices are placed in the plane. Fully connected layers occur when no connection is forbidden a priori by the solar farm planner (and on top of that, the construction in the proof of Theorem 1 that forces a certain input to each Y-connector cannot be readily replicated). The remaining simplifications are rather a feature of the complexity proof, i.e., we do not need multiple cable types nor lower capacities to show NP-hardness. Nonetheless, they may be reasons for hardness themselves.

Variants of the problem

The order of the layers in the graph resembles a solar farm with central inverters. Solar farms can also be designed using string inverters (Mertens 2019), in which case $V_S, V_Y, V_I, V_C, V_R, V_T$ would be the order of the layers. The MILP formulation proposed in this paper can easily be adapted to the setting with string inverters.

Strings can also be understood as having multiple connection points (*outlets*), out of which one is chosen for the cable layout. Each outlet may have different Y-connectors it can be connected to. The model above already covers this variant: For any pair of string and Y-connector, only the connection point closest to the Y-connector could possibly be used in an optimal solution. Therefore, all outlets of a string can be contracted into a single one (and therefore identified with the string itself). The edge lengths from the string to Y-connectors are then adjusted to reflect the actual distance between Y-connector and closest outlet. The synthetic benchmark instances proposed in this paper employ this variant using three connection points per string.

The problem statement can naturally be extended to incorporate more potentially interesting aspects of solar farm layouting. Cable types can not only vary between the AC- and the DC-side but also between the layers. They may even vary between the edges, i. e. each edge may have its own set of cable types. This can for example be used to incorporate vertex installation costs or costs for different levels of utilization in relation to the capacity of a vertex.

SoFaCLaP from a practical perspective

We have mentioned that SoFaCLaP co-optimizes two aspects: picking a subset of potential intermediate vertices and finding a cable layout using vertices from that subset. A solar farm planner who wishes to use SoFaCLaP and any algorithmic approach has to specify the input, i.e., graph and cable types. For the graph, the planner has to decide on the (fixed) positions of all strings. They also have to provide possible locations for any intermediate device and the respective capacities. From those locations, the algorithmic approach will pick a suitable subset to be used in the cable layout. In their decisions, the planner has many degrees of freedom, which might not be obvious. Take, for example the edge set and the edge lengths. The planner can (but need not) choose to disallow certain connections from the graph for whichever reason. The edge lengths can (but need not) represent Euclidean lengths. One might choose to use the ℓ_1 metric ["Manhattan metric", as in Luo et al. (2021)] which only allows "horizontal" and "vertical" cable sections. Or the planner can assign arbitrary lengths to account for difficult terrain. This also includes a constant additive offset to edge lengths to account for necessary cables used inside the different devices.

With these and many more degrees of freedom, SoFaCLaP is supposed to be adaptable to many different modelling decisions a solar farm planner might want to take.

A mixed-integer linear program

SoFaCLaP can be formulated as a MILP using variables $f_{ij} \geq 0$ for the flow on $(i,j) \in E$ and binary variables x_{ijk} stating whether cable type k is used on edge (i, j). Here, the zero-capacity cable type is omitted. This way of modelling the cable types has been used for the Wind Farm Cabling Problem (Lehmann et al. 2017). The following formulation is based on the standard variant of SoFaCLaP with the implicit inclusion of string connection points as outlined in Variants of the problem.

The goal of SoFaCLaP is to minimize the total installation cost

$$\min \sum_{(i,j)\in E} \sum_{k\in K} x_{ijk} \cdot c(k) \cdot \mathrm{len}((i,j)). \tag{3}$$

The total flow leaving each string is the production of the string

$$\sum_{(i,j)\in E} f_{ij} = 1 \quad \forall i \in V_S, \tag{4}$$

and flow must be conserved at intermediate vertices

$$\sum_{(i,j)\in E} f_{ij} = \sum_{(j,l)\in E} f_{jl} \quad \forall j \in V \setminus (V_S \cup V_T). \tag{5}$$

The capacities at the vertices must be respected

$$\sum_{(i,j)\in E} f_{ij} \leq \mathrm{u}(j) \quad \forall j \in V \setminus V_S, \tag{6}$$

$$\sum_{(i,j)\in E} f_{ij} \geq \ell(j) \cdot \sum_{(i,j)\in E} \sum_{k\in K} x_{ijk} \quad \forall j \in V_I, \tag{7}$$

whereas lower capacities only apply if a vertex is indeed used.

For all vertices except transformers, only one outgoing edge may have flow and thereby non-zero capacity

$$\sum_{(i,j)\in E} \sum_{k\in K} x_{ijk} \leq 1 \quad \forall i \in V \setminus V_T, \tag{8}$$

which implies that only one cable type is used per edge. This cable type must have sufficient capacity to hold the flow

$$f_{ij} \leq \sum_{k \in K} x_{ijk} \cdot \mathrm{cap}(k) \quad \forall (i,j) \in E. \tag{9}$$

In total, this linear program has $\mathcal{O}(|K| \cdot |E|)$ binary and $|E|$ real variables as well as $\mathcal{O}(|V| + |E|)$ constraints.

