Parameterized Complexity of Power Network Design: Coordinating Cable Placement Is Hard

Thekla Hamm
Eindhoven University of Technology, The Netherlands

Bart M. P. Jansen
Eindhoven University of Technology, The Netherlands

Faezeh Motiei
Eindhoven University of Technology, The Netherlands

Abstract

We study several generalizations of the Steiner Tree problem that are motivated by the design of power networks. While Steiner Tree asks for a single minimum-cost tree that connects a given set of terminal vertices, a power network typically consists of *multiple trees*. Each tree connects to a subset of the terminals, to avoid electrical overloads. The cost of installing a power network is therefore determined by two factors: the total length of the cables in the network and the cost of digging underground trenches into which the cables are placed. Since the digging costs can be substantial, to minimize the total cost of the network it might be necessary to place multiple cables into the same trench. These characteristics lead to variations of Steiner Tree in which the goal is to compute a minimum-cost set of Steiner trees, all with a common root, that together connect a given terminal set while balancing the power demand of the terminals in each tree. Two important variations arise depending on whether the network is intended for low-voltage or high-voltage power. In the low-voltage setting, there is substantial power loss across the cables which effectively means that the maximum depth of any tree in the solution has to be bounded. No such depth bound applies to the high-voltage setting.

We investigate the parameterized complexity of several power network design problems, using the number of terminals as the parameter. While this parameterization of the standard Steiner Tree problem is fixed-parameter tractable, many of our variants are W[1]-hard. For low-voltage networks (bounded-depth trees), we present an XP-algorithm for planar inputs, which exploits a nontrivial bound on the treewidth of solution subgraphs. We provide an intricate reduction from Grid Tiling to establish that the resulting algorithm is tight under the Exponential Time Hypothesis. The XP-algorithm extends to the high-voltage setting and to general graphs, albeit at a cost in the running time. For high-voltage networks, we prove that the problem remains W[1]-hard on planar graphs. Finally, we explore a variation of the cost model for sharing digging costs in which both problems become fixed-parameter tractable.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability

Keywords and phrases Steiner Tree, Network Design, Parameterized Complexity, ETH-tight Algorithm

Digital Object Identifier 10.4230/LIPIcs.WG.2026.23

Related Version *Full Version*: https://arxiv.org/abs/2606.25859

1 Introduction

Steiner Tree is a classic NP-complete problem [19, 24] in network design with numerous applications [5, 12, 26, 32]. Given a graph G and a subset of its vertices called *terminals*, the goal is to compute a minimum-weight subtree of G that contains all terminals. The study of Steiner Tree has led to a wide range of algorithmic techniques in parameterized complexity [4, 10, 13, 16, 18, 20, 22, 27, 30, 31, 33]. In this work, we investigate the

parameterized complexity of several generalizations of Steiner Tree concerning the design of power networks. They arose from a cooperation with an industrial partner that develops underground infrastructure in the form of low-voltage electrical power networks.

Background and Motivation. A low-voltage power network distributes electricity from a transformer station to an urban neighborhood. A transformer station has a fixed number (typically less than a dozen) of power outputs that each feed one aluminum or copper power cable. Such a cable supplies power to multiple households and can fork off into a tree layout to do so. The maximum amount of power that can be drawn from any single output on a transformer station is limited, which is why the households have to be balanced over the power cables based on their demand. Due to power losses during transport, the *depth* (maximum length of a root-to-leaf path) of the tree-structure formed by a power cable determines its maximum load. Hence, the design of a low-voltage power network involves deciding which households are connected to a common cable, as well as finding a suitable low-depth tree structure for each cable. A high-voltage network has a similar structure, but does not require a depth bound on the tree structures since losses on high-voltage connections are negligible.

The *cost* of installing a network depends on two factors. Naturally, there is a cost for the material of the cables that scales linearly with their total length. A less obvious factor, which can be even more important in practice, is the cost of *installing* the network, i.e., digging trenches in the ground in which the cables are placed. The cost of digging a unit-length trench depends heavily on the terrain: it is cheap when the surface is covered by grass or small pavement tiles that can temporarily be set aside, but it is expensive when the surface is covered by an asphalt road that has to be sliced open and repaired. When multiple cables are placed in the *same* trench, the costs for digging the trench can be shared among all the cables placed inside it. Hence, it may be advantageous to route cables over slightly longer trajectories if this avoids digging through asphalt or allows cables to share a trench.

As the energy transition necessitates the development of better power networks that support a society that increasingly relies on electricity to operate cars, heat pumps, and cooking appliances [3, 14], designing power networks has important applications.

Problem Statement. We model the task of designing a power network as a graph problem. Each edge e of the input graph G represents a trench that can be dug to place cables underground. Each edge has a length $\ell(e) \in \mathbb{N}$ which determines the material costs for placing a cable in the corresponding trench, and a separate cost $\mathrm{dig}(e) \in \mathbb{N}$ for digging the trench. The transformer station is represented by a vertex r in G, which serves as the root of the tree-structured cables. The power consumers to be connected are represented by *terminal* vertices T, which are dead ends in the network and have degree one in G. Since power consumption differs per consumer, each terminal vertex $u \in T$ has an associated power *demand* $d(u) \in \mathbb{N}$. Finally, the input specifies the number of power outputs t on the transformer station, the depth constraint λ of each tree-structured cable (in the low-voltage setting), and the maximum power demand α that can be supplied by a single cable.

To turn the corresponding algorithmic task into a decision problem, there is a threshold value β for the total cost. An input of the problem then asks whether there is a feasible network of cost at most β. In the cost function, we pay the digging cost $\mathrm{dig}(e)$ once for every edge e in the *union* of the solution trees, and for each occurrence of an edge e in a solution tree, we additionally pay $\ell(e)$ in cable costs. This leads to the following formalization, in which we employ the number of trees t and the number of terminals $|T|$ as the parameters since they are typically much smaller than the total size of the network of potential trenches.

LOW-VOLTAGE NETWORK DESIGN (LVND) ***Parameter:*** $t + |T|$

Instance: An undirected graph G with *edge lengths* $\ell\colon E(G) \to \mathbb{N}_0$ and *digging costs* $\mathrm{dig}\colon E(G) \to \mathbb{N}_0$, a *root vertex* $r \in V(G)$, a set of degree-1 *terminal* vertices $T \subseteq V(G)$ with *demands* $d\colon T \to \mathbb{N}$, and integers α, β, λ, and t, encoded in unary.

Question: Does G contain t tree subgraphs $H_1, \ldots, H_t$, each rooted at r, such that:

- $T \subseteq \bigcup_{i=1}^{t} V(H_i)$ [*covering constraint*: each terminal is covered],
- each root-to-leaf path P in each tree H_i satisfies $\sum_{e \in E(P)} \ell(e) \leq \lambda$ [*depth* constraint],
- $\sum_{u \in V(H_i) \cap T} d(u) \leq \alpha$ for each $1 \leq i \leq t$, [*demand* constraint: the total demand of terminals in each tree H_i is at most α], and
- $\mathrm{cost}(\{H_1, \ldots, H_t\}) := \left(\sum_{i=1}^{t} \ell(H_i)\right) + \mathrm{dig}(\bigcup_{i=1}^{t} H_i) \leq \beta$? [*cost constraint*]

The high-voltage variation of the problem arises from the above by setting $\lambda = \infty$, so that the depth constraint plays no role. We refer to it as HIGH-VOLTAGE NETWORK DESIGN (HVND). The assumption that integers are encoded in unary means we focus our analysis on how the size of the graph and the number of terminals affect the complexity of solving the problem. With input integers encoded in binary, LVND remains NP-complete for $t = 1$ and $|T| = 1$ since it generalizes SHORTEST WEIGHT-CONSTRAINED PATH [19, ND 30].

Our Contribution. We initiate the parameterized-complexity analysis of power network design, focusing on the parameterization by $t + |T|$. While STEINER TREE [13] is fixed-parameter tractable (FPT) for the parameter $|T|$, which easily extends to generalizations such as (PRIZE-COLLECTING) STEINER FOREST, our problems turns out to be significantly harder. We prove that LVND and HVND are W[1]-hard parameterized by $t + |T|$, even when restricted to *planar* graphs. Planar inputs are particularly relevant since the graph models potential locations for digging trenches on the surface of the earth. As in previous W[1]-hardness proofs for (planar) network design problems [2, 7, 8], our reductions start from GRID TILING[1] ([28], cf. [9, S14.4.1]) and produce a grid-like graph in which the solution has to route a *horizontal part* and a *vertical part*, which have to interact in a prescribed way to meet the cost requirement.

