Conceptual Framework for Determining the Treewidth of Distribution Grids

Andrew Eliseev
Research Training Group KRITIS
Technical University of Darmstadt
Darmstadt, Germany
eliseev@kritis.tu-darmstadt.de

Florian Steinke
Energy Information Networks & Systems
Technical University of Darmstadt
Darmstadt, Germany
florian.steinke@eins.tu-damstadt.de

***Abstract*—While distribution grids are often operated radially, they are typically designed to be more redundant, so that each load has multiple connections to the main grid. For complex networks like these, the notion of treewidth can be used to quantify their complexity. In this paper, we propose a new conceptual framework and derive an exact formula for computing treewidth with the help of our constructs. We argue that our framework effectively captures complexities in the structure of distribution grids and has a potential to simplify the calculation of treewidth. After analysing our findings, we hypothesise that the treewidth of distribution grids will typically be low implying that some difficult power system problems can be solved on them in parameterised polynomial time with dynamic programming. We demonstrate this with an example problem of dividing a distribution grid into tree-like operational subgraphs around the primary substations so that no voltage violations occur.**

***Index Terms*—distribution grid, dynamic programming, tree decomposition, treewidth.**

I. Introduction

Dynamic programming is an algorithm design paradigm that has been successfully applied to many problems in power systems such as, e.g. state estimation [1], [2], operational topology estimation [3], or optimal dispatch [4], [5], [6], [7]. In cases where the power system graph is a tree, dynamic programming algorithms can provide exact solutions to non-convex optimal power flow formulations while showing linear time complexity w. r. t. the size of the graph [5]. More complex structure of the graph, however, can obstruct the straightforward application of dynamic programming. A pragmatic approach in this case usually involves the development of approximation schemes or heuristics [4], [8], [9]. These often work well in practice but do not provide strong performance guarantees.

Nevertheless, even for complex networks, dynamic programming can retain its provable performance and practical utility provided that both the underlying network and the specific problem exhibit exploitable structural properties. One such structural property discussed in this paper is treewidth.

This work has been funded by the DFG (German Research Foundation) within the Research Training Group KRITIS.

Treewidth is an indicator that shows 'how close' a graph is to being a tree, which is measured with the help of tree decompositions. Tree decompositions embed the original graph into a tree of subsets of its vertices that have certain valuable properties. Treewidth shows how small the cardinality of these subsets can be for the given graph. Over past decades, many problems including some NP-hard problems have been shown to be fixed-parameter tractable when parameterised by treewidth [10]. This means that there exists an exact (dynamic programming) algorithm that solves the problem for any graph G in time $O(f(t) \cdot |G|^d)$ where $f(t)$ is a monotonic non-decreasing function that only depends on the treewidth t of G, $|G|$ the size of graph G and $d \in \mathbb{Z}_{\geq 1}$ is a constant. Usually $f(t) \in \Theta(c^t)$ where $c \in \mathbb{Z}_{\geq 2}$ is a constant. If the graphs one has to work with have a fixed treewidth or a universal upper bound on their treewidth (say, some k), then such algorithms achieve parameterised polynomial time complexity, since $f(t)$ in the asymptotic time estimation can be replaced by its constant upper bound $f(k)$. The smaller the value of k is, the better theoretical performance such algorithms will have.

When it comes to electricity networks, their treewidth has been previously studied in [11] where a series of computational experiments showed that values can reach two-digit numbers for transmission grids making dynamic programming algorithms based on tree decompositions practically infeasible due to a huge leading coefficient $f(t)$ of the polynomial. Besides this, in [12] it was argued that many electrical circuits can be represented with series-parallel graphs, an important class of graphs with bounded treewidth.

In this work we advance the knowledge about treewidth for power systems in the following way:

1) We define a new conceptual framework that effectively describes complexities in the topology of real distribution grids on the medium- (MV) and low-voltage (LV) levels.
2) With the help of our notions, we derive a formula to compute the exact value of the treewidth for any distribution grid.
3) By analysing the derived formula, we hypothesise that real distribution grids are likely to have low treewidth.
4) We showcase benefits of low treewidth with an example problem of voltage control in distribution grids by explic-

itly constructing a dynamic programming algorithm that solves it in parametrised quadratic time.

Besides the algorithm shown in the paper, there are several general schemes of designing dynamic programming algorithms based on tree decompositions for certain important classes of problems [13], [14].

The rest of the paper is structured as follows. In Section II, we present our notation and review some previously known definitions and results. Next, in Section III, we define our novel concepts. Section IV contains rigorous proofs of all our theoretical results that show how our concepts can be used to compute the treewidth of distribution grids. In Section V, we demonstrate how treewidth can be applied to an example problem of voltage control in distribution grids. Finally, we conclude in Section VI.

II. Notation & Background

We begin by presenting essential definitions important for the rest of the paper.

First and foremost, we stipulate that all graphs considered in this work are finite, free of self-loops or parallel edges. For any graph G, we denote the set of its vertices and edges as $\mathrm{V}(G)$ and $\mathrm{E}(G)$ respectively. In the case of an undirected graph G, we assume that $\mathrm{E}(G)$ contains doubleton subsets (subsets of cardinality 2) of $\mathrm{V}(G)$. $G[A]$, where $A \subseteq \mathrm{V}(G)$, denotes a subgraph of G induced by the vertices in A.

