A Genetic Algorithm for Finding Microgrid Cable Layouts

Max Göttlicher
max.goettlicher@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Matthias Wolf
matthias.wolf@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

Microgrids play a crucial role in the electrification of rural areas. Designing a microgrid comprises of multiple parts including finding suitable sites for generation units, sizing the components of the microdgrid, and determining the layout of the cables to connect the components. In this work we focus on the latter part, which we formalize as the Microgrid Cable Layout problem. In this problem we assume that the locations and sizes of the generators and consumers are already given. The goal is to find a cost-minimal cable layout that connects these locations and that is sufficient to handle the consumer demands. The cables may be of different cable types, and we may connect multiple cables not only at the locations of the generators and consumers but also at any other point. Microgrid Cable Layout is a strongly $\mathcal{NP}$-hard, non-linear optimization problem.

We present a hybrid algorithm for the Microgrid Cable Layout problem, which employs a genetic algorithm for optimizing the topology of the layout and a heuristic for assigning the cables to the edges of the topology. An evaluation on a set of benchmark instances indicates that this algorithm is able to find good solutions within a short amount of time. We further evaluate the performance of the algorithm in a case study on a real-world microgrid in the Democratic Republic of the Congo.

CCS CONCEPTS

• **Theory of computation** → *Algorithm design techniques*; *Graph algorithms analysis*.

KEYWORDS

Microgrids, Cable Layout, Genetic Algorithm, Cable Types, Steiner Points

1 INTRODUCTION

In 2015 the United Nations have presented an agenda for sustainable development [28], which formulates 17 goals to reach until 2030. Sustainable Development Goal 7 is to "ensure access to affordable, reliable, sustainable and modern energy for all" [28]. Part of this goal is to provide everyone with access to electricity. According to a case study in rural Kenya [17] access to electricity significantly improves the productivity per worker by up to 200 %. Providing access to electricity is particularly challenging in rural areas in the global south. According to the 2021 report on the sustainable development goals [27] there were still 471 million people in rural areas of sub-Saharan Africa without access to electricity in 2019.

In particular in these rural areas microgrids can play a crucial role in providing access to electricity. They do not require possibly long and costly transmission lines to the main power grid to be built. Instead, they can be installed anywhere and work completely autonomously provided that there is an energy source available. Traditionally, diesel generators were used to power microgrids, but also more sustainable sources like small hydropower plants, photovoltaic systems, or wind turbines can be used. They do not require expensive fuel, but can be less reliable.

Designing a microgrid is a complex task (see e.g., the guide by Sumanik-Leary et al. [26]), and one needs to answer many questions such as: Where should the microgrid be built? Which places shall be connected to the microgrid? How much power is needed? Which generators shall be placed where? How does the cable layout look like? In this work we focus on the last question, which deals with routing the cables efficiently. We formalize this problem as the Microgrid Cable Layout problem. In this problem we assume the locations and sizes of the generators and consumers to be fixed by a previous step of the microgrid design process. The goal is to find a cost-minimal cable layout that connects these locations subject to certain electrical constraints. There are multiple cable types to choose from, and we may introduce additional distribution nodes, which may be placed not only at the location of a generator or consumer in the input but also at any other point in the plane. Microgrid Cable Layout further includes costs for poles, which need to be placed to support the cables. We give a formal definition of this problem in Section 2. Note that this problem should be thought of as a part of the full microgrid design problem. While solving the full problem, multiple instances of the Microgrid Cable Layout problem may need to be solved.

We present a hybrid genetic algorithm for the Microgrid Cable Layout problem, with which we aim to achieve the following two main goals.

(1) The algorithm should find good solutions reasonably quickly such that it can be included as a part of a software that considers the full microgrid design problem.

(2) The algorithm should be flexible, i.e., it should be easy to add further constraints that needs to be considered. It is infeasible to gather a set of constraints beforehand that suffices for all possible use cases.

1.1 Related Work

Determining good designs for microgrids encompasses a lot of different steps, which range from determining the size of the components over finding a good topology to computing a suitable strategy to control the operation of the microgrid. For an overview on this large range of topics we refer to two surveys by Al-Ismail [2] and by Gamarra and Guerrero [10]. In the following we focus on works that are related to the cable layout design aspect of migrogrid design.

Lambert and Hittle [21] tackle the problem of designing a good low- and medium voltage grid layout for the electrification of a rural village. They consider the design of the low-voltage layout and of the medium-voltage layout as two separate layers of their optimization algorithm. They use a single cable in each layer and limit resistive losses by constraining the maximum distance between demand points and transformers. For the low-voltage layout they propose to use an approach based on simulated annealing. Their medium-voltage layout is always a minimum spanning tree between the transformers, which connect the two voltage levels. The number and placement of the transformers is part of their optimization problem. However, these transformers may only be placed at demand nodes.

Kahveci et al. [16] present a heuristic for finding cable layouts for microgrids. Their heuristic includes three steps: computing a minimum spanning tree, adding Steiner points in triangles if they reduce the costs, and finding clusters that may form independent islands. They insert Steiner points to reduce the total network length but do not take into account different cable types or electric constraints.

Corigliano et al. [8] present an approach to identify good electrification strategies in rural ares. Within this approach they apply an algorithm to compute good grid layouts considering the terrain. They first discretize the problem by defining a grid graph with weights that represent the difficulty of the terrain. Then, they compute a minimum spanning tree between the populated vertices (the metric they use is not stated explicitly however), and finally, they replace the edges in the tree by shortest paths in the grid. Similar approaches were used in case studies in Nigeria [5] and in the Philippines [4]. Another case study for a rural an urban microgrid in India [25] uses Homer [1] to optimize the sizing of the components but does not consider the geographic layout.

In contrast to the works above, Nolan et al. [24] consider multiple cable types. They present a genetic algorithm using Prim-predecessor encoding [22]. Unlike our algorithm, their algorithm does not consider adding additional points to the network. That is, their resulting network topology is always a spanning tree on the input points.

A complementary version of the microgrid layout problem we consider in this work is studied by Vallem and Mitra [29] and Vallem et al. [30]. They consider the problem where to place distributed generation units given a grid topology. They solve this problem with a simulated annealing approach. Note that this approach is not directly comparable to the one we use since it solves a different problem. We assume the locations of the generation units to be fixed in order to include cases where, e.g., the generators already exist or where there is only one sensible place for a small hydroplant.

The Microgrid Cable Layout problem is closely related to other geometric layout optimization problems such as the Euclidean Minimum Steiner Tree problem.[1] A survey on the history of this problem is written by Brazil et al. [7]. Genetic algorithms have been used both for this problem [3] and the Capacitated Minimum Spanning Tree problem in graphs Jesus et al. [14], another related network design problem.

1.2 Contribution and Outline

In this work we present a hybrid genetic algorithm for the Microgrid Cable Layout problem, which splits the problem into two parts. The outer part aims at find a good cable layout topology (represented as an undirected graph), for which we use a genetic algorithm. To assess the quality of a topology, the inner part is to assign suitable cable types to the edges of the topology. For the inner part we design multiple heuristics and formulate a mixed-integer linear program, which yields optimal cable assignments for given topologies. The combination of the algorithms for the two parts is able to find very good cable layouts within seconds. This makes the algorithm viable to be used within a larger framework that optimizes the whole micogrid design including siting and sizing of the equipment. Moreover, the chosen approach is very flexible in the sense that additional constraints or changes to the cost function may be included with little effort, which is one of our main design goals.

We formally define the Microgrid Cable Layout problem and give a short analysis of its computational complexity in Section 2. In Section 3 we then describe our algorithmic approach in detail, including both the genetic algorithm for finding a topology and the algorithms for the cable assignment. We evaluate the algorithms in Section 4 on a set of benchmark instances. This evaluation includes a comparison of the layout computed by the hybrid genetic algorithm with the planning of a real-world microgrid in the Democratic Republic of the Congo. We conclude the work with a summary and outline possible further research directions in Section 5.

2 THE MICROGRID CABLE LAYOUT PROBLEM

As input we are given a set P of points in the plane, which represent the positions of the available generators and the consumers. Each point $p \in P$ has a maximum power generation $g(p) \in \mathbb{R}_{\geq 0}$ and a maximum power demand $d(p) \in \mathbb{R}_{\geq 0}$. We may assume that for each point p one of $g(p)$ and $d(p)$ is non-zero since we can ignore all points where both are zero. A point $p \in P$ is a *generator* if $g(p) > 0$ and a *consumer* if $d(p) > 0$. Moreover, we are given a set C of cable types. Each cable type $c \in C$ has a cost per meter $c_{\text{line}}(c)$, a resistivity $\rho(c)$ measured in $\Omega\,\text{m}^{-1}$, and a thermal capacity $\text{cap}(c)$ measured in kW, which represents the maximum power that can be transferred via a cable of type c.

[1] The (Euclidean) Minimum Steiner Tree problem is often called the Steiner Minimum Tree problem and abbreviated by (E)SMT to distinguish it from the Minimum Spanning Tree problem (MST). We stick to the name *Minimum Steiner Tree* and prevent ambiguities by always writing the full problem name.

A *topology* is an undirected graph $G = (V, E)$ with $P \subseteq V \subseteq \mathbb{R}^2$, i.e., the vertices are a set of points in the plane that contains P. Borrowing from the nomenclature for Steiner trees, we call the points in $V \setminus P$ *Steiner points*. All topologies that we consider in this work are trees, i.e., they are connected and contain no cycles. The length $\ell(e)$ of an edge $e \in E$ is the euclidean distance between its endpoints. A *cable assignment* is a function $a\colon E \to C$. Each edge is assigned exactly one cable type. Note however that placing multiple cables in parallel can be modeled by including cable types in C that represent placing multiple cables of (actual) cable types in parallel.

Typically, the installed power generation capacity, which is equal to the sum of the maximum power generations at all points, is less than the total maximum demand. That is, based on the generation capacity, we cannot fulfill all demands if all points try to consume their maximum demands at the same time. But we want to prevent the grid infrastructure and especially the cables from being the limiting factor instead of the generation capacity. Hence, we want to design the grid such that it allows for all possible distributions of generation and consumption within the bounds given by the maximum generations and demands. This does not only include the cable power rating but also losses of power and voltage within the network [26].