Simulations and evaluation

An evaluation of the solution methods outlined in A mixed-integer linear program on example instances shall give the reader insights into the performance of the MILP and thereby establish it as a (first) solution method to SoFaCLaP. Since we have not been able to obtain suitable real-world instances[1] to evaluate the MILP, we resort to generating synthetic instances as described in Framework for benchmark generation and in Generating benchmark instances. The benchmark instances use the GraphML format (Brandes et al. 2002) and are available from http://www.doi.org/10.35097/676, where a thorough specification of the format can be found as well.

If the benchmark instances are sufficiently variable to cover a wide range of plausible inputs to the solution approaches, the use of synthetic instances has multiple advantages. First, a larger number of different instances can be generated so that tendencies in the comparison of two approaches are less likely to be a result of statistical noise. Second, the solar farms in consideration can include more strings than any currently existing solar farms. Thus, the applicability of the algorithms for a future increase in solar farm sizes can already be investigated now.

Framework for benchmark generation

We describe the general approach we propose for the generation of benchmark instances on a high level in this section. The next section is dedicated to breathing life into the framework by describing the process in more detailed steps with concrete values for the parameters. These values are based on real-world examples of devices used in solar farms as evidenced by Tables 1 and 2.

The first steps in the framework deal with placing the strings. For synthetic instances we opt for randomly sampling the strings with a minimum distance. Next, the location of vertices of higher layers are determined, starting at the Y-connectors. We let the number of vertices in a layer depend on the number of vertices in a previous layer. We assume fully connected layers, which leads to the maximum number of possible connections between layers. After the graph is determined, vertex capacities are set. We choose these randomly with bounds based on the minimum capacity necessary to connect all strings. Lastly, cable types are included which we derive from real-world photovoltaic cables.

Users of the proposed framework may want to alter any intermediate step to realize certain additionally desired properties on the solar farm instances. For example, users could prespecify all the string positions or fix the number of vertices within a layer. They may want to change the sampling process, for example to have all recombiner boxes

[1] As alluded to in SoFaCLaP from a practical perspective, it does not suffice to merely have the positions of the strings and other devices in an existing solar farm. We rather need all potential positions of devices. This represents an intermediate step in the planning process which is, understandibly, not publicly available.

Table 1 Design parameters for graph layer sizes

Layer		Category								
In consideration	**Reference**	**Small**			**Medium**			**Large**		
V_S	1	120	–	180	500	–	750	1200	–	1500
V_Y	V_S	1.5	–	3	1.5	–	3	1.5	–	3
V_C	V_S[a]	10	–	20	10	–	20	10	–	20
V_R	V_C[b]	3	–	8	3	–	10[c]	3	–	10[c]
V_I	V_S	$\|V_I\|$	=	1	200	–	300	200	–	300
V_T	V_I	$\|V_T\|$	=	1	1	–	3[d]	1	–	3[d]

[a] In the example solar farm by ABB (ABB Ltd. 2019, Annex B), each combiner box connectes 13 or 14 strings. A combiner box by LS Electric has possible inputs of 12, 16, or 20 (LS ELECTRIC Co., Ltd. 2021). Combiner boxes by SMA connect up to 32 strings, but no recombiner boxes are used (SMA Solar Technology AG 2020)

[b] Based on the recombiner boxes in SolarBOS, Inc. (2022)

[c] 10 or (26 — ratio $V_C : V_S$), whichever is lower, based on the idea that parallel connections of many strings in both combiner and recombiner boxes yields excessive amounts of electric current

[d] An example of two inverters connected to a transformer can be found in Gajda (2022)

Table 2 Cable types used in simulations

Cable type		Specifications as in HELUKABEL GmbH (2021)	
Cost c	**Capacity cap**	**Ampacity [A] at 60 °C**	**Copper Weight [kg per km]**
4	5	55	38.4
34	22	218	336
120	50	488	1152
230	80	775	2304
750	180	Extrapolated	
2300	400	Extrapolated	

close to each other. The edge set can be thinned out if, for example, cables should not exceed a certain length. Users can define their own cable types.

Generating benchmark instances

For our purposes, the framework introduced in the previous section is used as follows. Benchmark instances are generated in three categories: *Small* (120–180 strings per instance), *Medium* (500–750 strings), and *Large* (1200–1500 strings). We describe the process of generating an instance: First, a rectangle is fixed that simulates the area of the solar farm. One of the corners is defined as the origin giving rise to a coordinate system. The (virtual) unit of length is irrelevant since SoFaCLaP solution values scale with the underlying unit of length. Next, an angle α for the orientation of the strings in relation to the coordinate system is chosen randomly. The total number of strings is drawn from the intervals above and, one after another, the strings are placed in the rectangle: A random point P is sampled from the rectangle serving as one of the connection points of the string. For the purpose of describing the process, we refer to this point as the *base* of the string. If P respects a globally specified minimum distance from the bases of all previously successfully sampled strings, the connection points are defined. On a ray starting at P with angle α to the coordinate system's x-axis two more connection points are placed equidistantly, with the distance being the same for all strings. If, however, P is too close to another base, a new base P is sampled. If the process of placing a string fails too often, the whole instance is discarded and a new rectangle is chosen.