For LVND (Theorem 4.1), we only utilize two trees ($t = 2$). By a suitable setting of the demand values for the terminals, we can ensure that the first solution tree H_1 must contain a prescribed subset T_1 of terminals, which are attached to the top row of the grid, whereas the remaining tree H_2 contains the remainder, connected to the rightmost column. By a careful choice of depth constraint, we then ensure that the main part of tree H_1 consists of horizontal paths through the grid, and the tree H_2 consists of paths that are vertical apart from using some horizontal edges. To obtain a solution whose cost meets the desired threshold, each horizontal edge used by H_2 must lie on a horizontal path in H_1 so that some digging cost is shared among H_1 and H_2. By setting the cost of such edges based on feasible combinations of values in the input of GRID TILING, we can ensure the correctness of the reduction. It shows that coordinating the cable placement for the two trees H_1, H_2 is hard.

[1] GRID TILING receives as input a $k \times k$ matrix, each cell of which contains a set of pairs of integers. The task is to decide whether a pair can be selected from each cell such that the first entries of selected pairs are equal in cells of the same row, while the second entries are equal for cells of the same column.

Our hardness reduction for HVND (Theorem 4.6) is significantly more involved. Since solutions to the high-voltage problem are not bounded in depth, we have to insert additional terminals inside the grid to enforce the solution to consist of separate horizontal and vertical parts. Embedding these terminals in a planar way while still allowing a solution sufficient degrees of freedom to model Grid Tiling is difficult. We overcome this obstacle by encoding the vertical and horizontal choices using *two* trees each, leading to $t = 4$.

By known lower bounds [9, Theorem 14.28] on the running time for solving Grid Tiling, our first reduction implies that under the Exponential Time Hypothesis (ETH) LVND on planar input graphs cannot be solved in time $f(|T|, t) \cdot (|V(G)| \cdot \lambda^{f'(t)})^{o(|T|)}$ for any computable functions f and f'. We give an algorithm for planar inputs whose running time matches this bound, thereby showing that the *square-root phenomenon* [29] does not manifest itself for this problem. The algorithm works by enumerating a bounded-size set of choices for a certain *abstract structure* $\mathcal{A}(\mathcal{S})$ of the desired solution $\mathcal{S} = \{H_1, \ldots, H_t\}$ and encoding the search for the best solution matching that structure as a Valued Binary Constraint Satisfaction Problem (CSP). We give an exchange argument to prove that if there is a solution, there is one whose abstract structure has $\mathcal{O}(|T|^2)$ vertices. When the input graph G is planar, then the abstract structure (which is a minor of G) must be as well, and therefore its treewidth is $\mathcal{O}(|T|)$. The treewidth of the abstract structure governs the treewidth of the primal graph of the CSP, which yields the claimed running time using known CSP algorithms. A similar approach also yields an XP-algorithm if the input graph G is not planar, and for the high-voltage problem, albeit at a cost in the running time.

Our lower bounds show that even when the parameterization of number of terminals and solution trees is bounded, power network design is intractable due to the difficulties of coordinating cable placement. These difficulties effectively arise when two solution trees leave the root node in different directions, but later converge to place some of their cables in a common trench. As our last contribution, we present an alternative model for the cost computation in which digging costs can only be shared along *prefixes* of the solution trees. We develop an FPT-algorithm for the resulting Prefix-Sharing Low-voltage Network Design problem parameterized by $|T|+t$ by adapting Dreyfus-Wagner [13], thereby shedding further light on which aspect of the cost sharing model leads to intractability.

Related Work. In Strongly Connected Steiner Subgraph, the goal is to find a minimum-weight subgraph of a directed graph that is *strongly connected* and contains all terminals. This problem, and several variations, are known to be W[1]-hard parameterized by the number of terminals [15, 34]. However, the direction of the edges plays a crucial role in this hardness. Common variations of Steiner Tree on *undirected* graphs tend to be FPT parameterized by $|T|$. We are not aware of earlier work on the parameterized complexity of network design problems in which costs can be shared. But, in the operations research community, there has been research [1] on the Minimum-Cost Shared Arborescence problem, which also incorporates this cost-sharing aspect. It lacks other features of Low-voltage Network Design though: most importantly, the depth constraint. When it comes to network design problems involving costs and depth-constraints, there has been some work on the Shallow-Light Steiner Tree problem, which aims to compute a low-diameter Steiner tree of minimum cost [6, 21, 25]. This problem formulation, therefore, incorporates the depth constraint, but not the cost-sharing feature.

Organization. After presenting basic preliminaries in Section 2, we provide our algorithmic results in Section 3. The complementary hardness proofs are given in Section 4. We conclude in Section 5.

2 Preliminaries

For $i \in \mathbb{N}$, we denote by $[i]$ the set $\{1, \ldots, i\}$. For sets A, B we denote by 2^A the powerset of A, and by $A \times B$ the set $\{(a, b) \mid a \in A \wedge b \in B\}$. Given a total function $f\colon A \to B$ and a subset $A' \subseteq A$, we define $f(A') = \{f(a) \mid a \in A'\}$. We assume familiarity with graph theory [11] and parameterized complexity theory [9] and use standard terminology and notation from both. Unless stated otherwise, we consider undirected simple graphs and denote an edge between vertices u and v by $uv = vu$. The *interior* of a path P is the subgraph induced by its non-endpoint vertices. For a rooted tree (T, r) with edge weights $w\colon E(T) \to \mathbb{N}$, its *$w$-depth* is $\max_{t \in V(T)} \sum_{e \in E(P_{r,t})} w(e)$, where $P_{r,t}$ is the unique path from r to t in T. When w is clear from context, we simply refer to this quantity as depth. For brevity, for $w\colon E(G) \to \mathbb{N}$ and $H \subseteq G$, we denote $w(H) = \sum_{e \in E(H)} w(e)$. For a set of subgraphs $S = \{H_1, \ldots, H_t\}$ and a subgraph G', the cost of S in G' is defined as $\text{cost}_{G'}(H_1, \ldots, H_t) = \text{cost}(H_1 \cap G', \ldots, H_t \cap G')$. We denote the treewidth of a graph G by $\text{tw}(G)$.

For our algorithms, we use two results from the literature as subroutines; one for certain path computations, and one for solving a certain *constraint satisfaction problem (CSP)*.

▶ **Theorem 2.1** ([23]). *Given a directed graph G, two functions $f, g\colon E(G) \to \mathbb{N}$, two vertices $s, t \in V(G)$, and a threshold $\tau \in \mathbb{N}$, there is an algorithm that computes a path P from s to t that satisfies $\sum_{e \in E(P)} f(e) \leq \tau$ for which $\sum_{e \in E(P)} g(e)$ is minimized, if one exists, in time $\mathcal{O}(\tau \cdot |V(G)| \cdot |E(G)|)$. If no such path exists, the algorithm reports failure.*

A *valued binary CSP* is a tuple $(\mathcal{V}, \mathcal{D}, \mathcal{C})$ where $\mathcal{V}$ and $\mathcal{D}$ are two finite sets called the set of *variables* and the *domain* respectively. The third component $\mathcal{C}$ is a finite set of *value constraints* which are tuples (x, y, c) with $x, y \in \mathcal{V}$ and $c\colon \mathcal{D} \times \mathcal{D} \to \mathbb{N}$. Given an *assignment* $\phi\colon \mathcal{V} \to \mathcal{D}$, its *value* is $\sum_{(x,y,c) \in \mathcal{C}} c(\phi(x), \phi(y))$. The *primal graph* $G_P(\mathcal{V}, \mathcal{D}, \mathcal{C})$ of a valued binary CSP $(\mathcal{V}, \mathcal{D}, \mathcal{C})$ is the graph $(\mathcal{V}, \{xy \mid (x, y, c) \in \mathcal{C} \text{ for some } c\colon \mathcal{D} \times \mathcal{D} \to \mathbb{N}\})$.

▶ **Theorem 2.2** (folklore, e.g. [17]). *Given a valued binary CSP $(\mathcal{V}, \mathcal{D}, \mathcal{C})$, an assignment of minimum value can be computed in time $|\mathcal{V}| \cdot |\mathcal{D}|^{\mathcal{O}(\text{tw}(G_P(\mathcal{V}, \mathcal{D}, \mathcal{C}))}$.*

The following lemma can be used to show that for a given partition $\mathcal{P}$ of terminals, a demand function and a value α can be defined such that $\mathcal{P}$ is the unique partition into sets of total demand at most α. This will be used later, in Section 4.2, to prove that both Low-voltage Network Design and High-voltage Network Design are W[1]-hard.