Therefore, we compute for each directed edge $(v, w) \in \vec{E}$ (where $\vec{E} = \{(v, w), (w, v) \mid vw \in E\}$) the maximum power $p(v, w)$ that may need to be transmitted via (v, w) from v to w. Recall that for $p \in P$ the values $g(p)$ and $d(p)$ give bounds for the maximum generation and maximum demand at p, respectively. The actual generation and consumption may (and usually will) differ from these values. These two functions can naturally be extended to V by setting their values to 0 outside of P. As G is a tree, removing an edge vw disconnects G into two components G^v_{vw} and G^w_{vw}, where the former contains v and the latter contains w. The maximum flow $p(v, w)$ on the edge from v to w is limited by both the total generation in G^v_{vw} and the total demand in G^w_{vw}. We have

$$p(v, w) = \min \left\{ \sum_{x \in V(G^v_{vw})} g(x), \sum_{x \in V(G^w_{vw})} d(x) \right\}. \tag{1}$$

Note that in general $p(v, w)$ and $p(w, v)$ may differ. We can compute these values for all edges in $O(|V|)$ time by performing a single depth-first search of G; see Appendix A for details. Since we want the cables to not limit the amount of power transmitted, we must ensure that their capacity is large enough in any cable assignment a, i.e., for every edge $vw \in E$ we require

$$\max\{p(v, w), p(w, v)\} \leq \operatorname{cap}(a(e)). \tag{2}$$

We further want the line losses and the maximum voltage drop along any generator-consumer-path to be small. For a given topology $G = (V, E)$ and a cable assignment a we bound these values as follows. The maximum voltage drop $U_d(v, w)$ along a single edge vw from v to w in a three-phase transmission system is given by $r(vw) \cdot i(v, w)$, where $r(vw) = \ell(vw)\rho(a(vw))$ is the resistance of the edge according to the cable assignment a and $i(v, w) = p(v, w)/(\sqrt{3}U \cos\varphi)$ is the maximum current on vw from v to w. Here, U is the grid voltage, φ is the load factor. The voltage drop along a path Q is then bounded by

$$\begin{aligned} U_d(Q) &= \sum_{(v,w) \in \vec{E}(Q)} U_d(v, w) \\ &= \sum_{(v,w) \in \vec{E}(Q)} \ell(vw)\rho(a(vw)) \cdot \frac{p(v, w)}{\sqrt{3}U \cos\varphi}, \end{aligned}$$

where $\vec{E}(Q)$ are the edges of Q in the same direction as Q.

The maximum power loss $p_l(vw)$ on a single edge $vw \in E$ is $3r(vw) \cdot i_{\max}(vw)^2$, where $i_{\max}(vw) = \max\{i(v, w), i(w, v)\}$ is the maximum current along the edge in either direction. The sum of all these values is then an upper bound for the total power loss P_l.

$$\begin{aligned} P_l &= \sum_{vw \in E} p_l(vw) \\ &= \sum_{vw \in E} \ell(vw)\rho(a(vw)) \cdot \left(\frac{\max\{p(v, w), p(w, v)\}}{U \cos\varphi} \right)^2. \end{aligned}$$

Our goal is find a topology G with a cable assignment a of minimum costs such that the maximum voltage drop along any generator-consumer-path is at most $\alpha_{\text{drop}} \cdot U$, and the total line losses P_l are at most a factor of α_{loss} of the total maximum generation and consumption. More formally, we want for each path Q from a generator to a consumer that

$$U_d(Q) \leq \alpha_{\text{drop}} \cdot U, \tag{3}$$

and

$$P_l \leq \alpha_{\text{loss}} \cdot \min \left\{ \sum_{p \in P} g(p), \sum_{p \in P} d(p) \right\} \tag{4}$$

We call any cable assignment that satisfies Eqs. (2) to (4) a *feasible* cable assignment.

A topology $G = (V, E)$ with cable assignment a incurs certain costs. In this work we consider the costs for the cables, the poles the cables are placed on, and the equipment needed to connect the cables together. We assume that these include both the costs of the materials and the installation. Some costs like the costs of the generators depend neither on the topology nor on the cable assignment and can be ignored for the purpose of determining a good topology and cable assignment. Note that the algorithms described in the following section are designed to be flexible such that additional costs may be introduced easily if the need arises.

To compute that cable costs, recall that each cable $c \in C$ has a cost per meter $c_{\text{line}}(c)$. Hence, we have

$$\text{cost}_{\text{cables}}(G, a) = \sum_{e \in E} \ell(e) \cdot c_{\text{line}}(a(e)).$$

To support the cables we need to place poles such that two poles along a line are at most $d_{\max}$ (measured in m) apart. We further have one pole at each vertex. We consider a fixed cost of c_{pole} per pole. We have

$$\text{cost}_{\text{poles}}(G) = c_{\text{pole}} \cdot \left(|V| + \sum_{e \in E} \left\lceil \frac{\ell(e)}{d_{\max}} \right\rceil - |E| \right).$$

Further we need equipment at each vertex to connect the incident lines. The cost of this equipment at $v \in V$ is determined by a function c_{equip}, which is given as an input. This function depends

on the degree $\deg_G(v)$ of v and the maximum power on any edge incident to v, which is defined as

$$\hat{p}_G(v) = \max\{p(v,w), p(w,v) \mid w \in N_G(v)\}.$$

We then have

$$\text{cost}_{\text{vertex}}(G) = \sum_{v \in V} c_{\text{equip}}(\deg_G(v), \hat{p}_G(v)).$$

In our experiments we assume that c_{equip} depends linearly on its parameters, i.e., we have

$$c_{\text{equip}}(x,y) = \alpha + \beta \cdot x + \gamma \cdot y,$$

for some values $\alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}$. In total, the costs are

$$\text{cost}(G,a) = \text{cost}_{\text{cables}}(G,a) + \text{cost}_{\text{poles}}(G) + \text{cost}_{\text{vertex}}(G).$$

The Microgrid Cable Layout problem can be summarized as follows. We are given a set of points $P \subseteq \mathbb{R}^2$ with generation function $g\colon P \to \mathbb{R}_{\geq 0}$ and demand function $d\colon P \to \mathbb{R}_{\geq 0}$, a set of cables C with their properties (c_{line}, ρ, cap), and the properties of the desired grid (voltage U, load factor φ, maximum voltage drop factor α_{drop}, maximum power loss factor α_{loss}, maximum distance between poles $d_{\max}$, and costs of poles c_{pole} and other equipment c_{equip}). Find a tree topology $G = (V, E)$ and a cable assignment $a\colon E \to C$ that satisfy Eqs. (2) to (4) and minimize $\text{cost}(G, a)$.

This problem generalizes the strongly $\mathcal{NP}$-hard Euclidean Minimum Steiner Tree problem [7, 11]; see Appendix B for details of the reduction. Hence, Microgrid Cable Layout is strongly $\mathcal{NP}$-hard as well. In fact, even finding a cost-minimal cable assignment given a fixed topology is already $\mathcal{NP}$-hard; see Appendix C.

3 ALGORITHMS

We want to compute good solutions for the Microgrid Cable Layout problem. A solution consists of a topology and an assignment of cables to the edges of the topology. Computing a topology is related to the Euclidean Steiner Tree problem and various constraint spanning tree problems (e.g., the Constrained Minimum Spanning Tree), for which genetic algorithms have proven to work well [3, 14, 23]. However, one major difference to these problems is that we also need to find a cable assignment. The introduction of Steiner nodes makes the problem difficult to solve using a general purpose optimizer such as Gurobi as they result in a non-convex problem. Current combinatoric algorithms for the Euclidean Steiner Tree problem [15] do not work with weighted edges and thus cannot model different cable types. We therefore adapt the genetic operators for these problems to the Microgrid Cable Layout problem and use a hybrid approach, in which a genetic algorithm computes the topology (see Section 3.1) but not the cable assignments. We base the operators of the genetic algorithm on the genetic algorithms by [3, 23]. Within the selection phase our genetic algorithm uses one of several cable assignment algorithms (see Section 3.2) to evaluate the quality of the computed topologies.

One could also envision a combined genetic algorithm that computes both the topology and the cable assignment. We describe how to include the cable assignment directly in our genetic algorithm in Section 3.4.

3.1 Genetic Algorithm for the Topology

A genetic algorithm maintains a set of individuals (the *population*) and repeatedly creates new individuals by either modifying the individuals slightly (*mutation*) or by combining parts of two individuals (*crossover*). Based on a *fitness function* it then selects some old and some new individuals to form the next *generation*; see the book by Kramer [19] for a more detailed introduction to genetic algorithms.

The population of our genetic algorithm consists of topologies. Each topology is represented by a singly-linked adjacency list. To assess the quality of these topologies, we compute cable assignments according to one of the approaches that we explain in Section 3.2. These cable assignments then allow us to compute the costs of the topologies (with these cable assignments).

3.1.1 Initialization. We want the topologies in our initial population to have a reasonable structure. In particular, we want them to be trees without edge crossings, since edge crossings are likely suboptimal. However, we do want a variety of such trees in order to be able to explore the solution space. To achieve both goals we compute random spanning trees on a planar triangulation of the input points. More precisely, we use the Delaunay triangulation as the planar triangulation, and we compute a random spanning tree by first choosing an order of the edges uniformly at random and then adding edges in this order to our topology unless the insertion of an edge forms a cycle.

3.1.2 Crossover. All our crossover operations work on two *parent* topologies $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. In the *separate crossover* we choose which Steiner points and edges of the two parent topologies to keep in two separate phases. In a third phase we ensure that the created topologies are connected. Let S_1 and S_2 be the Steiner points in the two parent topologies. In the first phase we select a subset S of $\max\{|S_1|, |S_2|\}$ Steiner points from $S_1 \cup S_2$ uniformly at random, which we keep. Then, $V = P \cup S$ is the set of vertices of the new topology. Not keeping all Steiner points ensures that the number of Steiner points remains bounded.

In the second phase we select a subset of the edges to insert. This phase is based on an operator for constrained balanced trees [23]. However, due to the first phase an edge of a parent may have endpoints that are not present in V. We therefore map each edge $vw \in E_1 \cup E_2$ to the edge $\mu(vw)$ between the point of V closest to v and the point of V closest to w and work with the mapped edges. First, we insert all of $\mu(E_1)$. Then, we select a random subset E_2' of $\mu(E_2) \setminus \mu(E_1)$ by first selecting the desired cardinality k and then k edges uniformly at random. We insert each of these edges into the child topology, breaking each cycle C that occurs by deleting random edge on C except the newly added edge.

Finally, we check in the third phase whether the resulting topology is connected. If not, we randomly add edges between the connected components until the topology is connected.

In the *subtree crossover* we aim to maintain (possibly optimal) local substructures. The main idea is to take one subtree of G_2 and to add it to G_1 removing all edges of G_1 in this process that would cause cycles. We select a subtree of G_2 by first choosing an edge $vw \in E_2$ uniformly at random and then picking the subtree $T = (V_T, E_T)$ to one side of vw. We initialize the child topology with $(V_1 \cup V_T, E_T)$.

Then, we consider the edges of G_1 in a random order, and insert those edges that connect disconnected components. Since G_1 is connected, the resulting topology is connected as well.