With all the strings placed, the higher layers of vertices are randomly placed, starting with the Y-connectors. For each layer, a ratio according to Table 1 is chosen randomly and the number of vertices on the next layer is defined as $V_{\text{next}} = \lceil V_{\text{reference}}/\text{ratio}\rceil$. The ratios are chosen in a way to ensure decreasing sizes of layers. For the *small* instances, the number of transformers and inverters is fixed as 1. Y-connectors and combiner boxes are placed in close proximity to string connection points. Recombiner boxes are placed close to combiner boxes. Inverters and transformers are placed on random points in the rectangle, respecting minimum distances to the string bases as well as previously placed inverters and transformers. The reasoning is that strings, inverters and transformers are themselves bigger components and need to be more easily accessible for maintenance.

Picking (upper and lower) capacity values for the different layers can be a delicate issue from an algorithmic point of view: With very tight capacities, the design questions of picking one position for a, say, combiner box over another becomes easy: all possible positions have to be used. Very loose capacities mitigate the algorithmic difficulty of picking a suitable subset of, say, strings to be assigned to a specific Y-connector. We deem the assumption fair that all vertices of a layer have the same capacities since planners presumably would not design solar farms with two different versions of, say, inverters. Given a layer V_i a capacity of at least $\text{lb}_i := \lceil |V_S|/|V_i|\rceil$ is required. For each layer, a random integer between lb_i and $\lambda \cdot \text{lb}_i$ is chosen, where $\lambda = 1.2$ for inverters and $\lambda = 1.5$ for all other layers. The smaller value for inverters is selected with the idea in mind that solar farm planners will not overly deviate the capacities of inverters from the capacity best suited for the intended use. For recombiner boxes and inverters only, the capacity will, however, be at least twice the capacity of the previous layer to ensure that two fully-used vertices of the previous layer can be connected. The randomly chosen lower capacity for inverters is at least 0.5 and at most 0.8 times the inverters' upper capacity.

The edge set is assumed to be complete, i.e., equality holds in Eq. (2). We have no reason to prohibit specific edges (although solar farm planners might). Thus, we leave it to the algorithms to pick the most suitable edges from the complete set.

The cable types for the simulations as shown in Table 2 are based on the photovoltaics cable SOLARFLEX-X PV1-F by HELUKABEL (HELUKABEL GmbH 2021). Four sizes are considered, the forth one being the biggest size available. The rated ampacity is approximately translated into a capacity on the amount of strings, where a current of 10 A per string is assumed (ABB Ltd 2019, p. 121). Since cost quotations are not readily available, we assumed that the costs are in essence proportional to the amount of copper used in a cable. The exact unit for the costs is irrelevant since the optimization function in SoFaCLaP can be scaled arbitrarily. For aesthetical reasons, we opted for somewhat round values in the bigger cable types. No influence on the performance of algorithms is expected from those changes. These four cable types are not big enough to be used in the largest benchmark instances. Thus, we included two artificial cable types to conduct meaningful experiments on the biggest instances. These cable types were extrapolated from the smaller cables types, roughly following the motto "Doubling the capacities yields triple the cost."

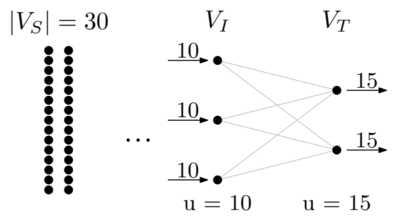

Fig. 2 A toy example of a solar farm, in which all vertex capacities are sufficient but their interaction implies infeasibility. Due to 30 strings, each inverter needs an inflow of 10 and each transformer an inflow of 15. But with only one outgoing edge per vertex allowed, inverters and transformers cannot be connected

Table 3 Characteristics of instances used in simulations

Category	$\|V_S\|$			$\|V\|$			$\|E\|$		
	min	**median**	**max**	**min**	**median**	**max**	**min**	**median**	**max**
Small	120	145	180	176	224.5	305	5566	10528	20515
Medium	502	632	750	781	966.5	1234	108654	190792	345969
Large	1200	1248	1500	1726	1920	2459	538923	732858	1369140

Some of the benchmark instances generated as outlined above may be infeasible. Since capacities are chosen in a way that the total capacity of a layer is at least the number of strings and since the edge set is complete, infeasibility can only arise from the interaction of the sizes and capacities of the layers as a whole. A toy example of such a constellation can be seen in Fig. 2. We purposefully keep such instances because it may be a feature of algorithms developed for SoFaCLaP to be able to detect infeasibility (see also "Results: optimality and (in-)feasibility").

Simulation setup

The MILP formulation from A mixed-integer linear program is translated into C++14 code. The code uses the Open Graph Drawing Framework (Catalpa release) (Chimani et al. 2013) for parsing the benchmark instances and the C++-API provided by Gurobi (Gurobi Optimization, Inc. 2018). For compilation, GCC 10.3.0 is used with the `-Ofast -march=native` flags. All simulations are run on a 64-bit architecture with four 12-core AMD CPUs each with 2.1 GHz with the openSUSE Leap 15.3 operating system. The MILP formulation is solved by Gurobi 9.5.0 in single-threaded mode with a maximum running time of one day per instance, with 40 instances running simultaneously.

For the simulations, 80 randomly generated instances of each of the three categories *Small*, *Medium*, and *Large* are considered. An overview of the characteristics of these 240 instances is shown in Table 3.