▶ **Lemma 2.3** (⋆). *There exists a polynomial-time algorithm that, given a set of items U and a partition $\mathcal{P}$ of U into t sets $\mathcal{P}_1, \ldots, \mathcal{P}_t$, computes a function $f\colon U \to \left[|U|^{t+2}\right]$ such that $\mathcal{P}$ is the unique partition of U into t sets that satisfies $\sum_{x \in \mathcal{P}_i} f(x) \leq \frac{1}{t} \sum_{x \in U} f(x)$ for each $i \in [t]$.*

Proof sketch. To provide such a function, we assign a small value $\epsilon_i = |U|^{i-1}$ to all but one of the elements in P_i, with the last item having value $\Omega_i = |U|^{t+2} - (|\mathcal{P}_i| - 1) \cdot \epsilon_i$. This ensures no part can contain two such items, while the distribution of the rest is unique. ◀

3 Algorithms

3.1 Algorithms for LVND and HVND

In this section, we present our algorithmic results for LVND and HVND. These are based on first analyzing the structure of solutions and then exploiting this structure to be able to compute their cost using Theorem 2.1 and Theorem 2.2.

3.1.1 Solution structure

We will now analyze the structure of a solution viewed as a set of minimum-cost bounded-length paths between certain endpoints. In particular, we want to bound the number of such endpoints that we need to consider. For this, we introduce the *abstract structure* of a solution whose vertices correspond to such endpoints and whose edges correspond to minimum-cost bounded-length paths between them.

▶ **Definition 3.1.** *For a solution $\mathcal{S} = \{H_1, \ldots, H_t\}$ of LVND or HVND, the* abstract structure $\mathcal{A}(\mathcal{S})$ *is a graph H with a color from $\{c_S \mid \emptyset \neq S \subseteq 2^{[t]}\}$ associated to each of its edges, constructed as follows.*

- *Initially, let $H = \bigcup_{i=1}^{t} H_i$ and color each $e \in E(H)$ by c_S, where $S = \{i \in [t] \mid e \in E(H_i)\}$.*
- *While a vertex $v \neq r$ has degree 2 in H and its incident edges have the same color c, we remove v and connect the neighbors of v with an edge of color c.*

The subgraph of $\mathcal{A}(\mathcal{S})$ corresponding to H_i, denoted by $\mathcal{A}_i(\mathcal{S})$, is the subgraph of $\mathcal{A}(\mathcal{S})$ consisting of all edges with a color c_S for which $i \in S$.

Considering the color of the edges of $\mathcal{A}(\{H_1, \ldots, H_t\})$, we can find $\mathcal{A}_i(\mathcal{S})$ in polynomial time for each $i \in [t]$. Clearly, $\mathcal{A}_i(\mathcal{S})$ is obtained from H_i by dissolving some degree-2 vertices.

The following lemma guarantees the existence of a solution for which the number of vertices in its abstract structure is bounded by a function of $|T|$. This bound enables us to enumerate all possible such structures and, accordingly, to search for a solution.

▶ **Lemma 3.2 ($\star$).** *For every yes-instance $(G, \ell, \mathrm{dig}, r, T, d, \alpha, \beta, \lambda, t)$ to LVND and HVND, there exists a solution $\mathcal{S} = \{H_1, \ldots, H_t\}$ such that $|V(\mathcal{A}(\mathcal{S}))| \leq 1 + 2|T| + 8|T|^2$.*

Proof sketch. We sketch the proof of a bound of $\mathcal{O}(|T|^2)$ vertices; the exact argument can be found in the full version. We define an ordering on solutions to the instance, and then prove that a maximal element in this ordering has an abstract structure whose number of vertices is suitably small. For an arbitrary solution $\mathcal{S} = \{H_1, \ldots, H_t\}$, for each edge $e \in E(G)$, we set $x_e(\mathcal{S})$ to be the number of trees in $\mathcal{S}$ that contain e. Then, we define *the coloring sequence* of $\mathcal{S}$ as a tuple $\mathcal{T}(\mathcal{S}) := (y_1, \ldots, y_{|E(G)|})$ where $y_1, \ldots, y_{|E(G)|}$ is the decreasingly sorted sequence of values $x_e(\mathcal{S})$ for $e \in E(G)$. Let $\mathcal{S} = \{H_1, \ldots, H_t\}$ be a solution where the value of $\mathrm{cost}(\mathcal{S})$ is minimized and which, among all solutions that minimize $\mathrm{cost}(\mathcal{S})$, has a maximum tuple $\mathcal{T}(\mathcal{S})$ with respect to the lexicographic ordering.

To bound the number of vertices in the abstract structure $\mathcal{A}(\mathcal{S})$, we exploit the conditions under which Definition 3.1 removes a degree-2 vertex. These conditions can be expressed using the following concept. For a tree H_i, we define its *special vertices* to be the root r of the instance together with the vertices of H_i whose degree in H_i is unequal to 2. We define a *plain path* of H_i as a path P in H_i whose endpoints are special but whose internal vertices are not. Since degree-2 non-root vertices of $\bigcup_i H_i$ are dissolved when both incident edges have the same color, while the edge colors encode which trees contain the edge, it follows that the vertices of $\bigcup_i H_i$ that remain in $\mathcal{A}(\mathcal{S})$ are (1) the root vertex r, (2) each vertex that has degree unequal to 2 in at least one tree H_i, and (3) vertices v that lie on plain paths P_i, P_j of two distinct trees H_i, H_j, such that v has degree more than 2 in the union $H_i \cup H_j$, i.e., places where plain paths of distinct trees converge or diverge (see Figure 1).

To bound the number of vertices in the abstract structure, we bound the number of vertices of each of the three types. For (1), there is only one root vertex r in the instance. For (2) the argument uses the assumption of cost-minimality, which implies that each leaf of a tree H_i is either the root or a terminal. Since a terminal vertex in T has degree-1 in G by

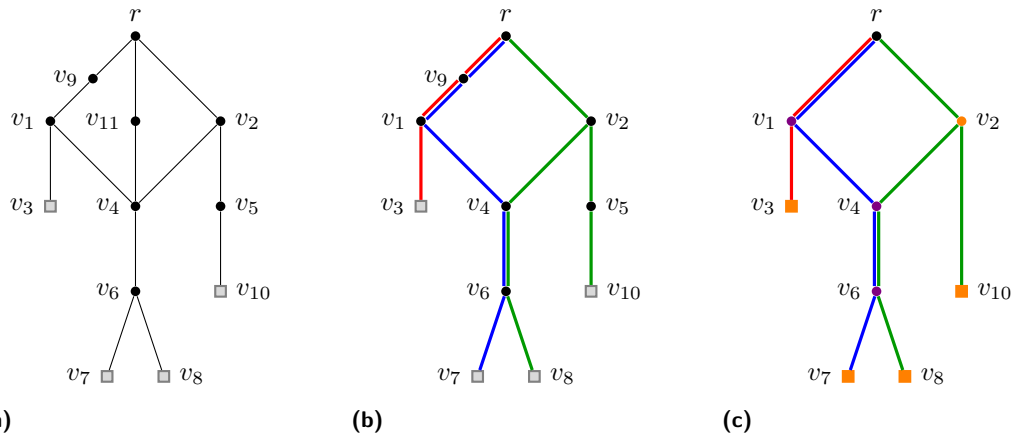


Figure 1 (a) An instance of LVND/HVND; each edge has length 1 and digging cost 100. Squares represent terminals. (b) a solution $\mathcal{S}$. (c) $\mathcal{A}(\mathcal{S})$, with vertices of the same type in the same color.

the definition of LVND and only has to be covered once, cost-minimality also implies that no terminal appears in more than one tree. As the number of vertices of degree more than 2 in a tree is less than its number of leaves, a simple counting argument then yields a bound of $\mathcal{O}(|T|)$ on the number of vertices of type (2). It remains to analyze vertices of type (3), which dominate the asymptotics. The following claim will be instrumental to do so.

▷ Claim 3.3. For plain paths P_i, P_j of two distinct trees H_i, H_j, the graph $P_i \cup P_j$ has at most two vertices of degree more than 2.

Proof. Since both P_i and P_j are paths, a vertex of $P_i \cup P_j$ of degree more than 2 is a place where the two paths either converge or diverge. Hence if there are three or more such vertices, the graph $P_i \cup P_j$ has a cycle. A minimal such cycle can be decomposed into two parts where one part is a subpath of P_i and the other is a subpath of P_j. We call these sub-paths P_i' and P_j', respectively. Using P_i' and P_j', we now derive a contradiction by building a minimum-cost solution S' whose coloring sequence is larger than that of $\mathcal{S}$.