3.1.3 Mutations. Note that neither the initialization nor the two crossover operators introduces new Steiner points. This only happens in the mutation operators, which we describe next. All mutation operators operate on a single topology $G = (V, E)$.

The *close endpoint mutation* is an adaptation of a mutation operator for diameter constrained spanning tree problems [23]. The operator is parameterized by a mutation probability π_e. Before the mutation we temporarily insert Steiner points in the middle of all edges of G resulting in a topology $G' = (V', E')$. Then, each edge is considered individually and mutated with probability π_e. To mutate an edge vw, we randomly select on of its endpoints x as the first endpoint of the new edge and remove vw. We then select another point $y \in V'$ as the second endpoint, where the probability that a point z is chosen is proportional to $\text{dist}(x, z)^{-2}$. However, the insertion of xy may create a cycle and disconnect the topology. As in separate crossover, we break cycles by randomly deleting an edge on the cycle. To repair a disconnected topology, we re-insert vw. Finally, we revert the introduction of many Steiner points in the beginning by contracting all Steiner points of degree at most 2; see Section 3.1.4 how to do this efficiently.

To move Steiner points we apply the following two operators. Both decide for each Steiner point independently whether to move it with probability π_s. They differ in how they move the Steiner point then. The *nudge mutation* (called *compress* by Barreiros [3]) moves the Steiner point to random point on an incident edge. The *link-length minimization* mutation tries to reduce the total weighted length of the incident edges, where we set the weight as the unit costs of the assigned cable types before the modification. We employ an iterative algorithm for finding the generalized Weber point [20]; the number of iteration is another input parameter of this operator. Note that link-length minimization disregards the poles on the incident edges. Moreover, during the minimization we ignore the option of assigning different cable types. Both restrictions may cause the iterative algorithm to converge to a non-optimal position.

3.1.4 Pruning Steiner points. The operators may cause the topologies to contain Steiner points of degree at most 2. Such Steiner points are never necessary in an optimal topology, and they can be removed by contracting an incident edge. They can be determined by a single traversal of the tree in $O(|P|)$ time. Note that removing points also reduces the number of options the operators have, e.g., for deciding where to move an edge in the close endpoint mutation. However, we artificially subdivide the edges before performing such operations. Hence, pruning Steiner points in this way has only a small impact on these operators. Pruning has the benefit that it keeps the number of Steiner points bounded by a constant factor of the number of points. Otherwise, the number of Steiner points could increase with each iteration, which degrades the performance of the operators.

3.1.5 Selection. At the end of every iteration we have a set of parent topologies and a new set of child topologies, which have been created by applying the operators. To choose which topologies to keep, we need to assess their quality, which we interpret as the total cost of the topology. Since the cost depends both on the topology and a cable assignment, we first compute a cable assignment for each topology; we present different exact and heuristic methods for this in Section 3.2. We then use tournament selection [19] with elitist reinsertion to determine the set of topologies to keep. More precisely, 95 % of the new population is obtained by repeatedly selecting a random subset of k child topologies and inserting the cheapest of these. The remaining 5 % are the 5 % best parent topologies. Keeping these implies that the quality of the best topology in the population does nott decrease over time.

3.2 Cable Assignment

The formulation of the constraints of the cable assignment in Section 3.2 suggests formulating the cable assignment problem as a mixed-integer linear program (MILP). In this formulation we have one binary variable $z(e, c) \in \{0, 1\}$ per edge $e \in E$ and cable type $c \in C$. All other values that occur in the formulation are constant for a given topology and can be precomputed; see Appendix A for a linear-time computation of the power flows on the edges. We ensure that exactly one cable type is selected per edge with the following constraints.

$$\sum_{c \in C} z(e, c) = 1, \qquad \forall e \in E. \tag{5}$$

The resistance of an edge $e \in E$ can then be described by the expression

$$R(e) = \sum_{c \in C} z(e, c) \cdot \ell(e)\rho(c). \tag{6}$$

Replacing $\ell(e)\rho(a(e))$ with the expression for $R(e)$ in Eq. (6) above, Eqs. (3) and (4) give linear constraints that ensure that the voltage drops and the line losses stay within the desired range. Note that it suffices to formulate the voltage drop constraint only for maximal paths from a generator to a consumer, i.e., they are not part of any longer path from a generator to a consumer.

To ensure that the capacity of the selected cables is sufficiently large, we use the following constraints, which adapt Eq. (2).

$$\max\{p(v, w), p(w, v)\} \leq \sum_{c \in C} \text{cap}(c) \cdot z(e, c) \qquad \forall e \in E. \tag{7}$$

Finally, we want to minimize the cost of the cable assignment $\text{cost}(G, a)$, which can be described by

$$\sum_{e \in E} \sum_{c \in C} \ell(e) \cdot c_{\text{line}}(c) \cdot z(e, c). \tag{8}$$

Note that this objective function just includes the costs that depend on the cable assignment and ignores the costs that only depend on the topology. In total, we obtain a mixed-integer linear program with $|V| \cdot |C|$ binary variables.

In the genetic algorithm that computes the topology, we repeatedly compute cable assignments for topologies in order to assess their qualities. Hence, we need a fast way to compute a good cable assignment, and even though our experiments show that the MILP can typically be solved within seconds (see Section 4.1), this is not fast enough for our purpose. We therefore employ a heuristic, which extends the work by Kraft [18] and works as follows.

As for formulating the MILP, we first determine the maximum flows on all edges ($p(v, w)$ and $p(w, v)$, as well as $p_{\max}(vw) =$

$\max\{p(v,w), p(w,v)\}$ for all $vw \in E$). We further compute for each edge vw the length $\hat{\ell}_{\text{path}}(vw)$ of the longest path from a generator to a consumer that goes through vw. These lengths can be computed in $O(|V|)$ time by traversing the tree twice; see Appendix D for details.

We then allocate each edge vw a portion of the allowed voltage drop that is proportional to the length of the edge relative to the length of the longest path with vw. That is, we set

$$U_d^{\max}(vw) = \frac{\ell(vw)}{\hat{\ell}_{\text{path}}(vw)} \alpha_{\text{drop}} U.$$

Similarly, we allocate a portion of the allowed line losses proportional to the edge length, i.e.,

$$p_l^{\max}(vw) = \frac{\ell(vw)}{\sum_{e \in E} \ell(e)} \alpha_{\text{loss}} \cdot \min \left\{ \sum_{p \in P} g(p), \sum_{p \in P} d(p) \right\}.$$

We then assign the cheapest cable to vw that has sufficient capacity and complies with the bounds above, i.e., we have for the final assignment a and all $vw \in E$

$$U_d(vw) \leq U_d^{\max}(vw)$$
$$p_l(vw) \leq p_l^{\max}(vw).$$

Since these requirements only strengthen the constraints in Eqs. (2) to (4), the cable assignment a is feasible.

Lemma 3.1. *The heuristic computes a feasible cable assignment in $O(|V|)$ time.*

This cable assignment may be improved upon by iterating over the edges and greedily replacing the currently selected cable by a cable of the cheapest cable type that suffices to satisfy all constraints. We distinguish between two variants based on the order in which we iterate over the edges. Since the cost of an edge in proportional to its length, we expect greater cost saving when we can select a cheaper cable on a longer edge. Therefore, we consider the edges in the order from the longest to the shortest. The running time of the greedy improvement phase is dominated by computing the maximum generator-consumer-paths. Since there may be a quadratic number of such paths, each of linear length, we obtain a cubic running time for the greedy improvement in the worst case.

Lemma 3.2. *The heuristic with greedy improvement computes a feasible cable assignment in $O(|V|^3)$ time.*

3.3 Summary of Hybrid Genetic Algorithm

To summarize, our algorithm consists of the main part, in which the topology is optimized by a genetic algorithm (see Section 3.1). It delegates computing suitable cable assignments to one of the algorithms in Section 3.2. In principle, we may use any of the three algorithms presented above (MILP, heuristic, heuristic with greedy improvement). However, we need to compute a cable assignment once per topology and iteration. It is therefore computationally infeasible to solve the MILP every time we need a cable assignment. We thus only consider the two heuristic cable assignment methods for use during the execution genetic algorithm. However, we do solve the MILP once at the end for the best topology found by the genetic algorithm.

3.4 Combined Genetic Algorithm for Topology and Cable Assignment

Instead of the hybrid scheme of combining a genetic algorithm for the topology and a heuristic for the cable assignment, one could also envision a genetic algorithm that includes the cable assignment. To investigate the feasibility of such an algorithm, we extend the genetic algorithm presented above as follows. An individual now contains a cable assignment in addition to the topology. Note that the cable assignment does not need to be feasible, i.e., it may violate one of the constraints in Eqs. (2) to (4). We therefore penalize infeasible individuals in the fitness function. More precisely, when comparing two individuals, we first compare the number of cables that violate the capacity constraint (Eq. (2)), then the violation of the power loss constraint (Eq. (4)), then the violation of the voltage drop constraint (Eq. (3)), and only then the costs of the cable assignments. In particular, this ensures that a feasible cable assignment is always better than an infeasible one.

The combined genetic algorithm includes all operators of the hybrid genetic algorithm (see Section 3.1). However, they need to be slightly adapted in order to deal with the cable assignment. Whenever they move edges, they now simply keep the cable type the same.

Additionally, we introduce a new mutation operator called *cable mutation*, which changes the cable assignment but not the topology. It is paramterized by a mutation probability π_c. The operator performs the following steps independently for all edges. With probability π_c it changes the cable type of the current edge to a cable type chosen uniformly at random. We often have cable types than are derived from placing the same cable type in parallel. If we know that this is the case, we determine whether to increment or decrement the number of cables in parallel with probability π_c; the choice between incrementing and decrementing is done uniformly at random. If we did not increment or decrement (which happens with probability $1 - \pi_c$), we halve or double the number of cables in parallel with probability π_c; again the direction of the change is chosen uniformly at random.

4 EVALUATION

We implemented the algorithms described in the previous section in Rust using a modified version of genevo[2] as a framework for our genetic algorithm. We parallelized the application of the genetic operators by using rayon[3]. The experiments were performed on a SuperMicro H8QG6 Server with four 12-core AMD Opteron 6172 processors clocked at 2.1 GHz with 256 GB RAM running OpenSUSE Leap 15.3.