The ranges for the number of strings are almost fully used. It stands out that the median of number of strings in the *large* instances is much smaller than 1350, which is what one would naïvely expect. A possible reason is that the process of placing a large number of strings, inverters and transformers, each of which observe a minimum distance to each other, failed too often. In that case, the instances would be

Table 4 Amount of instances per optimization status for different stages of MILP simulations

Category	Total	Optimal	Feasible	Infeasible	Unknown
MILP (A mixed-integer linear program), one day					
Small	80	79	1	0	0
Medium	80	16	48	12	4
Large	80	0	50	5	25
MILP (A mixed-integer linear program), one cable type, 1 day					
Small	0	–	–	–	–
Medium	4	–	0	2	2
Large	25	–	14	5	6
MILP, one cable type, additional constraints (Eq. 10)					
Small	0	–	–	–	–
Medium	2	–	0	1	1
Large	6	–	0	1	5

The specifications for the experiments are in the main text body

discarded and a new number of strings would be drawn randomly. A possible remedy is to enlarge the initial rectangle for the solar farms for the large instances.

The number of vertices is linear in the number of strings with a small coefficient. This is as expected from the ratios shown in Table 1. The number of edges grows fast with over 1.3 million edges for instances with up to 1500 strings. From a theoretical perspective, a fast increase is expected: Layers are fully connected and the number of vertices per layer is linear in the number of strings. Thus, the number of edges grows quadratically with the number of strings.

Results: optimality and (in-)feasibility

In our setting, Gurobi terminates with one of four optimization states: Ideally, an instance is solved to (proven) optimality (Status: *optimal*). Secondly, Gurobi may find a feasible solution but fail to prove that it is optimal (Status: *feasible*). In that case, the best proven lower bound (lb) does not match the solution value (ub), and the feasible solution may or may not be optimal. Thirdly, Gurobi may be able to prove that an instance does not admit any feasible solution (Status: *infeasible*). Lastly, Gurobi may not find any feasible solution but fail to prove infeasibility. In that case, the instance may or may not be feasible (Status: *unknown*).

In the following, we want to investigate how often Gurobi terminates in each of those states across the different categories of instances to see what we can learn about the applicability of the MILP formulation and Gurobi to solve SoFaCLaP.

The upper part of Table 4 shows how often Gurobi terminates in the different states at the end of the maximum running time of one day. Nearly all of the small instances are solved to optimality within one day. A solution is considered optimal once the MIP gap[2] is below the optimality tolerance of 0.0001%. Since the number of variables and constraints in the MILP formulation grow linearly with the number of edges, Gurobi can most probably not uphold the performance on bigger instances.

[2] Gurobi can bound the optimal value from above by the best incumbent solution (value of ub) and from below by more sophisticated considerations (lb). The MIP Gap is defined as $(\mathrm{ub} - \mathrm{lb})/\mathrm{lb}$.

Indeed, for the medium instances, the number of instances that are solved to optimality decreases by a big margin compared to the small instances, but still one in five instances is solved optimally. About one in seven instances are shown to be infeasible. The fact that Gurobi does not find any feasible solution on 4 out of 80 instances within one day (optimization status: unknown) makes us suspect that those instances are infeasible. On instances of those sizes one would expect that Gurobi would find a feasible solution within one day if a solution existed. We come back to those "unknown" instances later in this section.

As for the biggest instances, Gurobi is not able to solve any instance to optimality. For more than one in four instances, Gurobi can neither find an optimal solution nor prove infeasibility. We speculate that allowing Gurobi more time or simplifying the model can give further insights into the true status of these instances.

Thus, we run two additional sets of simulations on the remaining "unknown" instances. The first set is intended to find feasible solutions. We suspect any then remaining unknown instance to be infeasible. On those instances, we use a more restricted but (in the context of infeasibility) equivalent MILP formulation which we believe helps Gurobi to prove infeasibility. We describe the two sets of simulations in more detail and see what we can learn about the instances of unknown status.

To make Gurobi's life (supposedly) easier on the remaining instances of *unknown* status, we allow only one cable type. This greatly reduces the number of binary variables at the start of the optimization. If the capacity of the cable type is set to a value higher than the maximum vertex capacity, the simplified instance is feasible if and only if the original instance is, since the biggest cable capacity exceeds the biggest vertex capacity in all instances in question here. Again, we give Gurobi a maximum running time of one day per instance in single-threaded mode. The outcome of these simulations is shown in the middle part of Table . Gurobi can prove infeasibility on two additional medium and on five additional large instances. Gurobi also finds a first feasible solution on 14 out of 25 large instances. Yet, two medium and six large instances remain unknown.

Given the amount of help we gave Gurobi to find feasible solutions, we believe that the remaining instances are in fact infeasible. To facilitate Gurobi's proof of infeasibility, we again change the MILP formulation. Since the capacities within a layer are equal and layers are fully connected, we observe that any feasible solution can be transformed into other feasible solutions by a permutation of the vertices of a layer. In order to prove infeasibility, Gurobi needs to outrule all these possibilities. Presumably, this can be facilitated by imposing a fixed order on the vertices of each layer, in which the inflow into the vertices decreases. In formulae, fix a layer V' (other than the strings) and enumerate the vertices inside this layer as $V' = \{v_1, \ldots, v_k\}$. Then, add the constraints

$$|V_S| \geq \sum_{(u,v_1)\in E} f_{uv_1} \geq \ldots \geq \sum_{(u,v_k)\in E} f_{uv_k} \geq 0 \tag{10}$$

for all layers V'. Additionally, we set the target function to a constant, so that Gurobi terminates once it finds a feasible solution. With this formulation, we allow Gurobi four threads per instance and lift the time limit. Surprisingly, these changes do not help Gurobi much. Even with a total computation time over all threads per instance of more than six weeks, only two additional instances are proven infeasible (cf. lower part of Table 4). At the time of writing, the simulations are still running.