Since all internal vertices of P_i' and P_j' are plain, they have degree 2 and cannot be a terminal or the root. Hence we can replace P_i' in H_i by P_j' to obtain a tree H_i' containing the root that covers the same terminals as H_i. Similarly, we can replace P_j' in H_j by P_i' to obtain a tree H_j'. If $\ell(P_i') \neq \ell(P_j')$, then replacing the longer by the shorter segment yields a solution whose cost is strictly smaller: the digging costs do not increase (as all edges involved in the replacement were dug on account of the other tree anyway), the total depth does not increase, and the total length of the edges in the replaced tree does not increase either. Since we started from a cost-minimal solution, we must therefore have $\ell(P_i') = \ell(P_j')$. We now derive a contradiction by defining a minimum-cost solution that is lexicographically larger.

Let x_i' be the maximum value of $x_e(\mathcal{S})$ over all $e \in E(P_i')$ and x_j' be the maximum value of $x_e(\mathcal{S})$ over all $e \in E(P_j')$. Without loss of generality, assume that $x_i' \geq x_j'$. Let $\mathcal{S}'$ be the solution obtained from $\mathcal{S}$ by replacing H_j with H_j'. We argue the coloring sequence of $\mathcal{S}'$ is larger than that of $\mathcal{S}$ in the lexicographical ordering. The edge of $P_i' \cup P_j'$ that was used most in solution $\mathcal{S}$ (which appears on the most relevant position in the sorted sequence) appears at least one additional time in solution $\mathcal{S}'$ on account of replacing H_j (which does not contain that edge) by H_j' (which does). This contradiction proves the claim. ◁

Claim 3.3 allows us to bound the number of vertices of $\mathcal{A}(\mathcal{S})$ of type (3) by bounding the total number of plain paths in the solution. Note that the plain paths of H_i are in 1-to-1 correspondence with the edges of H_i' that result from dissolving all non-special vertices of H_i. By bounding the number of special vertices by that of terminals, the total number of edges in $H_1', \ldots, H_t'$ (and therefore the total number of plain paths) can be bounded by $\mathcal{O}(|T|)$. By Claim 3.3, each pair of plain paths contributes a constant number of vertices of type (3) to $\mathcal{A}(\mathcal{S})$. Hence the total number of vertices in $\mathcal{A}(\mathcal{S})$ is $1 + \mathcal{O}(|T|) + \mathcal{O}(|T|^2) \leq \mathcal{O}(|T|^2)$. ◀

3.1.2 XP-algorithms for Power Network Design

Using bounds on the abstract structure we present XP-algorithms for power network design.

▶ **Theorem 3.4 (⋆).** *The* Low-voltage Network Design *problem can be solved in time* $f(t, |T|) \cdot (|V(G)| \cdot \lambda^t)^{\mathcal{O}(|T|^2)}$ *for any arbitrary input instance* $(G, \ell, \mathrm{dig}, r, T, d, \alpha, \beta, \lambda, t)$.

Proof sketch. The algorithm enumerates all candidates for the (edge-colored) abstract structure $\mathcal{A}(\mathcal{S})$ of the solution $\mathcal{S}$. Lemma 3.2 ensures the number of candidates is bounded by $f(t, |T|)$. For a fixed guess of the abstract structure $\mathcal{A}(\mathcal{S})$, the task of determining whether there is a solution whose trees $H_1, \ldots, H_t$ realize the abstract structure is reduced to a Valued Binary Constraint Satisfaction Problem.

The CSP formulation has one variable per vertex of $\mathcal{A}(\mathcal{S})$. The domain consists of $V(G) \times (\{0, \ldots, \lambda\}^t)$, so that assigning a domain value to a vertex $v \in V(\mathcal{A}(\mathcal{S}))$ corresponds to realizing v at a particular vertex of G and choosing a t-tuple of integers that encode the depth of vertex v in each of the trees among $H_1, \ldots, H_t$ that contain v. For each edge uv in the guess $\mathcal{A}(\mathcal{S})$ of the abstract structure, there is a constraint acting on the variables for u and v. For this constraint, the cost of assigning values $\tau(u), \tau(v)$ to u and v is computed via Theorem 2.1 as the minimum cost for a $\tau(u)\tau(v)$-path P in G whose sum of ℓ-values matches the depth values encoded in the domain. Using the coloring encoded in the abstract structure, which encodes which of the trees $H_1, \ldots, H_t$ use edge e and therefore path P, we can set up the cost function for the CSP in such a way that solutions to the CSP correspond to solutions for LVND matching the prescribed guess $\mathcal{A}(\mathcal{S})$.

To optimize the CSP we use Theorem 2.2, which gives a running-time guarantee based on the treewidth of the primal graph of the CSP. In our formulation, the primal graph coincides with $\mathcal{A}(\mathcal{S})$ and therefore has $\mathcal{O}(|T|^2)$ vertices; this quantity also bounds the treewidth. Plugging this into Theorem 2.2 with domain size $|V(G)| \cdot (\lambda + 1)^2$ proves the theorem. ◀

The same algorithm can be used for the HVND problem by changing the cost of an edge constraint. When applying the algorithm to a *planar* input of LVND, which guarantees the abstract structure is also planar, the treewidth bound of the primal graph improves to $\mathcal{O}(\sqrt{|V(\mathcal{A}(\mathcal{S})|}) \leq \mathcal{O}(|T|)$. This improves the $\mathcal{O}(|T|^2)$ factor in the exponent to $\mathcal{O}(|T|)$.

3.2 FPT-Algorithms for Prefix-Sharing Low-voltage Network Design

In an attempt to understand how sharing digging costs causes intractability, we define a variation of the power network design problem whose cost function is easier to optimize. While the cost of a solution still arises as a digging term plus the sum of the cable lengths, we change the definition of the digging term. In the definition of Section 1, whenever multiple trees contain the same edge the digging costs of the corresponding trench only have to be paid once. The variation we introduce effectively means that when multiple trees contain the same edge, the digging costs can only be shared when the path from the root to this

edge is the same in both trees. This is formalized through a technical definition of the different *prefixes* leading to an edge. The digging costs are then paid once for each different prefix leading to that trench. We show that under this cost model, the problem is FPT parameterized by $|T| + t$ via dynamic programming.

We now formally define and analyze the resulting Prefix-Sharing Low-voltage Network Design problem. For the definition we need the concept of a *walk* in a graph, which is an alternating sequence of vertices and edges $(v_0, e_1, \ldots, v_k)$ such that edge e_i is incident on v_{i-1} and v_i for all $i \in [k]$. A walk is *simple* (sometimes called *trail* in the literature) when no edge occurs more than once. Note that a path is a simple walk with no repeated vertex. If the direction of the walk is irrelevant, we can also consider a simple walk as a set of edges.

▶ **Definition 3.5.** *Let G be a graph and let $W = (v_0, e_1, \ldots, v_k)$ be a simple walk in G starting at r. For each edge $e_i \in E(W)$ we define the prefix of e_i in W, denoted by $pfx(W, r, e_i)$, as the subwalk of W from r to v_i; in other words, $pfx(W, r, e_i) = (v_0, e_1, \ldots, e_i, v_i)$.*

This definition can be used to define the *prefix set* of an edge in a given set of simple walks.

▶ **Definition 3.6.** *Let G be a graph and let $\mathcal{F} = \{W_1, \ldots, W_x\}$ be a set of simple walks in G starting at $r \in V(G)$. For each $e \in E(G)$, we define the prefix set of e in $\mathcal{F}$, denoted by $pfx(\mathcal{F}, r, e)$, as the set of all prefixes for e among the walks in $\mathcal{F}$ that contain e. More formally:*

$$pfx(\mathcal{F}, r, e) := \{pfx(P_i, r, e) \mid i \in [x] \wedge e \in E(P_i)\}.$$

If there is no walk in $\mathcal{F}$ that contains e, then $pfx(\mathcal{F}, r, e) = \emptyset$.

The main idea behind our definition of the prefix-sharing version of the problem is as follows. We want to modify the cost function so that if $H_1, \ldots, H_t$ is a set of trees that forms a solution to a power network design problem, the number of times we pay the cost of digging a trench at a particular edge $e \in E(G)$ is equal to the number of different routes by which a tree H_i can arrive at edge e. Intuitively, this captures the idea that if the power network is installed starting at the root vertex, then digging costs are shared among different trees that place a cable in the same trench, as long as the work crews arrive at this edge by the same path from the root. This can be formalized as saying that the number of times the digging cost $\mathrm{dig}(e)$ has to be paid is equal to the size of the prefix set $\mathrm{pfx}(F, r, e)$, when taking $\mathcal{F}$ to be the following family of paths: for each tree H_i that contains edge e, the set $\mathcal{F}$ contains the path in H_i connecting e to the root r.