Due to a lack of suitable input instances, we resorted to adapting instances for the related Minimum Steiner Tree problem, which were supplied in the DIMACS 11 implementation challenge [9]. These benchmark instances come in sets of different sizes. In our evaluation we chose the sets of sizes 10, 20, 50 and 100. We randomly selected between 5 % and 20 % of the points as generators with a maximum generation of 80 kW. All other points are consumers with a maximum consumption of either 4 kW (with 50 %

[2] https://github.com/innoave/genevo
[3] https://github.com/rayon-rs/rayon

Table 1: The base cable types we use in the evaluation [18].

cross section [mm]	cost [USD m^{-1}]	capacity [A]	resistivity [Ω km^{-1}]
16	2.5	66	1.91
35	5.0	132	0.87
70	8.0	205	0.44
95	11.18	245	0.32

probability), 10 kW (30 % probability), or a uniformly randomly chosen value in [15, 75] (20 % probability). This distribution was chosen to resemble a collection of many households, some smaller and few larger workshops. We assume a cost of 180 USD per pole and a maximum distance between two poles of 50 m. Each distribution point incurs a fixed cost of $\alpha = 100$ USD plus $\beta = 200$ USD per connection and an additional $\gamma = 30$ USD kW^{-1}. The properties of the cables are given in Table 1. In addition, we assume that multiple cables of the same type may be placed in parallel to each other. We further set $\alpha_{\text{loss}} = \alpha_{\text{drop}} = 0.1$ and assume $\cos\varphi = 0.8$. We refer to the benchmark set with $k \in \{10, 20, 50, 100\}$ points by $\mathcal{B}_k$.

4.1 Cable Assignment

In this section we analyze the quality of the cable assignment heuristics compared to the optimal assignment. We used the points from the first 51 instances in our benchmark sets $\mathcal{B}_k$ and evaluated the heuristic and its greedy improvement. For each instance we used three different topologies for cable assignment: the minimum spanning tree, the minimum Steiner tree and a random spanning tree consisting of edges in the Delaunay triangulation of the terminals. We compared the results to the minimum cost cable assignment we obtained by solving the MILP using Gurobi [13]. We repeated this 5 times for each set of points.

We find that in all test instances from $\mathcal{B}_{50}$ and $\mathcal{B}_{100}$ the heuristic always yielded the minimum cost cable assignment on all three topologies. On the test instances from $\mathcal{B}_{10}$ the heuristic was optimal on all minimum spanning tree and minimum Steiner tree topologies. The results were optimal in 80 % of all random topologies with the most expensive being around 1.3 times more expensive than the optimum. Using the greedy improvement this number increased further to above 90 % with the most expensive being 1.1 times more expensive than the optimum. The heuristic did not perform as well on $\mathcal{B}_{20}$, the results of which are presented in Fig. 1. While there are still instances where the heuristic yielded the optimum result, it performed particularly bad on the random spanning trees. There, the base heuristic found the optimum solution in only 3 out of all 255 runs. Using the greedy improvement could, however, increase this to 49 runs or 19 %. With all three topology variants we see a large decrease in the final cost when using the greedy improvement phase.

This behavior does not seem to depend on the instance size but on the distribution of edge lengths. We scaled the coordinates of the instances in $\mathcal{B}_{20}$ by 0.25 and found that this resulted in the heuristic always finding the optimum solution. Similarly, we scaled $\mathcal{B}_{50}$ and $\mathcal{B}_{100}$ by 4 and 10, respectively, and got results similar to the ones in Fig. 1; see Fig. 6. These scaling factors were chosen to

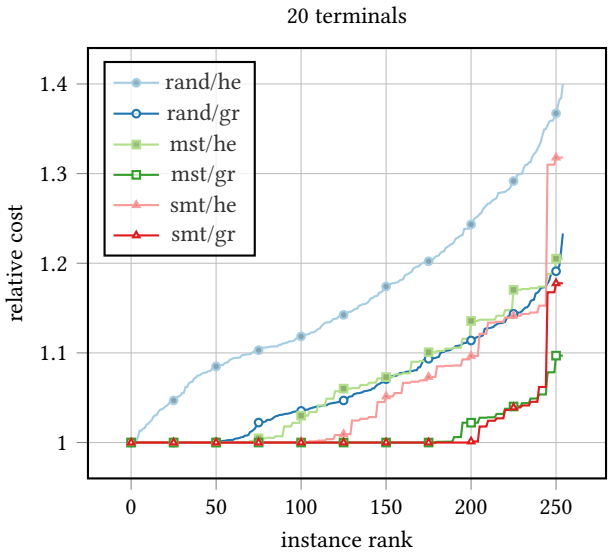


Figure 1: Distribution of the cost of the heuristic cable assignments relative to the optimal assignment on different generated topologies on instances from $\mathcal{B}_{20}$. Markers are placed every 25 runs corresponding to 5 instances repeated 5 times each.

Table 2: Average running times of each cable assignment method on different benchmark sets $\mathcal{B}_k$.

	mean running time (µs)		
k	heuristic	greedy	gurobi
10	27	53	58 345
20	62	108	223 979
50	120	332	908 331
100	220	1 094	5 070 330

resemble the differences in average edge length among the instances of the different benchmark sets. With fewer terminals than $\mathcal{B}_{50}$ and $\mathcal{B}_{100}$, but covering the same area, the average edge length in $\mathcal{B}_{20}$ is higher than in the other two instances. The longer the edges become on average, the more the constraints for the voltage drops and line losses become relevant. But these constraints are non-local, and the heuristic complies with stricter, localized versions of them. This is the reason why the heuristic may fail to find optimal cable assignments. In contrast, the capacity constraints are already local and are handled optimally by the heuristic. Moreover, they are independent of the edge lengths.

In terms of running time the heuristic performed best, closely followed by the heuristic with greedy improvement. The mean running times of the algorithms are given in Table 2. For the smaller instances adding the greedy improvement phase roughly doubles the running time. But we can also clearly see the super-linear growth of the running time for the greedy improvement that is predicted by the theoretical analysis in Lemma 3.2. The long running time

Table 3: Comparison of different mutation rates on $\mathcal{B}_{20}$ and $\mathcal{B}_{50}$. A value in row π_1 and column π_2 indicates the percentage of instances on which the genetic algorithm performed at least as good as with mutation rate π_1 than with rate π_2.

π	0.01	0.02	0.03	0.04	0.05	0.075
0.01	–	43.8	41.5	40.2	39.2	44.2
0.02	68.2	–	57.1	55.6	54.6	57.4
0.03	69.1	56.0	–	58.5	54.0	58.5
0.04	69.5	58.2	57.0	–	54.8	61.0
0.05	70.4	58.5	60.8	60.8	–	62.8
0.075	65.0	53.6	56.3	56.9	55.8	–

of Gurobi compared to the heuristic renders it less suitable for the evaluation phase of a genetic algorithm and is the reason for our use of a heuristic.

4.2 Selecting Parameters for the Hybrid Genetic Algorithm

The hybrid genetic algorithm has several parameters that may influence its performance. More specifically, we have the population size s and the mutation probabilities in the close endpoint mutation π_e and in the Steiner points mutation operator π_s. To find good values for these parameters we evaluated the result of the algorithm with different parameter choices. We call a choice of the parameters a *configuration*. Based on preliminary experiments we selected $s \in \{5000, 10\,000, 15\,000, 20\,000, 25\,000\}$ and $\pi_e, \pi_s \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.075\}$. To limit the amount of combinations, we always set $\pi_e = \pi_s$. We ran each combination of parameters on 45 randomly selected instances, 15 each from $\mathcal{B}_{20}$, $\mathcal{B}_{50}$, and $\mathcal{B}_{100}$; to limit the running time we skipped $\mathcal{B}_{10}$ in our formal study since the smaller instances tend to be easier, and they depend less on the choice of the parameters. Moreover, we test both choices of the cable assignment (heuristic with and without greedy improvement).

We first determine a suitable value for the mutation rate. Table 3 shows for each pair of mutation rates π_1 and π_2 in which percentage of the runs on $\mathcal{B}_{20}$ and $\mathcal{B}_{50}$ the algorithm with mutation rate π_1 obtained a result that is at least as good as with mutation rate π_2. According to these results a rate of 0.05 gives at least as good results on a majority of the instances compared with all other choices of mutation rates. The same is true for $\mathcal{B}_{20}$ and $\mathcal{B}_{50}$ individually. For $\mathcal{B}_{100}$ the results change (see Table 6 in Appendix F), and a smaller mutation rate of 0.03 performs better. This makes sense since the mutation rate describes the probability that any individual edge or Steiner point is mutated. As the instances become larger, the probability that some mutation occurs increases as well. If there are many changes at once, some bad changes may overpower the good ones. This may cause the good changes to be discarded even though they would improve the topology.

Next, we compare the results with respect to the population sizes. Table 4 (and Table 7 in Appendix F for $\mathcal{B}_{100}$) present a comparison of the population sizes similar to the tables for the mutation rates. On $\mathcal{B}_{20}$ and $\mathcal{B}_{50}$ the results are not as clear as for the mutation rates,

Table 4: Comparison of different mutation rates on $\mathcal{B}_{20}$ and $\mathcal{B}_{50}$. A value in row s_1 and column s_2 indicates the percentage of instances on which the genetic algorithm performed at least as good as with mutation rate s_1 than with rate s_2.

s	5000	10 000	15 000	20 000	25 000
5000	–	56.6	62.5	63.9	70.2
10 000	56.2	–	66.2	68.1	73.5
15 000	50.8	47.3	–	62.8	70.4
20 000	49.8	44.6	52.3	–	70.8
25 000	42.6	40.1	44.7	42.8	–

Table 5: Percentage of feasible solution with a population size of 5000 after 60s by mutation probabilities π_e and π_c. Table 8 shows the data for all population sizes.

		instance size (#terminals)			
π_e	π_c	10	20	50	100
0.005	0.005	100.00	100.00	100.00	66.67
0.005	0.010	100.00	100.00	100.00	100.00
0.005	0.020	100.00	100.00	100.00	100.00
0.010	0.005	100.00	100.00	100.00	0.00
0.010	0.010	100.00	100.00	100.00	8.33
0.010	0.020	100.00	100.00	100.00	41.67
0.020	0.005	100.00	100.00	100.00	0.00
0.020	0.010	100.00	100.00	100.00	0.00
0.020	0.020	100.00	100.00	100.00	0.00

but a population size of 10 000 seems to be the best choice. In contrast, on $\mathcal{B}_{100}$ the smaller population size of 5000 is a clear winner. This is likely because a smaller population size allows for a larger number of iterations. For larger instances, the number of iterations is already lower than for smaller instances since each iteration is slower. Hence, the larger number of iterations is more important here than having a larger and more heterogeneous population.

4.3 Selecting Parameters for the Combined Genetic Algorithm

In addition to the mutation rates π_e and π_s and the population size s, the combined genetic algorithm is parameterized by the mutation rate π_c of the cable mutation operator. To find good parameters, we selected 12 instances from each benchmark set. Based on preliminary experiments we chose $\pi_e, \pi_s, \pi_c \in \{0.005, 0.01, 0.02\}$ and $s \in \{5000, 10\,000, 20\,000, 30\,000\}$. As in Section 4.2 we only considered configurations with $\pi_e = \pi_s$ to limit the computational effort. We ran each configuration on each instance for 60 s.