We conclude from these investigations on the optimization status with which Gurobi terminates that Gurobi is very well suited for solving small instances and obtaining feasible solutions on quite a number of instances representing medium and large solar farms. We also note that Gurobi is able to prove infeasibility quite often in reasonable time (with a bit of help), even though our trick with the additional constraints from Eq. (10) proved less useful than hoped for. For some large instances, Gurobi cannot find a feasible solution within one day even though these instances are in fact feasible. While allowing Gurobi more time is always an option, one could think about other approaches to find a feasible solution (cf. Conjecture 1), give it to Gurobi as a warmstart and see what Gurobi can do from there.

Results: running time

We take another look into the instances that are solved to optimality concerning the time Gurobi needs to reach this state. 79 out of 80 small instances are solved to optimality. Gurobi reaches this result within half a minute on 55 instances and within 5 min on 78 instances. The longest proof of optimality on the small instances needs just under 10 min. We suspect that Gurobi has no difficulties on the small instances, since the number of potentially non-zero binary variables can be greatly reduced by combining the constraints: For string edges it is never beneficial to use another than the smallest cable type. Also, the biggest cable type is never used on small instances since the second biggest suffices. More sophisticatedly, the small number of strings and therefore small number of vertices in further layers limits the number of cable types that can be used to due the vertex capacities. For example, consider a solar farm with 180 strings and 60 Y-connectors. By the formulae outlined in Generating benchmark instances, Y-connectors can only have a capacity of 3 or 4. In both cases, the smallest cable type also suffices for the edges between Y-connectors and combiner boxes. For all these reasons, we believe Gurobi is able to cut off many suboptimal solutions very easily, which results in short running times.

One medium instance is solved to optimality within 1 h (within 20 min actually) and seven more within 4 h. The longest time for a proof of optimality here is 18 h. Clearly, with more running time, the number of instances solved to optimality would increase further. Unfortunately, it cannot be known how long Gurobi needs. So it is more insightful to look into the worst-case deviation of a best solution from the (unknown) optimal solution value. For this, we go back to the MIP gap in the following section.

Results: solution quality

We look into the quality of the solutions Gurobi has found within one day using the original MILP formulation. By the upper part of Table , the instances in consideration are all with status *optimal* or *feasible*. These are 80 small, 64 medium and 50 large instances. There is not much to say about the small instances; they are almost all solved to optimality. The remaining instance has 180 strings (the maximum) and 11749 edges. On this instance, Gurobi terminates with a MIP gap of 0.0161%. That means between the best solution found is at most 0.0161% worse than the optimal solution.

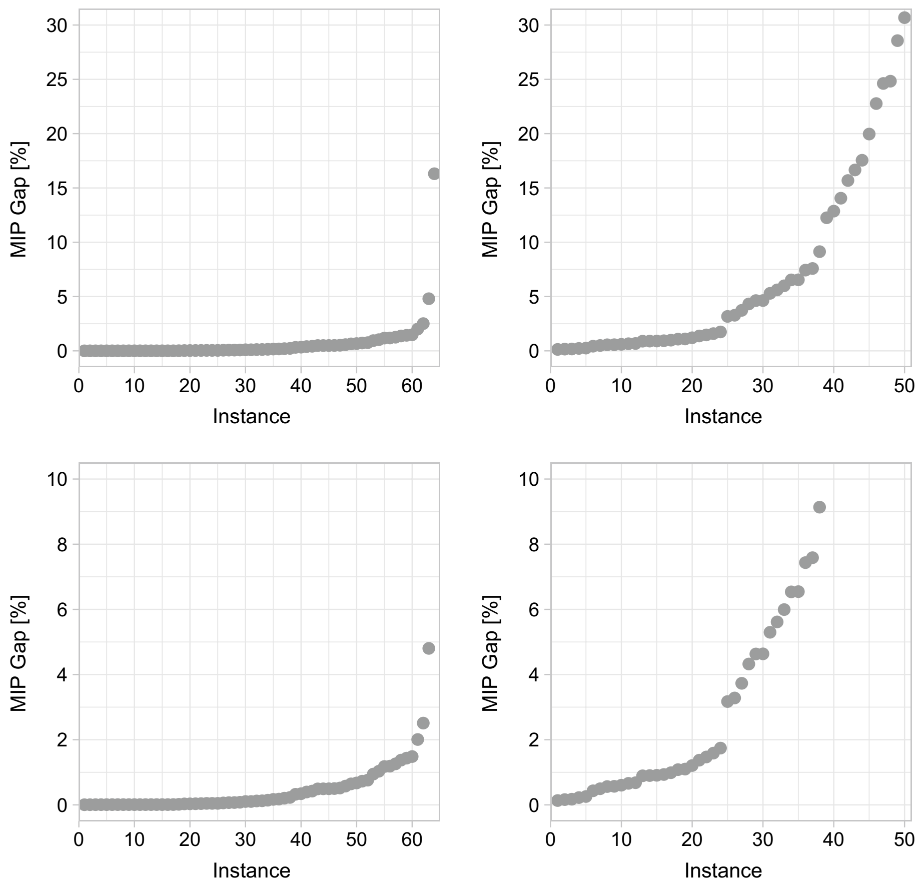

Fig. 3 MIP gaps sorted in ascending order for medium (*on the left*) and large instances (*on the right*). Depicted are 64 medium and 50 large instances, including those solved to optimality. The upper figures show the gaps for all instances, the lower figures are zoomed in for better view