When incorporating this concept into the definition of a power network design problem as in Section 1, a surprising feature arises. To minimize the resulting cost function it might be beneficial for a single power cable H_i to be placed in a network of trenches that does not form a tree, but admits cycles among its trenches: the (seemingly superfluous) additional edges that form cycles can lead to a reduction in the number of different prefixes formed in combination with other power cables H_j. In our formulation of the problem, we therefore allow cycles in the connected subgraph H_i that represents the trenches into which a single power cable is placed. To avoid short-circuits, the route by which electricity is transferred from the root to a terminal is still a simple path, but different terminals fed by the same power cable can use different paths through the (possibly cyclic) network of trenches that is dug. Using this intuition and the definitions above, the Prefix-Sharing Low-voltage Network Design problem can now be defined as follows.

PREFIX-SHARING LOW-VOLTAGE NETWORK DESIGN (PLVND) ***Parameter:*** $t + |T|$

Instance: An undirected graph G with *edge lengths* $\ell\colon E(G) \to \mathbb{N}$ and *digging costs* $\mathrm{dig}\colon E(G) \to \mathbb{N}$, a *root vertex* $r \in V(G)$, a set of degree-1 *terminal* vertices $T \subseteq V(G)$ with *demands* $d\colon T \to \mathbb{N}$, and integers α, β, λ, and t, encoded in unary.

Question: Does there exist a partition $\mathcal{P}$ of T into t sets $T_1, \ldots, T_t$ and a function $f\colon T \to 2^{E(G)}$ that maps each terminal $u \in T$ to a path in G from r to u such that:

- for all $u \in T$, the path $P = f(u)$ satisfies $\sum_{e \in E(P)} \ell(e) \leq \lambda$ [*depth* constraint],
- $\sum_{u \in T_i} d(u) \leq \alpha$ for all $i \in [t]$ [*demand* constraint], and
- $\mathrm{cost}_r^{\mathrm{PLV}}(\mathcal{P}, f) \leq \beta$?[*cost* constraint]

The value $\mathrm{cost}_r^{\mathrm{PLV}}(\mathcal{P}, f)$, referred to as PLV-cost, is defined as follows:

$$\mathrm{cost}_r^{\mathrm{PLV}}(\mathcal{P}, f) := \sum_{\substack{T' \in \mathcal{P} \\ e \in E(G)}} (\ell(e) \cdot |\mathrm{pfx}(f(T'), r, e)|) + \sum_{e \in E(G)} \left(\mathrm{dig}(e) \cdot \left| \bigcup_{T' \in \mathcal{P}} \mathrm{pfx}(f(T'), r, e) \right| \right).$$

Contrasting the W[1]-hardness of LOW-VOLTAGE NETWORK DESIGN and HIGH-VOLTAGE NETWORK DESIGN, the PREFIX-SHARING LOW-VOLTAGE NETWORK DESIGN problem is in FPT. We can employ the Dreyfus-Wagner dynamic-programming approach that solves the STEINER TREE problem to solve PLVND. For this algorithm, instead of guessing the abstract structure and using CSP for the realization of this abstract structure, which are the main bottlenecks of the previous algorithms, we branch on and fix the partition of terminals and use a dynamic programming approach to check the existence of a desired solution.

▶ **Theorem 3.7 (⋆).** *PREFIX-SHARING LOW-VOLTAGE NETWORK DESIGN can be solved in time* $2^{|T| \cdot \mathcal{O}(\log t)} \cdot |V(G)|^2 \cdot \lambda$ *for any arbitrary input instance* $(G, \ell, \mathrm{dig}, r, T, d, \alpha, \beta, \lambda, t)$.

4 Hardness results

For our hardness results, we provide reductions from GRID TILING, which takes as input a $k \times k$ matrix where each cell $(i, j) \in [k]^2$ contains a set of pairs $S_{i,j} \subseteq [n] \times [n]$. The task is to decide whether there are $s_{i,j} \in S_{i,j}$ for each (i, j) such that for all $i, i', j, j' \in [k]$, $s_{i,j} = (a, b)$ and $s_{i,j'} = (a', b')$ implies $a = a'$, and $s_{i,j} = (a, b)$ and $s_{i',j} = (a', b')$ implies $b = b'$. GRID TILING cannot be solved in time $f(k) \cdot n^{o(k)}$ under the ETH and is W[1]-hard [9].

4.1 W[1]-hardness of Low-voltage Network Design on planar graphs

Our first reduction shows that our algorithm for planar inputs of LVND is ETH-tight.

▶ **Theorem 4.1.** *The LOW-VOLTAGE NETWORK DESIGN problem is W[1]-hard for* $t = 2$ *and under the ETH admits no* $f(|T|, t) \cdot (|V(G)| \cdot \lambda)^{f'(t) \cdot o(|T|)}$*-time algorithm for any computable functions* f *and* f'*, even when restricted to planar input graphs.*

Proof. From a GRID TILING instance $(S_{i,j})_{(i,j) \in [k]^2}$, construct an LVND-instance as follows.

***G, r* and *T*.** Let G^0 be a grid graph with $N := k \cdot n$ rows and $M := k \cdot (k+1) \cdot n$ columns. For notational convenience, for $i \in [N]$ and $j \in [M]$, we denote the vertex in the i-th row and j-th column of G^0 by $v[i, j]$. For $i \in [N]$ and $j \in [M-1]$, we denote the edge $v[i, j]v[i, j+1]$ by $e[i, j]$. We define G by adding the following vertices to G^0:

- A vertex r_{cons} which we call *row consistency root* and make adjacent to all $v[i,1]$ with $i \in [N]$ and to a new terminal $p(r_{\text{cons}})$ as pendant.
- A vertex r_{select} which we call *selection root* and make adjacent to all $v[1,(j-1)(k+1)+1]$ with $j \in [k \cdot n]$ and to a new terminal $p(r_{\text{select}})$ as pendant.
- The root vertex r of the LVND-instance which we make adjacent to r_{cons} and r_{select}.
- For each $i \in [k]$,
  - a vertex t_i called a *row consistency preterminal* (for value of row i) and connect it to $v[i', M]$ for all $i' \in \{(i-1)\cdot n+1, \ldots, i \cdot n\}$ and to a new terminal $p(t_i)$ as pendant.
  - a vertex t'_i called a *selection preterminal* (for value of column i) and make adjacent to $v[N, (i-1)n(k+1)+i'(k+1)]$ for all $i' \in [n]$ and to a new terminal $p(t'_i)$ as pendant.

By construction G is planar; an embedding is indicated in Figure 2.

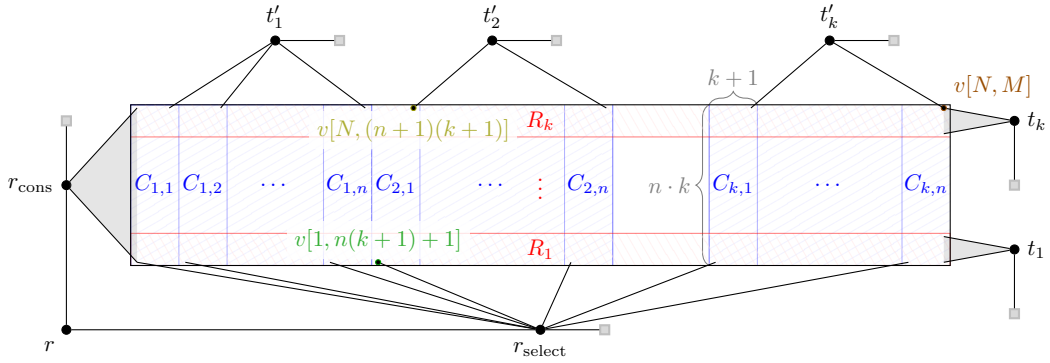


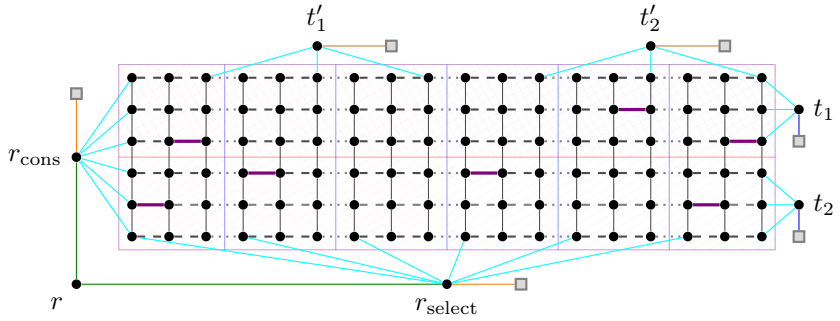
Figure 2 Illustration of the graph G in the proof of Theorem 4.1. Terminals are shown as squares; only selected vertices of G^0 and their indices are indicated explicitly. The hatched rectangle represents G^0, with $v[1,1]$ at the bottom left and $v[N,M]$ at the top right corner.