Even though the combined genetic algorithm prefers topologies with feasible cable assignments, there are some configurations on $\mathcal{B}_{50}$ and $\mathcal{B}_{100}$ for which feasible solutions were rarely (and sometimes even never) found. Table 5 shows the percentage of instances with a feasible solution for configuations with population size $s = 5000$; the full table is Table 8 in Appendix F. In fact, the two configurations in Table 5 that show 100 % for $\mathcal{B}_{100}$ are the only two

configurations that resulted in feasible solutions for all instances. This already indicates that it is hard to find suitable configurations.

Moreover, taking the solution qualities into account we were not able to determine a configuration that performed consistently well on all benchmark sets. In the end, we settled with $\pi_e = 0.02$, $\pi_c = 0.01$, $s = 30\,000$ for $\mathcal{B}_{10}$, $\pi_e = 0.02$, $\pi_c = 0.01$, $s = 20\,000$ for $\mathcal{B}_{20}$, and $\pi_e = 0.005$, $\pi_c = 0.02$, $s = 5000$ for both $\mathcal{B}_{50}$ and $\mathcal{B}_{100}$.

4.4 Evaluation of the Solution Quality

In this section we analyze the quality of the results produced by the genetic algorithm. We aim to answer two questions. First, how good are the topologies produced by the genetic algorithms? And second, how consistent is the output of the genetic algorithms?

We start by tackling the first question. To this end we randomly selected 50 instances each from $\mathcal{B}_{10}$, $\mathcal{B}_{20}$, $\mathcal{B}_{50}$, and $\mathcal{B}_{100}$. Note that we chose these instances independently from the instances used in Sections 4.2 and 4.3 to determine the best configurations. This was done to prevent giving the genetic algorithms an unfair advantage, which would happen if we assessed the quality of the topologies and determined the best configuration on the same instances. We ran both variants of our hybrid genetic algorithm (`ga_heuristic` with the heuristic cable assignment and `ga_greedy` with the additional greedy improvement of the cable assignment) and the combined genetic algorithm ten times on each instance for 60 s. A (relatively short) running time was selected, since the cable layout algorithm is meant to form a part of a larger algorithmic framework that addresses the full microgrid design problem. In the hybrid genetic algorithm the final cable assignment was found by solving the MILP formulation with Gurobi [13]. To ensure a fair comparison we did the same for the combined genetic algorithm even though it directly optimizes the cable assignment. In addition, we compare the results with the optimal cable assignments (computed by solving the MILP formulation) on the minimum spanning tree and the minimum Steiner tree. Both are solutions to geometric optimization problems that are closely related to the Microgrid Cable Layout problem. Moreover, there are approaches to similar microgrid design problems that use the minimum spanning tree [16, 21] and a procedure introducing Steiner points [8]. We used the software library *geosteiner*[4] to compute minimum Steiner trees.

Figure 2 shows the results of the comparison of the algorithms. As different instances have different costs, we normalize the values by the cost of the optimal cable assignment on the minimum Steiner tree. Each line in the plot corresponds to one algorithm, and each point describes the median cost of that algorithm over the ten runs divided by the costs of the minimum Steiner tree based layout. The instances are sorted by increasing values independently for all lines. We use the median cost as this is a good indicator for a typical result of the algorithm. For example, the line for `ga_greedy` contains the point $(100, 0.193)$; this means that on 100 instances the costs of the median result of ga_greedy are at most 19.3 % of the costs of the minimum Steiner tree based layout.

Since all values are less than 1, we can clearly see that the layout based on the minimum Steiner tree is the worst on all instances. This can be explained by the high number and relative high cost of Steiner points in these layouts. The two variants of the genetic

[4] http://www.geosteiner.com/

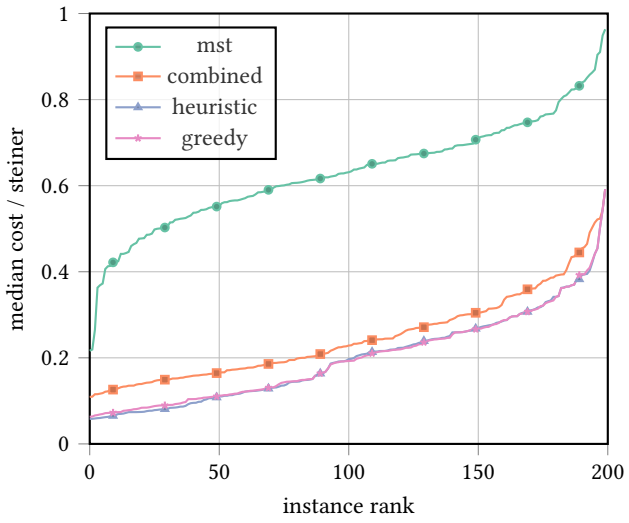


Figure 2: The median costs of the topologies computed by the algorithms (both the configurations of the genetic algorithm and the minimum spanning tree with optimal cable assignment) relative to the costs of the optimal cable assignment on the minimum Steiner tree. Each line represents the results of one algorithm on the different instances. For each instance the median costs of the results has been selected. The instances are sorted by increasing cost ratio.

algorithm have an almost identical performance. This is somewhat surprising as the greedy improvement phase is able to improve the heuristic cable assignment considerably; see Section 4.1. Note however that we always compute the optimal cable assignment via the MILP formulation for the final topology. Otherwise, we would see the same gap between the two variants as in Section 4.1. In fact, on $\mathcal{B}_{100}$, which contains most of the instances with the largest improvement, `ga_heuristic` even produces better results than `ga_greedy` (on average 9.3 % better). This can be explained by the fact that the greedy improvement of the cable assignment increases the time needed for one iteration so that `ga_heuristic` is able to perform more iterations.

A more detailed analysis also reveals that the results by the hybrid genetic algorithms are never worse than the layout based on the minimum spanning tree. There are two instances on which the minimum spanning tree is as good as the solution found by the hybrid genetic algorithm. But on the vast majority of the instances the minimum spanning tree is not optimal.

The combined genetic algorithm performs worse than both hybrid variants. On average the median costs are 19.9 % more expensive than the median costs of `ga_heuristic` and 20.7 % more expensive than `ga_greedy`. This shows that it is hard to optimize the topology and the cable assignment at the same time.

Multiple runs of the genetic algorithm on the same instance may yield different results. If the costs of the results differed a lot between different runs, this would imply that the algorithm rarely finds (near-)optimal solutions. Conversely, if the results are often of similar quality, this may suggest that the solutions are

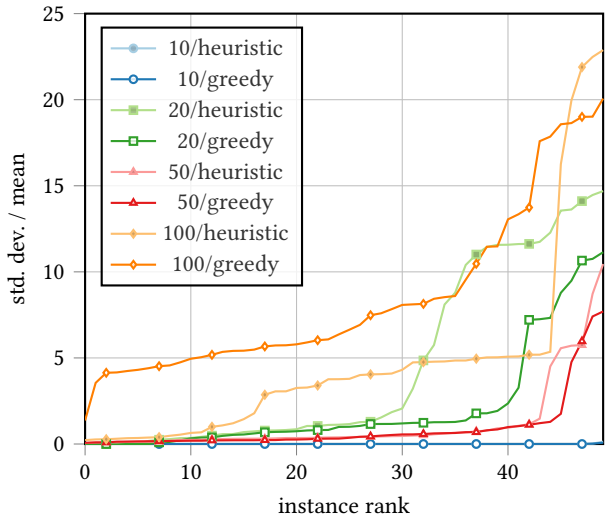
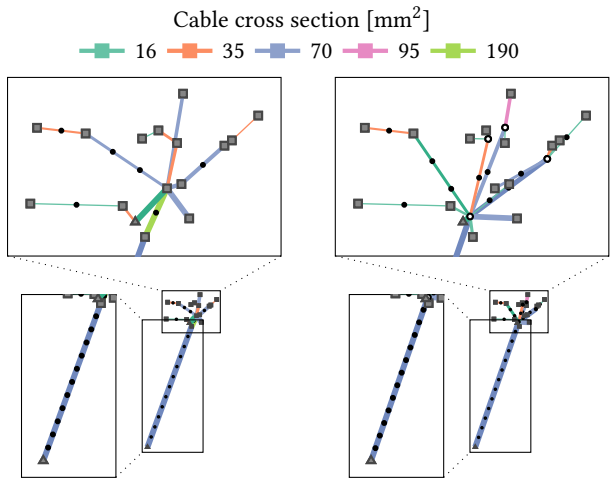


Figure 3: The standard deviation over ten runs of each algorithm per instance normalized by the mean cost of layouts for the instance.



(a) Output of our hybrid genetic algorithm. Cost: 23 671 USD.

(b) Planned network topology. Cost: 29 649 USD.

Figure 4: A comparison of the topology found by our hybrid genetic algorithm and a manually planned reference topology on an instance derived from a real-world microgrid in Idjwi. In both cases the optimal cable assignment for these topologies is shown. The triangles represent generators, the squares represent consumers, and the black dots are poles.

(near-)optimal, in particular, because the genetic algorithm starts with different topologies every time. We compute the standard deviations of the costs of the ten runs for each instance. To be able to compare the standard deviations on different instances, we normalize the values by the mean costs of the layouts for each instance. The results are shown in Fig. 3. Each line represents the normalized standard deviations of one algorithm on the instances of one benchmark set sorted increasingly.

The comparatively large spread on $\mathcal{B}_{100}$ can likely be explained by the fact that the results are not yet optimal after 60 s; in particular, if `ga_greedy` is used. This fits to our observation that `ga_heuristic` performs slightly better on $\mathcal{B}_{100}$. On $\mathcal{B}_{10}$, $\mathcal{B}_{20}$, and $\mathcal{B}_{50}$ the relative standard deviation is small on the majority of instances; in all cases it is below 1.2 % for more than half of the instances, and on average below 4.5 %. As argued above, this may indicate that the results are often close to optimal. We observe that the relative standard deviation for `ga_greedy` (on average 0.0 % on $\mathcal{B}_{10}$, 4.5 % on $\mathcal{B}_{20}$, and 1.2 % on $\mathcal{B}_{50}$) is smaller than for `ga_heuristic` (on average 0.0 % on $\mathcal{B}_{10}$, 2.2 % on $\mathcal{B}_{20}$, and 0.9 % on $\mathcal{B}_{50}$). This means that `ga_greedy` seems to give more consistent results than `ga_heuristic`.