In the other two categories, there are a lot more feasible but not optimal instances. Figure depicts the gaps sorted increasingly for the medium and large instances. Note that these instances include all instances for which Gurobi has found a solution, whether these solutions have been proven to be optimal or not. All but one medium instance are solved with a gap of less than 5%, that means that the best solution found by Gurobi is at most 5% worse than the optimal solution. A gap of less than 1% has been computed for 53 out of 64 instances and a gap of less than 0.1% for 29 instances. On the large instances, the gap remains bigger than 10% on twelve instances. A gap of less than 5% has been proven for 30 instances and of less than 1% for 17 instances. The smallest gap on the large instances is at approximately 0.13%.

One may reasonably expect that the bigger the instances, the more difficult it is for Gurobi to prove optimality or infeasibility. While this is certainly true when comparing medium with large instances (evidence shown in Fig. 3 and Table 4), it is not clear when only somewhat similarly sized instances are considered, e. g. only the medium instances. Since the number of provably infeasible and of "unknown" instances is very small, we will not address that relation in the context of infeasibility. Any observation on such a small data set might be pure coincidence. Concerning the proof of optimality, however, we depict the MIP gaps as a function of the number of edges for the

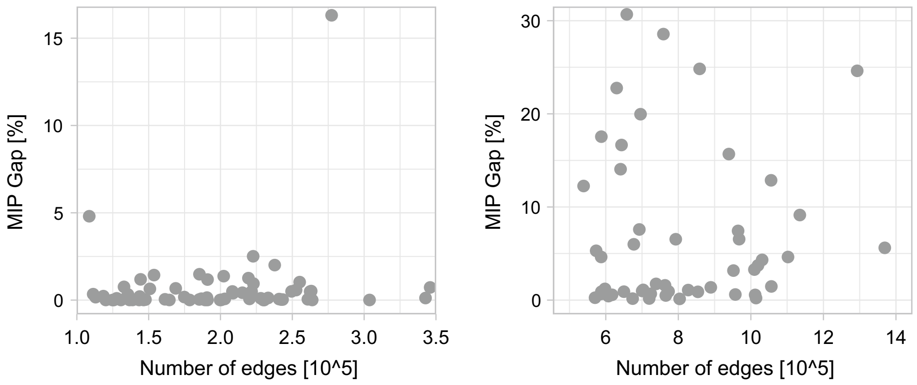

Fig. 4 The MIP gap as a function of the number of edges in medium (*left*) and large instances (*right*). Depicted are 64 medium and 50 large instances, including those solved to optimality. Be aware of the differently scaled y-axes

aforementioned 64 medium and 50 large instances in Fig. 4. For the medium instances it stands out is that the smallest instance in terms of edges has the second highest gap and that the three biggest instances have very small gaps, including the fifth lowest. Looking at the bulk of the instances, no tendency between MIP gap and number of edges is apparent. In the large instances, the gaps are more spread but also no pattern can be seen. With these pieces of evidence, we conclude that the number of edges cannot solely explain the different performance of Gurobi within the categories of instances. Presumably, the interaction of layer sizes with vertex and cable capacities also plays a role here.

As a side note: Fig. 4 shows only a small number of instances at the upper end of the edge set sizes. We have made a similar observation concerning the number of vertices of large instances in the discussion of Table 3 in Simulation Setup. We believe that the problem that many of the bigger instances in the category *large* are discarded in the process of placing strings, inverters, and transformers also persists for the medium instances (and maybe even for the small instances).

Comparison to gaps from a similar optimization problem

Having small MIP gaps provides certainty that the best solution found is close to the optimal. As we have seen, Gurobi delivers this certainty on a vast amount of instances across all categories. One might suspect that Gurobi is just very good in general at proving such small gaps. We look into simulations using Gurobi for a somewhat similar optimization problem to see if that is the case.