***t*, *d* and α.** We fix $t = 2$, i.e. a hypothetical solution for the LVND-instance under construction consists of two trees H_1 and H_2. Consider the following partition $\mathcal{P}$ of terminals into two parts: 1) $p(r_{\text{cons}})$ along with all row consistency terminals, and 2) $p(r_{\text{select}})$ along with all selection preterminals; more formally, $\mathcal{P} = \{\{p(r_{\text{cons}}), p(t_1), \ldots, p(t_k)\}, \{p(r_{\text{select}}), p(t'_1), \ldots, p(t'_k)\}\}$. By Lemma 2.3, there exists a function $f : T \to [|U|^4]$ such that $\mathcal{P}$ is the unique partition of terminals into at most two parts that the summation of f-values in each part is at most $\sum_{u \in T} \frac{f(u)}{2}$. Since the demand values are integers, by setting the demand function $d = f$ and $\alpha = \lfloor \sum_{u \in T} \frac{d(u)}{2} \rfloor$, the aim is to find two trees H_1, H_2 where H_1 contains the first part and H_2 contains the second part of the terminals.

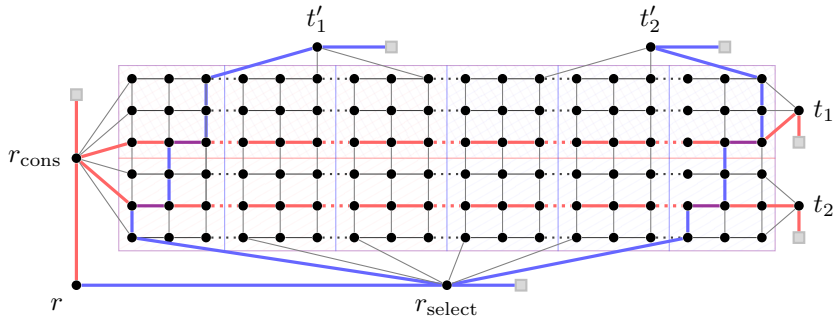
***ℓ*, dig, λ and β.** In the following it will be useful to consider certain subgrids in G^0, specifically ones (see Figure 2) of the form $C_{j,j''} := G^0[\{v[i,j'] \mid i \in [N] \wedge \exists z \in [k+1] \colon j' = (j-1)(k+1)n + (j''-1)(k+1) + z\}]$ for $(j,j'') \in ([k] \cdot [n])$ which we use to encode choices for the second entry in pairs for the j-th column of the Grid Tiling instance, and subgrids of the form $R_i := G^0[\{v[i',j] \mid i' \in \{(i-1)n+1, \ldots, i \cdot n\}, j \in [M]\}]$ which we use to encode choices for the first entry in pairs for the i-th row of the Grid Tiling instance. To specify the edge lengths, we distinguish some horizontal edges. In particular, we will set the lengths of some horizontal edges to be higher $(k+1)$ than that of others (1) and refer to the longer edges (see Figure 3a) as *blocking*:

- For $(j, j'') \in ([k] \times [n]) \setminus \{(k, n)\}$, we call horizontal edges that connect $C_{j,j''}$ to $C_{j,j''+1}$ if $j'' < n$ and to $C_{j+1,1}$ otherwise *column inconsistency blocking*.
- For $j, j' \in [k]$, and $j'' \in [n]$, we call edges between vertices in the j'-th and $(j'+1)$-th column in $C_{j,j''}$ *invalid selection blocking* if they are not in $R_{j'}$ or they are in the i'-th row of $R_{j'}$ for some $i' \in [n]$ for which $(i', j'') \notin S_{j',j}$.

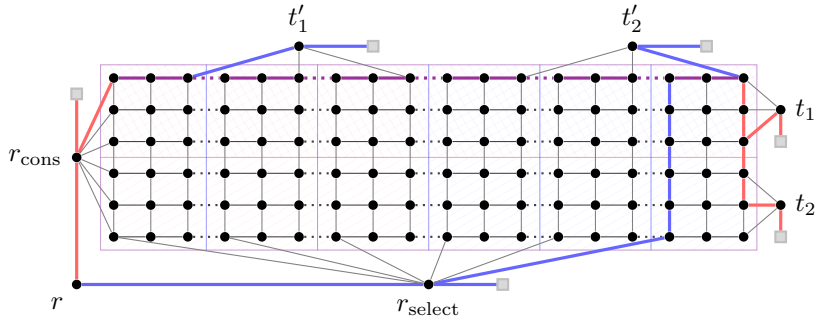
Horizontal edges of G^0 that are not blocking receive length 1. There are at most kn such edges per row. The edges with precisely one endpoint in $V(G^0)$ receive length $X := \sum_{e \in E(G^0)} 2\ell(e) + \mathrm{dig}(e) + 1$ each. (While we have not specified all lengths and the digging



(a) The LVND-instance corresponding to a Grid Tiling instance with $n = 3$, $k = 2$, $S_{1,1} = \{(2,1),(3,2)\}$, $S_{1,2} = \{(2,3),(3,1)\}$, $S_{2,1} = \{(1,1)\}$, and $S_{2,2} = \{(1,3),(2,2)\}$. Equal colors and edge styles indicate equal lengths. Dotted and dashed edges represent column inconsistency and invalid selections and are too long to appear in an r_{select}-t'_1-path in H_2. The purple edges are the only length-1 edges. These are the only horizontal edges that can appear in such a path.



(b) A solution for the instance in Figure 3a. Red and blue denote the two trees; purple edges are shared and indicate the selected cell pairs $s_{1,1} = (2,1)$, $s_{1,2} = (2,3)$, $s_{2,1} = (1,1)$, and $s_{2,2} = (1,3)$.



(c) Illustrating why the construction fails for HVND when $\lambda = +\infty$. The red and blue trees share purple edges and achieve lower cost than the solution in Figure 3b.

Figure 3 Illustration of the reduction from Grid Tiling.

costs of edges in $E(G^0)$ yet, they will not depend on X.) The edges between r and r_{cons} and between r and r_{select} receive length X^* each, where X^* denotes the sum of the length and digging costs of all edges other than rr_{select}, rr_{cons}, and edges incident to terminals. The edges $r_{\text{cons}}p(r_{\text{cons}})$ and $r_{\text{select}}p(r_{\text{select}})$ receive length $\lambda - X^*$, with λ as defined below. For $i \in [k]$, the edge $t'_i p(t'_i)$ receives length 2 and the edge $t_i p(t_i)$ receives length $\lambda - X^* - 2X - (M-1)(k+1)$. The length of all vertical edges is set to $2k^2n(k+1)+1$.

The digging cost of horizontal edges is $nk^2(k+1)$, and the digging cost of all other edges is 0. Finally, we set $\lambda = X^* + 2X + 2 + (N-1)(2k^2n(k+1)+1) + k$. This will constrain each root-terminal path in H_2 to be of minimum length. The total cost threshold β is the sum of k-times this minimum length minus X^* (one for each path to a selection terminal), k-times λ minus X^* (one for each path to a row consistency terminal), two times λ to connect $p(r_{\text{cons}})$ via r_{cons} and $p(r_{\text{select}})$ via r_{select}, and lastly the digging cost necessarily incurred by k horizontal rows of G^0. This completes the LVND-instance.

We will show that $(S_{i,j})_{(i,j)\in[k]^2}$ and $\mathcal{I} := (G, \ell, \text{dig}, r, T, d, \alpha, \beta, \lambda, t)$ are equivalent. In the full version we show how to construct a solution to $\mathcal{I}$ from one to $(S_{i,j})_{(i,j)\in[k]^2}$; see Figure 3b. For the backward direction, assume that $\{H_1, H_2\}$ is a solution for $\mathcal{I}$. To derive a solution for Grid Tiling, it is useful to analyze the behavior of H_1 and H_2. The lengths of the edges, in particular between r and r_{cons} and r and r_{select}, and β are chosen in such a way that each terminal-root-path in H_1 passes through r_{cons} and each terminal-root-path in H_2 passes through r_{select}. By construction, for each $j \in [k]$, the path from r_{select} to $p(t'_j)$ in H_2 contains t'_j. The implied paths from r_{select} to t'_j in H_2 turn out to have a very specific structure.