4.5 Case Study: Idjwi

We evaluate the results of the hybrid genetic algorithm on an instance that is derived from a real-world microgrid in the Democratic Republic of the Congo [6]. The microgrid is situated on the island of Idjwi in lake Kivu and powers a small industrial campus. The positions of the points in the instance resemble the real-world locations of the generators and consumers. We deviate from the actual site by including a solar plant, which has not been built yet, and excluding the diesel generator. The grid uses 400 V AC line voltage and is operated at 50 Hz. We assume a load factor of $\cos\varphi = 0.8$ and limit the voltage drops and line losses to within a factor of $\alpha_{\text{drop}} = \alpha_{\text{loss}} = 0.1$.

Figure 4 compares the solution of our hybrid genetic algorithm (with mutation probabilities set to $\pi = 0.05$) with a manually planned reference topoglogy. The cable assignments are the optimal ones according to our model. The two topologies are in general quite similar. They both include a long path from the generator in the bottom left to a central point in the top right. The location of the central point differs slightly in the two topologies. From this point the consumers are connected mostly along paths; in the reference solution some paths branch to two consumers.

The costs of our solution are 23 671 USD, which is significantly cheaper than the 29 649 USD for the reference solution. The cost decrease comes mainly from having fewer vertices with degree larger than 2, which incur a significant cost according to our model. Moreover, the reference topology represents a network that has grown in multiple steps. Even if each step were to be optimal on its own, the resulting topology very likely is not. Nevertheless, this shows that our hybrid genetic algorithm is able to significantly improve the topology.

5 CONCLUSION

In this work we present a hybrid genetic algorithm for the Microgrid Cable Layout problem. Within the algorithm we split this problem into two parts: finding a topology and computing a cable assignment for the topology. The former part is directly optimized by the genetic algorithm; the latter is done by a heuristic. The hybrid genetic algorithm achieves the two goals we formulated. First, it is able to find good solutions quickly, which is evidenced by its performance on a set of benchmark instances. It is able to consistently find solutions within 60 s, which are much better than,

e.g., using a minimum-spanning-tree based layout. Second, adding more constraints or changing the cost function is comparatively easy for a genetic algorithm. In that sense, the algorithmic approach is very flexible, which was the second goal.

The goals for the algorithm are formulated with a larger algorithmic framework covering all aspects of microgrid design in mind. Including the algorithm and studying the interactions with the other components of such a framework is therefore the main future task. This includes finding good locations for the generation units. Another interesting modification would be to compute cable capacities based on realistic generation and load scenarios. Load scenarios also allow proper power flow models to be used to obtain more accurate insights into the projected grid utilization. As in our algorithm cable assignment is part of the evaluation this can be implemented without major changes to other parts of the genetic algorithm. The hybrid genetic algorithm itself may be extended to include, e.g., obstacles, which restrict where the lines or poles may be placed, or more detailed cost functions for the equipment. For example, the choice of a pole and its foundation, and thus their costs, may depend on the forces they are subject to.

For the Microgrid Cable Layout problem it would be interesting to have an efficient algorithm for finding optimal solutions or at least solutions with approximation guarantees. A natural extension of the problem is to allow cycles in the network. One can then try to adapt the algorithm to this more general setting.

ACKNOWLEDGMENTS

This work was funded (in part) 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. The authors would like to thank Johann Kraft and Matthias Luh for discussing the planning of the microgrid in Idjwi with them and for sharing the data on this microgrid. The authors would further like to thank Sascha Gritzbach for helpful discussions. Part of this work is based on the first author's master's thesis (https://i11www.iti.kit.edu/_media/teaching/theses/magoettlicher-21.pdf).

REFERENCES

[1] 2021. HOMER – Hybrid Renewable and Distributed Generation System Design Software. https://www.homerenergy.com/ Accessed: 2021-12-09.

[2] Fahad Saleh Al-Ismail. 2021. DC Microgrid Planning, Operation, and Control: A Comprehensive Review. *IEEE Access* 9 (2021), 36154–36172. https://doi.org/10.1109/ACCESS.2021.3062840

[3] Jorge Barreiros. 2003. An hierarchic genetic algorithm for computing (near) optimal Euclidean Steiner trees. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.319.3516&rep=rep1&type=pdf

[4] Paul Bertheau and Catherina Cader. 2019. Electricity sector planning for the Philippine islands: Considering centralized and decentralized supply options. *Applied Energy* 251 (2019), 113393. https://doi.org/10.1016/j.apenergy.2019.113393

[5] Philipp Blechinger, Catherina Cader, and Paul Bertheau. 2019. Least-Cost Electrification Modeling and Planning–A Case Study for Five Nigerian Federal States. *Proc. IEEE* 107, 9 (2019), 1923–1940. https://doi.org/10.1109/JPROC.2019.2924644

[6] Engineers Without Borders. 2019. *Hydroélectricité Idjwi: Bericht zur ersten Bauphase, September – Dezember 2018.* Technical Report. Engineers Without Borders. https://www.ludwig-boelkow-stiftung.org/wp-content/uploads/2019/08/EWB-Kongo-Bericht_Bauphase1.pdf

[7] Marcus Brazil, Ronald L Graham, Doreen A Thomas, and Martin Zachariasen. 2014. On the history of the Euclidean Steiner tree problem. *Archive for history of exact sciences* 68, 3 (2014), 327–354. https://doi.org/10.1109/TPWRS.2012.2224676

[8] Silvia Corigliano, Tommaso Carnovali, Darlain Edeme, and Marco Merlo. 2020. Holistic geospatial data-based procedure for electric network design and least-cost energy strategy. *Energy for Sustainable Development* 58 (2020), 1–15. https://doi.org/10.1016/j.esd.2020.06.008

[9] Center for Discrete Mathematics and Computer Science. 2014. 11th DIMACS Implementation Challenge. https://dimacs11.zib.de/downloads.html

[10] Carlos Gamarra and Josep M. Guerrero. 2015. Computational optimization techniques applied to microgrids planning: A review. *Renewable and Sustainable Energy Reviews* 48 (2015), 413–424. https://doi.org/10.1016/j.rser.2015.04.025

[11] M. R. Garey, R. L. Graham, and D. S. Johnson. 1977. The Complexity of Computing Steiner Minimal Trees. *SIAM J. Appl. Math.* 32, 4 (1977), 835–859. https://doi.org/10.1137/0132072

[12] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Freeman, San Francisco.

[13] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[14] Mário Jesus, Sérgio Jesus, and Alberto Márquez. 2004. *Steiner Trees Optimization using Genetic Algorithms.* Technical Report 01. Centro de Simulação e Cálculo. https://esght.ualg.pt/sites/ualg.pt/files/ise/tr_2004-01.pdf

[15] Daniel Juhl, David M Warme, Pawel Winter, and Martin Zachariasen. 2018. The GeoSteiner software package for computing Steiner trees in the plane: an updated computational study. *Mathematical Programming Computation* 10, 4 (2018), 487–532.

[16] Onur Kahveci, Thomas J. Overbye, Nathan H. Putnam, and Ahmet Soylemezoglu. 2016. Optimization Framework for Topology Design Challenges in Tactical Smart Microgrid Planning. In *2016 IEEE Power and Energy Conference at Illinois (PECI).* 1–7. https://doi.org/10.1109/PECI.2016.7459262

[17] Charles Kirubi, Arne Jacobson, Daniel M. Kammen, and Andrew Mills. 2009. Community-Based Electric Micro-Grids Can Contribute to Rural Development: Evidence from Kenya. *World Development* 37, 7 (2009), 1208–1221. https://doi.org/10.1016/j.worlddev.2008.11.005

[18] Johann Kraft. 2020. Untersuchung der Kombination von Solar und Wasserkraft zur Off-Grid-Stromversorgung. Fakultät für Elektrotechnik und Informationstechnik.

[19] Oliver Kramer. 2017. *Genetic Algorithms Essentials.* Studies in Computational Intelligence, Vol. 679. Springer. https://link.springer.com/book/10.1007%2F978-3-319-52156-5

[20] Harold W. Kuhn and Robert E. Kuenne. 1962. An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics. *Journal of Regional Science* 4, 2 (1962), 21–33. https://doi.org/10.1111/j.1467-9787.1962.tb00902.x

[21] T. W. Lambert and D. C. Hittle. 2000. Optimization of autonomous village electrification systems by simulated annealing. *Solar Energy* 68, 1 (2000), 121–132. https://doi.org/10.1016/S0038-092X(99)00040-7

[22] Lin Lin and Mitsuo Gen. 2006. Node-Based Genetic Algorithm for Communication Spanning Tree Problem. *IEICE transactions on communications* E89-B, 4 (2006), 1091–1098. https://doi.org/10.1093/ietcom/e89-b.4.1091

[23] Riham Moharam and Ehab Morsy. 2017. Genetic algorithms to balanced tree structures in graphs. *Swarm and Evolutionary Computation* 32 (2017), 132–139. https://doi.org/10.1016/j.swevo.2016.06.005

[24] Steven Nolan, Scott Strachan, Puran Rakhra, and Damien Frame. 2017. Optimized Network Planning of Mini-Grids for the Rural Electrification of Developing Countries. In *2017 IEEE PES PowerAfrica.* 489–494. https://doi.org/10.1109/PowerAfrica.2017.7991274

[25] Chitaranjan Phurailatpam, Bharat Singh Rajpurohit, and Lingfeng Wang. 2018. Planning and optimization of autonomous DC microgrids for rural and urban applications in India. *Renewable and Sustainable Energy Reviews* 82 (2018), 194–204. https://doi.org/10.1016/j.rser.2017.09.022

[26] Jon Sumanik-Leary, Milan Delor, Matt Little, Martin Bellamy, Arthur Williams, and Sam Williamson. 2014. Engineering in Development: Energy. https://thewindyboy.files.wordpress.com/2014/10/energy-book-final-for-press.pdf

[27] United Nations. 2021. The Sustainable Development Goals Report 2021. https://unstats.un.org/sdgs/report/2021/

[28] United Nations General Assembly. 2015. Transforming our world: the 2030 Agenda for Sustainable Development. https://sdgs.un.org/2030agenda A/RES/70/1.

[29] M.R. Vallem and J. Mitra. 2005. Siting and sizing of distributed generation for optimal microgrid architecture. In *Proceedings of the 37th Annual North American Power Symposium, 2005.* 611–616. https://doi.org/10.1109/NAPS.2005.1560597

[30] Mallikarjuna R. Vallem, Joydeep Mitra, and Shashi B. Patra. 2006. Distributed Generation Placement for Optimal Microgrid Architecture. In *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition.* 1191–1195. https://doi.org/10.1109/TDC.2006.1668674

A DEPTH-FIRST SEARCH FOR POWER FLOWS ON EDGES

To assign cables to a given topology $G = (V, E)$ we need to determine the maximum amount of power that may need to be transmitted along each edge. These amounts are given by Eq. (1). A direct application of this equation allows us to compute each flow value in linear time, which results in quadratic time for all edges. However, we can do better than that by performing a single depth-first search; see Algorithm 1.

```
   Data: Tree G = (V, E)
   Result: Maximum power flow values p: E⃗ → ℝ≥0
1  g_total ← Σ_{p∈P} g(p)
2  d_total ← Σ_{p∈P} d(p)
3  r ← arbitrary point in V
4  computeFlow(r, ⊥)
5  function computeFlow(v, u):
6      (g, d) ← (g(v), d(v))
7      for w ∈ N_G(v) \ {u} do
8          (g, d) ← (g, d) + computeFlow(w, v)
9      if u ≠ ⊥ then
10         p(u, v) ← min{d, g_total − g}
11         p(v, u) ← min{d_total − d, g}
12     return (g, d)
```

Algorithm 1: The DFS to determine the maximum flows on the edges.