As we have mentioned in Contribution, a similar MILP formulation has been used for computing cable layouts of wind farms. The difference lies in the layered structure of solar farms including vertex capacities and the tree structure enforced by Eq. (8). [Gritzbach et al. (2019), ArXiv version, Fig. 6] report MIP gaps for their MILP simulations on the synthetic benchmark instances proposed in Lehmann et al. (2017). These benchmark instances each supposedly have a number of edges linear in the number of vertices, since an approach employing k-nearest neighbours is used to define the edge set. To the contrary, the edge sets of the solar farm instances proposed here grow quadratically in the number of vertices. One set of wind farm instances (originally denoted by $\mathcal{N}_3$) has between 80 and 180 turbines per instance and is therefore a bit smaller but still comparable to our small instances. Another set of wind farms (denoted by $\mathcal{N}_4$) has between 200 and 499 turbines. These wind farms are a lot smaller than our medium instances. The MIP gaps reported by Gritzbach et al. for these wind farm benchmark sets are roughly between 25% and 30% with no outliers after a running time of one day per instance. In comparison, all but one small solar farm have been solved to optimality and all but one medium solar farm have a MIP gap below 5%. Even for the large solar farms, our MIP gaps look better, at least as long Gurobi finds a feasible solution. To be fair, Gritzbach et al. have used Gurobi 8.0.0, while we use Gurobi 9.5.0. Still, this cannot explain the remarkable difference in the MIP gaps. The reason for the low MIP gaps probably lies within the structural properties of SoFaCLaP and the corresponding MILP formulation. Most notably, the layered structure of the graph yields MILP constraints that only involve a local part of the graph.

Parting notes about the evaluation

The goal of the extensive evaluation of Gurobi's performance on synthetic instances was to establish the MILP as a first solution approach to SoFaCLaP. The performance characteristics we have reported can serve as a baseline for future evaluations of other algorithmic methods, be it improvements to the MILP itself or any other way of solving SoFaCLaP. In Conclusion, we give some hints on how SoFaCLaP can be approached alternatively.

Conclusion

The Solar Farm Cable Layout Problem is, to the best of our knowledge, the first graph-theoretic model of the optimization problem to find a cost-optimal cable layout from strings to transformers in a solar farm while also deciding which of several positions for intermediate vertices such as combiner boxes are best. This model can not only be easily adapted to consider solar farms with string inverters instead of central inverters. It also allows the inclusion of further aspects for the optimization such as vertex installation costs or further constraints such as prohibiting the use of mutually exclusive candidate positions. We introduce a MILP formulation to solve SoFaCLaP and propose a process to generate synthetic benchmark instances. This process builds instances modularly and customizations can be included at various points. We populate this process with input parameters based on real-world solar farms and electrical equipment to obtain three categories of benchmark instances covering solar farms with between 120 and 1500 strings.

The simulations on the MILP formulation from A mixed-integer linear program using our synthetic benchmark instances show that Gurobi is able to efficiently solve small- and medium-sized solar farm instances; almost all small and many medium instances are solved to optimality within one day. For the larger instances, one day is not quite enough on some instances to find feasible solutions. But when Gurobi does find solutions, these are for the most part already quite acceptable in terms of the MIP gap.

Moving forward, we see two main directions. From a practical perspective, the vast amount of customization in the problem formulation and in the process of generating benchmark instances can be exploited to investigate different aspects of optimizing the layout of solar farms. From an algorithmic perspective, Gurobi could use some help to find feasible solutions. Solutions from any heuristic approach could be passed on to

Gurobi as a warmstart solution. Inspiration for algorithmic approaches can be found in similar optimization problems: Network flows in many variations are a very classical and well-studied problem in theoretical computer science. Equation (8) enforces a tree structure, so the Steiner Tree Problem could be a different point of attack. There a given subset of vertices (think: strings) of a directed graph need to be connected to a specified vertex (here: grid point beyond the transformers) using a directed tree incorporating a subset of intermediate vertices. The layers in SoFaCLaP already give a sense of direction. A similar problem is the Multi-Level Facility Location Problem, in which in a layered graph lower-level vertices need to be assigned to higher-level vertices. Any algorithmic approach can of course also stand on its own. To evaluate these, we have established the MILP as a possible baseline, which is one of the main purposes of this work.

Acknowledgements
The authors would like to thank Robin Grab for giving most valuable insights into the finer details of solar farms, Torsten Ueckerdt for fruitful discussions on graph theory and algorithms, Dorothea Wagner for proofreading parts of this work, and an anonymous reviewer for a very helpful hint on literature previously unknown to the authors.

About this supplement
This article has been published as part of Energy Informatics Volume 5 Supplement 1, 2022: Proceedings of the 11th DACH+ Conference on Energy Informatics. The full contents of the supplement are available online at https://energyinformatics.springeropen.com/articles/supplements/volume-5-supplement-1.

Author contributions
SG: conceptualization, validation, formal analysis, investigation, resources, data curation, writing—original draft, review and editing, visualization. DS: methodology, software, investigation, writing—original draft, review and editing. MW: conceptualization, validation, resources, writing—review and editing. All authors read and approved the final manuscript.

Funding
This research was funded by the Helmholtz Association under the program "Energy System Design" and by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data—Informatics Methods for its Collection, Analysis and Exploitation.

Availability of data and materials
The benchmark instances introduced in this work are available in the RADAR4KIT repository, http://www.doi.org/10.35097/676. The dataset supporting the conclusions of this article is available in the RADAR4KIT repository, http://doi.org/10.35097/683.

Declarations

Competing interests
The authors declare that they have no competing interests.