▷ Claim 4.2 (⋆). For each $j \in [k]$ the interior of the path in H_2 from r_{select} to t'_j is contained in $C_{j,j''}$ for some $j'' \in [n]$ and consists of exactly $N-1$ vertical and exactly k horizontal edges, of which the latter have length 1.

Similarly, for each $i \in [k]$, the path from r_{cons} to $p(t_i)$ in H_1 contains t_i. The implied paths from r_{cons} to t_i in H_1 can also be shown to have a specific structure.

▷ Claim 4.3 (⋆). For each $i \in [k]$, the interior of the r_{cons}-t_i-path in H_1 is one row of R_i.

Our choice of β and dig prohibits horizontal edges that are not in H_1 to be dug for H_2.

▷ Claim 4.4 (⋆). Every horizontal edge in H_2 is also in H_1.

We can use the above analysis of arbitrary solutions for $\mathcal{I}$ to show:

▷ Claim 4.5. If $\mathcal{I}$ is a yes-instance then $(S_{i,j})_{(i,j)\in[k]^2}$ is a yes-instance.

Proof. Let $\{H_1, H_2\}$ be a solution set for $\mathcal{I}$. We show that selecting for each $(i,j) \in [k]^2$, the pair (a,b) such that the i-th horizontal edge of the r_{select}-t'_j-path in H_2 is in the a-th row of R_i and in $C_{j,b}$, yields a solution to the Grid Tiling instance from which we started the construction. By Claim 4.2, the horizontal edges in r_{select}-t'_j-paths in H_2 have length 1. Fix $j \in [k]$ and let P be the r_{select}-t'_j-path in H_2. Because of which horizontal edges receive length 1, all horizontal edges of P are contained in pairwise distinct R_i meaning that we indeed select a pair for each $(i,j) \in [k]^2$. Also by the definition of ℓ, the i-th horizontal edge of P (i.e. the one between the i and $(i+1)$-th column of $C_{j,b}$) having length one implies that it is in the b-th row in R_i, and $(a,b) \in S_{i,j}$. Moreover, as by Claim 4.2, P is entirely contained in $C_{j,b}$, this choice implies that all pairs selected for cells in column j have the same second entry. Lastly, horizontal edges of H_2 have to be contained in H_1 by Claim 4.4. By Claim 4.3, there is precisely one row of each R_i contained in H_1, implying that all selected pairs in the same row share their first entry. ◁

Overall, we have shown that a Grid Tiling-solution for $(S_{i,j})_{(i,j)\in[k]^2}$ implies a LVND-solution for $\mathcal{I}$. This concludes the proof of correctness of our reduction. Notice that the number of terminals in the LVND-instance is linear in k from the Grid Tiling instance, that $t = 2$ is constant, and that $|V(G)|, \lambda \in n^{\mathcal{O}(1)}$. Hence if there were f, f' such that LVND can be solved in time $f(|T|, t) \cdot (|V(G)| \cdot \lambda^{f'(t)})^{o(|T|)}$ for any instance $\mathcal{I}$ then we could solve the original Grid Tiling instance in time $\tilde{f}(k) \cdot n^{o(k)}$, contradicting the ETH. ◀

4.2 W[1]-hardness of High-voltage Network Design

In this section, we show that HVND is also W[1]-hard. We reduce from a Grid Tiling-variant, Regular Grid Tiling, which restricts the instances of Grid Tiling such that for each $x \in [n]$ and each $i, j \in [k]$, the number of pairs of the form (x, y) in $S_{i,j}$ is equal. A proof of the W[1]-hardness of Regular Grid Tiling can be found in the full version.

▶ **Theorem 4.6 (⋆).** *The High-voltage Network Design problem is W[1]-hard even for $t = 4$ and under the ETH admits no $f(|T|, t) \cdot |V(G)|^{o(\sqrt{|T|}) \cdot f'(t)}$-time algorithm for any computable functions f and f', even when restricted to planar input graphs.*

Proof sketch. Again, to encode a choice of pairs of numbers for each cell of the Grid Tiling-instance our HVND-solution should consist of two parts, one selecting k rows and one selecting k columns, such that the intersection of a selected row and a selected column consists of a single shared horizontal edge, which needs to be dug only once.

However, Theorem 4.1 crucially relies on the depth constraint in LVND. In particular, the depth constraint allowed us to severely restrict which paths could be used to connect terminals to r in the solution. If we use the same reduction structure and set $\lambda = +\infty$, then instead of having a path from r_{cons} (r_{select}) to t_i (t'_i) for each $i \in [k]$, a minimum-cost solution has a fork-like structure consisting of a single row (column) that then branches out on the last column (row) to reach all terminals on the other side of the grid (see Figure 3c). Hence, parts of the solution that should correspond to choices in separate rows or columns collapse into one. To prevent this, we introduce gadgets, called *connectors*, that contain further terminals throughout the inner parts of the grid. Concretely, the grid is divided into $k \times k$ subgrids of size $N \times M$, called *cell gadgets*. Whenever two cell gadgets are adjacent in the original grid, their corresponding boundary vertices are no longer directly connected; instead, they are each linked via a connector which contains terminals that force the part of the solution that must cover them to connect through it and prevent it from collapsing.

A connector consists of two intersecting types of paths; *linking* paths between cell gadgets and *supporting* paths between terminals. If the cell gadgets lie in the same row (column), the connector is called a *row (column) connector*. A schematic overview of the full graph is depicted in Figure 4. Linking paths have high lengths. In a minimum-cost solution, this prevents a tree encoding row (column) choices to intersect column (row) connectors.

To ensure that within a row (column) connector the solution does not switch between rows (columns), the final construction enforces a solution consisting of four (two for row and two for column choices) trees rather than two (one for row and one for column choices) (See Figure 5). The two trees encoding row (column) choices have almost identical structure to avoid any extra digging cost. However, they are each forced to cover terminals on opposite sides of the connectors. Together, these properties ensure that row choices and column choices are made independently and remain consistent across the grid. ◀

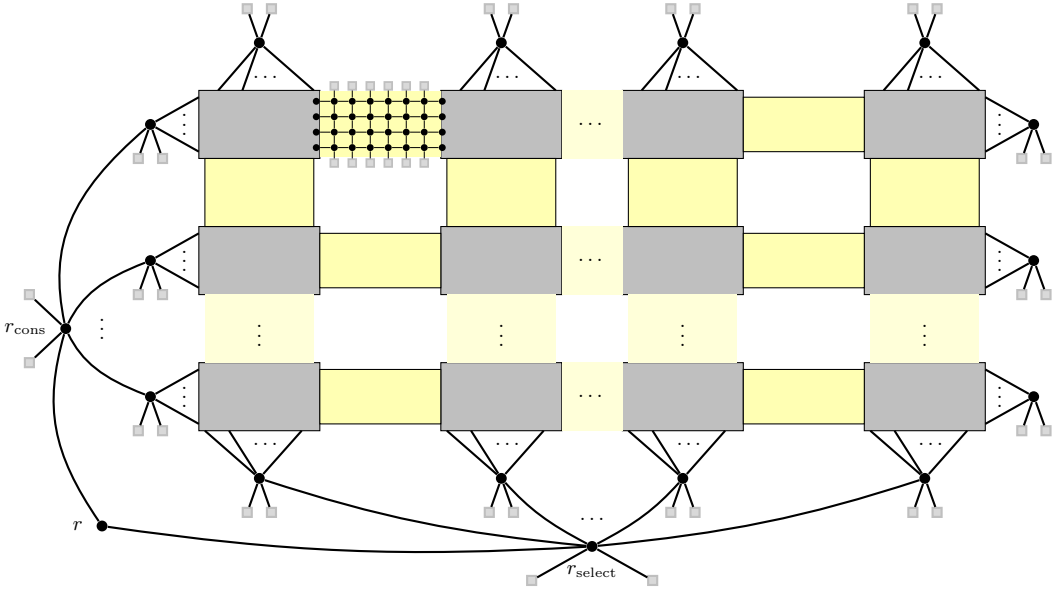
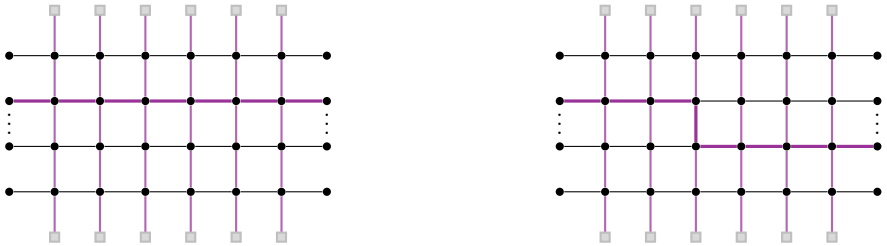
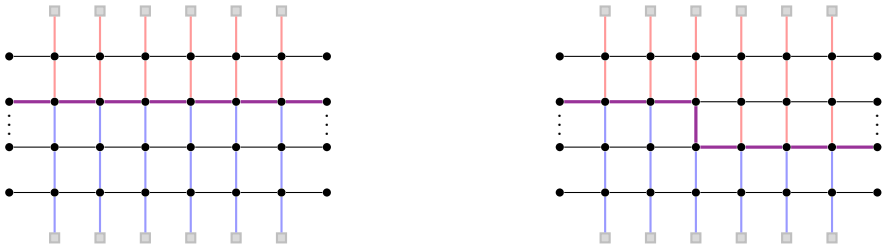


Figure 4 Schematic view of the reduction in Theorem 4.6. The top-left row connector is shown in detail. Other connectors are indicated in yellow and cell gadgets in gray.