We begin the DFS at an arbitrary vertex r (Lines 3 and 4). When the function `computeFlow()` is called for a vertex v and its parent in the search tree u, it first recursively calls `computeFlow()` on its children (Line 8). These calls return the total amount of generation and demands within the subtree rooted at the child. After the loop, g and d contain the sum of all generations and demands in the subtree rooted at v (including the generation and demand of v). These values suffice to compute the maximum flows on the edge between v and its parent u (Lines 10 and 11). Note that in the calls to the children (and recursively to their children and so on) the correct flow values of all edges in the subtree rooted at u are determined. Hence, when the algorithm finishes all desired values are computed, and we have the following result.

Lemma A.1. *For a given topology $G = (V, E)$, the maximum flow values can be computed in $O(|V|)$ time.*

B COMPLEXITY OF FINDING AN OPTIMAL CABLE LAYOUT

We reduce the strongly $\mathcal{NP}$-hard problem Euclidean Minimum Steiner Tree [11] to Microgrid Cable Layout. The input of the Euclidean Minimum Steiner Tree problem consists of a set of points $P \subseteq \mathbb{R}^2$ in the Euclidean plane. The goal is to connect these points by line segments (possibly ending at additional points called *Steiner points*) such that the total length of the segments is minimum. In the terms of Section 2 we aim to find a topology such that the sum of all edge lengths is minimum.

Theorem B.1. *Finding a cost-minimal solution to the Microgrid Cable Layout problem is strongly $\mathcal{NP}$-hard.*

Proof. Let $P \subseteq \mathbb{R}^2$ be the points of the Euclidean Minimum Steiner Tree instance. We use them as the points in the Microgrid Cable Layout instance we build. Let $p \in P$ be an arbitrary point, which we use as the single generator, and all other points are consumers. That is, we set $g(p) = 1$, $d(p) = 0$, and for all $p' \in P \setminus \{p\}$ we set $g(p') = 0$ and $d(p') = 1$. We have only one cable type c with $\mathrm{cap}(c) = 1$, $\rho(c) = 1$, and $c_{\mathrm{line}}(c) = 1$. As the total maximum generation is 1, the cable capacity is sufficient for every edge in every topology. To ignore the voltage drop and power loss requirements, we set $\alpha_{\mathrm{drop}} = \alpha_{\mathrm{loss}} = 1$. Hence, the resistivity of the cable does not actually matter. Moreover, we have no costs for the poles ($c_{\mathrm{pole}} = 0$) and do not require any poles in middle of the edges ($d_{\max} = \infty$). In total, for each topology $G = (V, E)$ there is a unique cable assignment a, and we have

$$\mathrm{cost}(G, a) = \sum_{e \in E} \ell(e),$$

which is precisely the objective function of the Euclidean Minimum Steiner Tree problem. Hence, the Euclidean Minimum Steiner Tree instance and the Microgrid Cable Layout instance we constructed are equivalent. This transformation is possible in linear time. Thus, the $\mathcal{NP}$-hardness of Euclidean Minimum Steiner Tree implies that Microgrid Cable Layout is $\mathcal{NP}$-hard as well. □

C COMPLEXITY OF CABLE ASSIGNMENT

We study the complexity of finding a cost-minimal cable assignment that satisfies all constraints mentioned in Section 2. The corresponding decision problem Cable Assignment Problem is defined by: Given a topology, a set of cables, the grid properties (voltage, power factor), and some $s \in \mathbb{R}_{\geq 0}$, is there a feasible cable assignment costing at most s? We prove that this problem is $\mathcal{NP}$-hard by a reduction from the $\mathcal{NP}$-hard problem Subset Sum [12]. An instance of Subset Sum consists of a set $X \subset \mathbb{N}$ and some $k \in \mathbb{N}$. The question then is, whether there is a subset $Y \subseteq X$ such that the sum of all elements of Y is equal to k.

Theorem C.1. *The Cable Assignment Problem is $\mathcal{NP}$-hard even if the topology is a path.*

Proof. Let (X, k) be an instance of Subset Sum. The topology is a path $G = (V, E)$ of $|X|$ edges, where each element $x \in X$ corresponds to one edge e_x of length x. The start vertex s of the path has a generation of $g(s) = 1$ and no demand, and the end vertex t has a demand of $d(t) = 1$ and no generation. All other vertices $v \in V \setminus \{s, t\}$ have $g(v) = d(v) = 0$. There are two cable types c_1 and c_2 with

$$c_{\mathrm{line}}(c_1) = 1, \qquad \rho(c_1) = 2, \qquad \mathrm{cap}(c_1) = 1,$$
$$c_{\mathrm{line}}(c_2) = 2, \qquad \rho(c_2) = 1, \qquad \mathrm{cap}(c_2) = 2.$$

Note that c_2 is equivalent to two cables of type c_1 in parallel. Since the topology is fixed, the costs of equipment other than the cables is constant. Hence, we may assume those costs to be 0 in this proof. Note that for $vw \in E$, we have $p(v, w) = p(w, v) = 1$. Hence, the

capacities of both cable types are sufficient, and we can ignore them in the remainder of this proof. We choose U and φ such that

$$2 \cdot \sum_{x \in X} x - k = \frac{\sqrt{3}}{10} U^2 \cos\varphi. \tag{9}$$

We further set

$$\alpha_{\text{drop}} = \frac{1}{10} \quad \text{and} \quad \alpha_{\text{loss}} = \frac{\sqrt{3}}{10 \cos\varphi}.$$

We claim that there is a cable assignment of cost at most $\sum_{x \in X} x + k$ if and only if there is a solution to the Subset Sum instance (X, k).

Suppose there is $Y \subseteq X$ such that $\sum_{y \in Y} y = k$. We claim that assigning the cables by

$$a(e_x) = \begin{cases} c_1, & x \notin Y, \\ c_2, & x \in Y. \end{cases}$$

results in a feasible cable assignment. The costs of a are

$$\begin{aligned} \text{cost}(a) &= \sum_{e \in E} \ell(e) \cdot c_{\text{line}}(a(e)) \\ &= \sum_{x \in X} x \cdot c_{\text{line}}(a(e_x)) \\ &= \sum_{x \in X \setminus Y} x \cdot 1 + \sum_{y \in Y} y \cdot 2 \\ &= \sum_{x \in X} x + \sum_{y \in Y} y \\ &= \sum_{x \in X} x + k. \end{aligned}$$

Moreover, taking into account that the power along each edge is 1, we obtain the bound for the total power loss

$$\begin{aligned} P_l &= \sum_{e \in E} \ell(e) \rho(a(e)) \cdot \frac{1}{(U \cos\varphi)^2} \\ &= \sum_{x \in X} x \rho(a(e_x)) \cdot \frac{1}{(U \cos\varphi)^2} \\ &= \left(\sum_{x \in X \setminus Y} x \cdot 2 + \sum_{y \in Y} y \cdot 1 \right) \cdot \frac{1}{(U \cos\varphi)^2} \\ &= \left(2 \sum_{x \in X} x - \sum_{y \in Y} y \right) \cdot \frac{1}{(U \cos\varphi)^2} \\ &= \left(2 \sum_{x \in X} x - k \right) \cdot \frac{1}{(U \cos\varphi)^2} \\ &= \frac{\sqrt{3}}{10 \cos\varphi} \\ &= \alpha_{\text{loss}} \cdot 1, \end{aligned}$$

where the second to last equality uses Eq. (9), and the 1 in the final row is the total generation in the grid. Hence, the cable assignment a satisfies the power loss constraint. A similar chain of equations yields

$$U_d(G) = \sum_{e \in E} \ell(e) \rho(a(e)) \cdot \frac{1}{\sqrt{3} U \cos\varphi} = \alpha_{\text{drop}} \cdot U.$$

Note that the whole path P is the only path we need to check for the voltage drop constraint. Hence, the assignment a is feasible and within the desired costs.

Conversely, suppose that there is a feasible assignment a of costs at most $\sum_{x \in X} x + k$. Let $Y = \{x \in X \mid a(e_x) = c_2\}$ be the set of elements whose corresponding edge has been assigned c_2. We have

$$\begin{aligned} \sum_{y \in Y} y &= \sum_{y \in Y} 2y + \sum_{x \in X \setminus Y} x - \sum_{x \in X} x \\ &= \text{cost}(a) - \sum_{x \in X} x \\ &\leq \sum_{x \in X} x + k - \sum_{x \in X} x \\ &= k. \end{aligned}$$

As for the converse above, we obtain

$$2 \sum_{x \in X} x - \sum_{y \in Y} y = P_l \cdot (U \cos\varphi)^2.$$

Applying the power loss bound, we then get

$$\begin{aligned} 2 \sum_{x \in X} x - \sum_{y \in Y} y &\leq \alpha_{\text{loss}} \cdot (U \cos\varphi)^2 \\ &= \frac{\sqrt{3}}{10 \cos\varphi} (U \cos\varphi)^2 \\ &= \frac{\sqrt{3}}{10} U^2 \cos\varphi \\ &= 2 \cdot \sum_{x \in X} x - k, \end{aligned}$$

where the final equation holds by the definition of U and φ. Rearranging this inequality yields

$$\sum_{y \in Y} y \geq k.$$

In total, we have that the sum of Y is equal to k, which shows that the cable assignment instance and the instance of Subset Sum are equivalent.

This transformation can clearly be done in polynomial time. As Subset Sum is $\mathcal{NP}$-hard [12], so is finding a cost-minimal feasible cable assignment even if the topology is a path. □

D COMPUTING THE MAXIMUM PATH LENGTHS

The algorithm to compute for each edge $e \in E$ the maximum lengths of generator-consumer-paths that contain e is given in Algorithm 2. It consists of two tree traversals (Lines 2 and 3) and a final iteration over all edges (Line 4). In the first two traversals, we compute the values of $\hat{\ell}_g$ and $\hat{\ell}_c$. For an edge $vw \in E$, the value $\hat{\ell}_g(v, w)$ is the length of the longest path from a generator to v via the edge vw. Similarly, $\hat{\ell}_c(v, w)$ is the length of the longest path from a consumer to v via vw. In the first traversal (`computeLengthsBelow`) we compute the values $\hat{\ell}_g(v, w)$ and $\hat{\ell}_c(v, w)$ where w is the parent of v in the search tree in a bottom-up fashion. In the second traversal (`computeLengthsAbove`) the remaining values are computed. Based on these values, we compute the desired maximum path lenghts (`computeLengths`). The second traversal and the final iteration may be combined in one traversal.