Published: 7 September 2022

References
ABB Ltd.: Photovoltaic Plants. (2019). Retrieved on January 20, 2022. https://library.abb.com/d/9AKK107492A3277
Alpiq Holding AG: Società Agricola Solar Farm 2. Retrieved on January 24, 2022. https://www.alpiq.com/power-generation/new-renewable-energy-sources/solar-energy/societa-agricola-solar-farm-2/
Bischoff M, Ewe H, Plociennik K, Schüle I (2014) Multi-objective planning of large-scale photovoltaic power plants. In: Helber S, Breitner M, Rösch D, Schön C, Graf von der Schulenburg J-M, Sibbertsen P, Steinbach M, Weber S, Wolter A (eds) Operations research proceedings 2012. Springer, Cham, pp 333–338
Bollipo RB, Mikkili S, Bonthagorla PK (2020) Critical review on pv mppt techniques: classical, intelligent and optimisation. IET Renew Power Gener 14(9):1433–1452
Brandes U, Eiglsperger M, Herman I, Himsolt M, Marshall MS (2002) Graphml progress report structural layer proposal. In: Mutzel P, Jünger M, Leipert S (eds) Graph Drawing. Springer, Berlin, pp 501–512
Chimani M, Gutwenger C, Jünger M, Klau GW, Klein K, Mutzel P (2013) The open graph drawing framework (OGDF). In: Tamassia R (ed) Handbook of graph drawing and visualization. CRC Press, Boca Raton, pp 543–569
Dutta S, Overbye TJ (2011) A clustering based wind farm collector system cable layout design. In: Power and Energy Conference at Illinois (PECI), IEEE, pp. 1–6
Every E (2022) Using Y Connectors in String Inverter Systems. Retrieved on January 24. https://solectria.com/blog/using-y-connectors-in-string-inverter-systems-part-i/

Gajda JW (2022) Solar Farms: Design & Construction. Retrieved on February 1. https://standards.ieee.org/wp-content/uploads/import/documents/presentations/nesc_workshop__solar_farms-design_construction.pdf
Garey MR, Johnson DS (1979) Computers and Intractability: a Guide to the Theory of NP-completeness. A series of books in the mathematical sciences. W. H. Freeman & Co., New York
Gritzbach S, Ueckerdt T, Wagner D, Wegner F, Wolf M (2019) Engineering Negative Cycle Canceling for Wind Farm Cabling. In: Bender, M.A., Svensson, O., Herman, G. (eds.) 27th Annual European Symposium on Algorithms (ESA 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 144, pp. 55:1–55:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl. ArXiv version available at https://arxiv.org/abs/1908.02129. http://drops.dagstuhl.de/opus/volltexte/2019/11176
Gurobi Optimization, Inc.: Gurobi Optimizer Reference Manual. (2018). Gurobi Optimization, Inc. http://www.gurobi.com
HELUKABEL GmbH: Cables and Cable Systems for Photovoltaic Installations. Retrieved on December 11, 2021. https://www.helukabel.com/media/publication/de/brochures/pv_20/PV_BROCHURE_Photovoltaik_EN~1.pdf
IRENA: Renewable power generation costs in 2020. Technical report, International Renewable Energy Agency, Abu Dhabi (June 2021). Retrieved on January 24, (2022). https://www.irena.org/publications/2021/Jun/Renewable-Power-Costs-in-2020
Kirchartz T, Rau U (2018) What makes a good solar cell? Adv Energy Mater 8(28):1703385
Lehmann S, Rutter I, Wagner D, Wegner F (2017) A simulated-annealing-based approach for wind farm cabling. In: Proceedings of the Eighth International Conference on Future Energy Systems. e-Energy '17, pp. 203–215. ACM, New York
Liu T, Yang X, Chen W, Li Y, Xuan Y, Huang L, Hao X (2019) Design and implementation of high efficiency control scheme of dual active bridge based 10 kv/1 mw solid state transformer for pv application. IEEE Trans Power Electron 34(5):4223–4238
Ljubić I (2021) Solving steiner trees: recent advances, challenges, and perspectives. Networks 77(2):177–204. https://doi.org/10.1002/net.22005
LS ELECTRIC Co., Ltd.: Photovoltaic Combiner Box. (2021). Retrieved on January 20, 2022. https://www.ls-electric.com/
Luo Z, Qin H, Cheng T, Wu Q, Lim A (2021) A branch-and-price-and-cut algorithm for the cable-routing problem in solar power plants. INFORMS J Comput 33(2):452–476
Mertens K (2019) Photovoltaics: fundamentals, technology, and practice, 2nd edn. Wiley, Chichester, West Sussex
Ortiz-Astorquiza C, Contreras I, Laporte G (2018) Multi-level facility location problems. Eur J Oper Res 267(3):791–805
Sanjay P (2022) With 2,245 MW of Commissioned Solar Projects, World's Largest Solar Park Is Now at Bhadla. Retrieved on January 25. https://mercomindia.com/world-largest-solar-park-bhadla/
SMA Solar Technology AG: SMA String-Combiner. (2020). Retrieved on January 20, 2022. https://files.sma.de/downloads/DC-CMB-U-DEN1834-V16web.pdf
SolarBOS, Inc.: Standard Recombiners. Retrieved on January 20, 2022. http://www.solarbos.com/Standard-Recombiners
SolarPower Europe: Global market outlook for solar power 2020–2024. Technical report (2020). Retrieved on January 31, (2022). https://www.solarpowereurope.org/global-market-outlook-2020-2024/
Tavana M, Santos Arteaga FJ, Mohammadi S, Alimohammadi M (2017) A fuzzy multi-criteria spatial decision support system for solar farm location planning. Energy Strategy Rev 18:93–105

Publisher's Note