(a) Purple edges are of trees, each passing through a connector and covering all terminals. Left: the tree intersects only one row. Right: the tree intersects/switches between two rows. Both trees have equal cost.



(b) Purple edges are shared by pairs of trees with red resp. blue private edges, each passing through a connector and covering all terminals. Left: trees intersects one row. Right: the pair of trees switches between two rows and has higher cost due to counting the vertical purple edge twice.

Figure 5 Illustration how the use of four trees (two encoding horizontal choices, two encoding vertical choices) to encode a solution facilitates a reduction from Grid Tiling to HVND.

5 Conclusion

In this paper, we introduced several power network design problems that arose from real-world applications. Despite their similarities to Steiner Tree, we showed that the parameterization by the number of terminals and solution trees is W[1]-hard. We leave the exploration of

other parameterizations, such as treewidth, to future work. A distinguishing feature of our power network design problems, which are shared by Minimum-Cost Shared Arborescence [1], is that a solution consists of multiple parts and a suitable sharing of resources among the parts leads to a lower overall cost. It would be interesting to introduce this setting to other problems. For example, consider the Subset Traveling Salesperson Problem in which the goal is to find a minimum-cost closed walk that visits all terminals. One could investigate the version in which we look for a set of closed walks that together visit all terminals, with the number of terminals visited by a single walk upper-bounded by a given threshold, while minimizing a weighted sum of the total length of the individual walks and the *visiting costs* of the edges in the union of the walks. Which algorithmic approaches for solving Subset TSP carry over to this setting?

References

1 Eduardo Álvarez-Miranda, Ivana Ljubic, Martin Luipersbeck, and Markus Sinnl. Solving minimum-cost shared arborescence problems. *Eur. J. Oper. Res.*, 258(3):887–901, 2017. doi:10.1016/J.EJOR.2016.11.004.

2 Matthias Bentert, André Nichterlein, Malte Renken, and Philipp Zschoche. Using a geometric lens to find k disjoint shortest paths. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.26.

3 Paolo Bertoldi, Javier López-Lorente, and Nicola Labanca. Energy consumption and energy efficiency trends in the eu-28 2000-2014. *Jt. Res. Cent. Ispra Italy*, 1:1–175, 2016.

4 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

5 Xiu Zhen Cheng and Ding-Zhu Du, editors. *Steiner Tree Problems in VLSI Layout Designs*, pages 101–173. Springer US, Boston, MA, 2001. doi:10.1007/978-1-4613-0255-1_4.

6 Markus Chimani and Joachim Spoerhase. Network design problems with bounded distances via shallow-light steiner trees. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 238–248. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.238.

7 Rajesh Chitnis. A tight lower bound for edge-disjoint paths on planar dags. *SIAM J. Discret. Math.*, 37(2):556–572, 2023. doi:10.1137/21M1395089.

8 Rajesh Chitnis, Samuel Thomas, and Anthony Wirth. Lower bounds for approximate (& exact) k-disjoint-shortest-paths. In Marcin Bienkowski and Matthias Englert, editors, *Approximation and Online Algorithms - 22nd International Workshop, WAOA 2024*, volume 15269 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2024. doi:10.1007/978-3-031-81396-2_4.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M.M. Van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Transactions on Algorithms (TALG)*, 18(2):1–31, 2022. doi:10.1145/3506707.

11 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

12 11th DIMACS Implementation Challenge, 2014. Accessed 21 Nov 2023. URL: https://dimacs11.zib.de/.

13 Stuart E. Dreyfus and Robert A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

14 European Parliament. Report on electricity grids: the backbone of the eu energy system (2024/2226(ini)), 2025. Document A-10-0091/2025, Rapporteur: Anna Stürgkh. URL: https://www.europarl.europa.eu/doceo/document/A-10-2025-0091_EN.html.

15 Andreas Emil Feldmann and Dániel Marx. The complexity landscape of fixed-parameter directed steiner network problems. *ACM Trans. Comput. Theory*, 15(3-4):4:1–4:28, 2023. doi:10.1145/3580376.

16 Fedor V Fomin, Petteri Kaski, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Parameterized single-exponential time polynomial space algorithm for Steiner tree. *SIAM Journal on Discrete Mathematics*, 33(1):327–345, 2019. doi:10.1137/17M1140030.

17 Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 1*, AAAI'90, pages 4–9. AAAI Press, 1990. URL: http://www.aaai.org/Library/AAAI/1990/aaai90-001.php.

18 Bernhard Fuchs, Walter Kern, D Molle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41:493–500, 2007. doi:10.1007/S00224-007-1324-4.

19 M. R. Garey and David S. Johnson. The rectilinear steiner tree problem is NP complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

20 Carla Groenland, Jesper Nederlof, and Tomohiro Koana. A polynomial time algorithm for steiner tree when terminals avoid a rooted K_4-minor. In Édouard Bonnet and Pawel Rzazewski, editors, *19th International Symposium on Parameterized and Exact Computation, IPEC 2024, September 4-6, 2024, Royal Holloway, University of London, Egham, United Kingdom*, volume 321 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.IPEC.2024.12.

21 Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximating buy-at-bulk and shallow-light k-steiner trees. *Algorithmica*, 53(1):89–103, 2009. doi:10.1007/S00453-007-9013-X.

22 Bart M. P. Jansen and Céline M. F. Swennenhuis. Steiner tree parameterized by multiway cut and even less. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 76:1–76:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.76.

23 H.C Joksch. The shortest route problem with constraints. *Journal of Mathematical Analysis and Applications*, 14(2):191–197, 1966. doi:10.1016/0022-247X(66)90020-5.

24 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

25 Guy Kortsarz and David Peleg. Approximating shallow-light trees (extended abstract). In Michael E. Saks, editor, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5-7 January 1997, New Orleans, Louisiana, USA*, pages 103–110. ACM/SIAM, 1997. URL: http://dl.acm.org/citation.cfm?id=314161.314191.

26 Ivana Ljubić. Solving Steiner trees: Recent advances, challenges, and perspectives. *Networks*, 77(2):177–204, 2021. doi:10.1002/net.22005.

27 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In *Proc. 42nd STOC*, pages 321–330. ACM, 2010. doi:10.1145/1806689.1806735.

28 Dániel Marx. On the optimality of planar and geometric approximation schemes. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 338–348. IEEE Computer Society, 2007. doi:10.1109/FOCS.2007.50.

29 Dániel Marx. The square root phenomenon in planar graphs. In Michael R. Fellows, Xuehou Tan, and Binhai Zhu, editors, *Frontiers in Algorithmics* and *Algorithmic Aspects in Information and Management, Third Joint International Conference, FAW-AAIM 2013*, Lecture Notes in Computer Science. Springer, 2013. doi:10.1007/978-3-642-38756-2_1.

30 Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. On subexponential parameterized algorithms for steiner tree and directed subset TSP on planar graphs. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 474–484. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00052.

31 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/S00453-012-9630-X.

32 PACE 2018, 2018. Accessed 21 Nov 2023. URL: https://pacechallenge.org/2018/.

33 Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. *ACM Trans. Algorithms*, 14(4):53:1–53:73, 2018. doi:10.1145/3239560.

34 Ondrej Suchý. On directed steiner trees with multiple roots. In Pinar Heggernes, editor, *Graph-Theoretic Concepts in Computer Science - 42nd International Workshop, WG 2016, Istanbul, Turkey, June 22-24, 2016, Revised Selected Papers*, volume 9941 of *Lecture Notes in Computer Science*, pages 257–268, 2016. doi:10.1007/978-3-662-53536-3_22.