Data: Tree $G = (V, E)$
Result: Maximum path lenghts $\hat{\ell}_{\text{path}} \colon E \to \mathbb{R}_{\geq 0} \cup \{-\infty\}$

```
1  r ← arbitrary point in V
2  computeLengthsBelow(r, ⊥)
3  computeLengthsAbove(r, ⊥)
4  computeLengths()
5  function computeLengthsBelow(v, u):
6      for w ∈ N_G(v) \ {u} do
7          computeLengthsBelow(w, v)
8      ℓ̂_g(v, u) ← max{ℓ̂_g(w, v) + ℓ(vw) | w ∈ N_G(v) \ {u}}
9      ℓ̂_c(v, u) ← max{ℓ̂_c(w, v) + ℓ(vw) | w ∈ N_G(v) \ {u}}
10     if g(v) > 0 and ℓ̂_g(v, u) = −∞ then
11         ℓ̂_g(v, u) ← 0
12     if d(v) > 0 and ℓ̂_c(v, u) = −∞ then
13         ℓ̂_c(v, u) ← 0
14 function computeLengthsAbove(v, u):
15     for w ∈ N_G(v) \ {u} do
16         ℓ̂_g(v, w) ← max{ℓ̂_g(x, v) + ℓ(vx) | x ∈ N_G(v) \ {w}}
17         ℓ̂_c(v, w) ← max{ℓ̂_c(x, v) + ℓ(vx) | x ∈ N_G(v) \ {w}}
18         if g(v) > 0 and ℓ̂_g(v, w) = −∞ then
19             ℓ̂_g(v, w) ← 0
20         if d(v) > 0 and ℓ̂_c(v, w) = −∞ then
21             ℓ̂_c(v, w) ← 0
22 function computeLengths():
23     for vw ∈ E do
24         ℓ̂_path(vw) ←
             max{ℓ̂_g(v, w) + ℓ̂_c(w, v), ℓ̂_c(v, w) + ℓ̂_g(w, v)} + ℓ(vw)
```

Algorithm 2: The algorithm to compute the lengths of the maximum paths containing an edge.

Note that a naive implementation of the maximum computations in Lines 16 and 17 requires linear time per computation, which results in quadratic running time of the algorithm in total. However, observe that at each point $v \in V$ only the two highest values in $\{\hat{\ell}_g(v, w) \mid w \in N_G(v)\}$ are relevant (and likewise for $\{\hat{\ell}_c(v, w) \mid w \in N_G(v)\}$). Hence, we can compute these values before, which allows us to compute the maxima in constant time each. This results in a linear running time for the whole algorithm.

Lemma D.1. *For all edges $e \in E$ together the maximum lengths of generator-consumer-paths that contain an edge e can be computed in $O(|V|)$ time.*

E CABLE ASSIGNMENT RESULTS

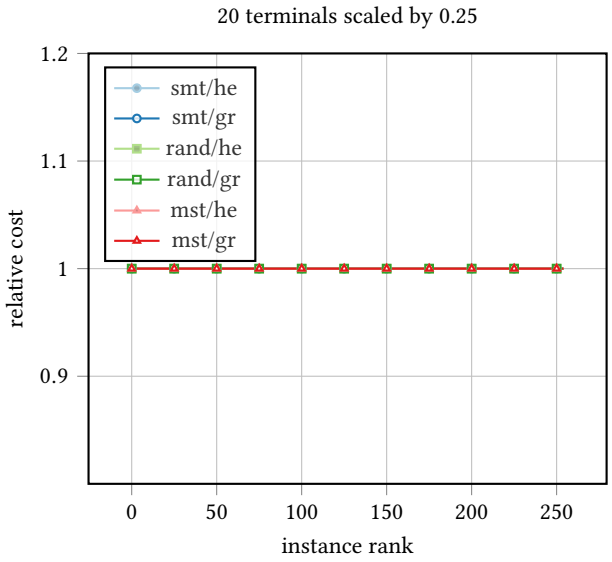


Figure 5: Distribution of the cost of the heuristic cable assignment relative to the optimal assignment.

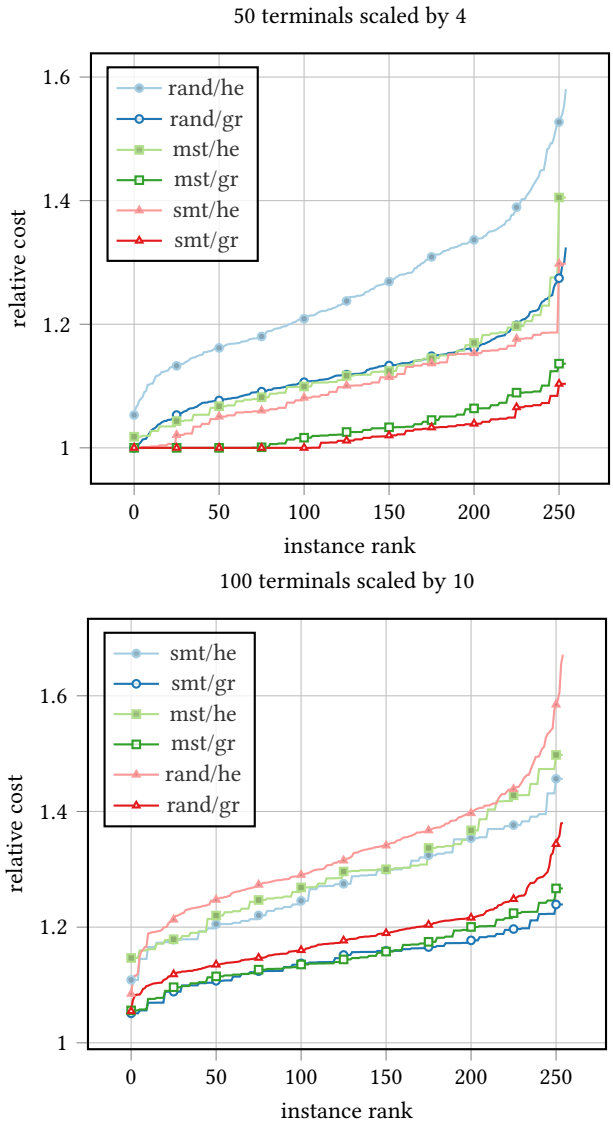


Figure 6: Distribution of the cost of the heuristic cable assignment relative to the optimal assignment. Coordinates in 50 and 100 terminal instances are scaled to match edge length distribution of 20 terminals.

F OMITTED TABLES

Table 6: Comparison of different mutation rates on $\mathcal{B}_{100}$. A value in row π_1 and column π_2 indicates the percentage of instances on which the genetic algorithm performed at least as good as with mutation rate π_1 than with rate π_2.

π	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.075**
0.01	–	43.8	41.5	40.2	39.2	44.2
0.02	68.2	–	57.1	55.6	54.6	57.4
0.03	69.1	56.0	–	58.5	54.0	58.5
0.04	69.5	58.2	57.0	–	54.8	61.0
0.05	70.4	58.5	60.8	60.8	–	62.8
0.075	65.0	53.6	56.3	56.9	55.8	–

Table 7: Comparison of different mutation rates on $\mathcal{B}_{100}$. A value in row s_1 and column s_2 indicates the percentage of instances on which the genetic algorithm performed at least as good as with mutation rate s_1 than with rate s_2.

s	**5000**	**10 000**	**15 000**	**20 000**	**25 000**
5000	–	92.2	99.4	100.0	100.0
10 000	7.8	–	90.0	98.3	100.0
15 000	0.6	10.0	–	91.7	99.4
20 000	0.0	1.7	8.3	–	85.6
25 000	0.0	0.0	0.6	14.4	–

Table 8: Percentage of feasible solutions after 60s by mutation probabilities π_e and π_c.

			instance size (#terminals)			
s	π_e	π_c	10	20	50	100
5,000	0.005	0.005	100.0	100.0	100.0	66.7
5,000	0.005	0.010	100.0	100.0	100.0	100.0
5,000	0.005	0.020	100.0	100.0	100.0	100.0
5,000	0.010	0.005	100.0	100.0	100.0	0.0
5,000	0.010	0.010	100.0	100.0	100.0	8.3
5,000	0.010	0.020	100.0	100.0	100.0	41.7
5,000	0.020	0.005	100.0	100.0	100.0	0.0
5,000	0.020	0.010	100.0	100.0	100.0	0.0
5,000	0.020	0.020	100.0	100.0	100.0	0.0
10,000	0.005	0.005	100.0	100.0	100.0	0.0
10,000	0.005	0.010	100.0	100.0	100.0	0.0
10,000	0.005	0.020	100.0	100.0	100.0	16.7
10,000	0.010	0.005	100.0	100.0	100.0	0.0
10,000	0.010	0.010	100.0	100.0	100.0	0.0
10,000	0.010	0.020	100.0	100.0	100.0	0.0
10,000	0.020	0.005	100.0	100.0	16.7	0.0
10,000	0.020	0.010	100.0	100.0	83.3	0.0
10,000	0.020	0.020	100.0	100.0	100.0	0.0
20,000	0.005	0.005	100.0	100.0	100.0	0.0
20,000	0.005	0.010	100.0	100.0	100.0	0.0
20,000	0.005	0.020	100.0	100.0	100.0	0.0
20,000	0.010	0.005	100.0	100.0	50.0	0.0
20,000	0.010	0.010	100.0	100.0	100.0	0.0
20,000	0.010	0.020	100.0	100.0	100.0	0.0
20,000	0.020	0.005	100.0	100.0	0.0	0.0
20,000	0.020	0.010	100.0	100.0	0.0	0.0
20,000	0.020	0.020	100.0	100.0	75.0	0.0
30,000	0.005	0.005	100.0	100.0	25.0	0.0
30,000	0.005	0.010	100.0	100.0	100.0	0.0
30,000	0.005	0.020	100.0	100.0	100.0	0.0
30,000	0.010	0.005	100.0	100.0	0.0	0.0
30,000	0.010	0.010	100.0	100.0	25.0	0.0
30,000	0.010	0.020	100.0	100.0	91.7	0.0
30,000	0.020	0.005	100.0	100.0	0.0	0.0
30,000	0.020	0.010	100.0	100.0	0.0	0.0
30,000	0.020	0.020	100.0	100.0	0.0	0.0