Let us now define tree decompositions, the first important construct of our work. The idea behind them is to represent a complex network G as a tree T, each vertex b of which corresponds to some subset X_b of vertices of G. The decomposition is also structured in a way so that every edge in the original graph is represented by at least one vertex in the tree.

Definition 1 (Tree decomposition). Let G be an undirected graph. Tuple (T, X) with tree T and family X of subsets of $\mathrm{V}(G)$ indexed by the vertices of T is a *tree decomposition* of G if the following properties are satisfied:

- $\bigcup_{b \in \mathrm{V}(T)} X_b = \mathrm{V}(G)$.
- For any $e \in \mathrm{E}(G)$ there exists $b \in \mathrm{V}(T)$ such that $e \subseteq X_b$.
- For any $v \in \mathrm{V}(G)$ $T[\{b \in \mathrm{V}(T) | v \in X_b\}]$ is a tree.

For convenience, sets from X will be also called *bags*.

A tree decomposition exists for any graph G. For example, it can be constructed for $n \in \mathbb{Z}_{\geq 1}$ by defining (T, X) with $\mathrm{V}(T) = \{b_i\}_{i=1}^{n}$, $\mathrm{E}(T) = \{\{b_i, b_{i+1}\}\}_{i=1}^{n-1}$ and $X_{b_i} = \mathrm{V}(G)$. It is easy to check that the above (T, X) satisfies the requirements of Definition 1.

It is also clear from this example that tree decompositions are not unique. In fact, since the value of n is unbounded, one can construct an infinite number of tree decompositions for any graph. A common way to measure the quality of a tree decomposition is to compute its width.

Definition 2 (Width). Let G be an undirected graph and (T, X) be its tree decomposition. Then the *width* of (T, X) is

$$\mathrm{w}(T, X) = \max_{b \in V(T)} |X_b| - 1.$$

An example graph G together with its two different tree decompositions can be found in Fig. 1.

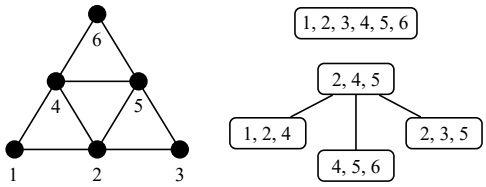

Fig. 1. An example graph and its possible tree decompositions of width 5 (upper) and of width 2 (lower).

Following the idea from Definition 2, we can define a quality parameter for the graph itself.

Definition 3 (Treewidth). Let G be an undirected graph and $\mathcal{T}$ be the set of all its tree decompositions. The *treewidth* of G is then

$$\mathrm{tw}(G) = \min_{(T,X) \in \mathcal{T}} \mathrm{w}(T, X).$$

Treewidth is always finite because the value $\mathrm{tw}(G)$ is bounded by -1 from below and by $|\mathrm{V}(G)| - 1$ from above.

Theorem 4 presents several well-known facts about treewidth.

Theorem 4. *Let G be an undirected graph. It is known that:*

- $\mathrm{tw}(G) \leq 0$ *if and only if G has no edges.*
- $\mathrm{tw}(G) \leq 1$ *if and only if G is a forest [15, Theorem 65].*
- *Deciding whether $\mathrm{tw}(G) \leq k$ for $k \in \mathbb{Z}_{\geq 1}$ is NP-complete [16, Theorem 3.3]*[1].
- *For any fixed $k \in \mathbb{Z}_{\geq 1}$ there exists an algorithm with the linear parameterised time complexity $O(2^{k^3} \cdot |G|)$ that either constructs a tree decomposition of G of width k or concludes that $\mathrm{tw}(G) > k$ [17, Theorem 1.1].*

The first statement in Theorem 4 is trivial: if and only if there are no edges, we can assign each vertex of G to its own bag and Definition 1 will be satisfied.

Besides what is stated in Theorem 4, there is also a general method to give a bound on the treewidth for an arbitrary graph. This method focuses on analysing the minors of the given graph.

Definition 5 (Minor). Let G be an undirected graph. A *minor* of G is any graph that can be obtained from G by deleting vertices, deleting edges, or contracting edges.

If M is a minor of G, we will to denote this as $M \sqsubseteq G$.

Theorem 6 ([15, Lemma 16]). *Let G be an undirected graph and $M \sqsubseteq G$. Then $\mathrm{tw}(G) \geq \mathrm{tw}(M)$.*

[1]Note that in the mathematical literature the term 'partial k-trees' is sometimes used to describe graphs of treewidth at most k.

Theorem 7 ([15, Theorem 15]). *Let $\mathcal{C}$ be a class of all graphs with treewidth at most k. Then there exists a finite set $\mathcal{F}_k$ of graphs such that $G \in \mathcal{C}$ if and only if $M \not\sqsubseteq G$ for all $M \in \mathcal{F}_k$.*

Such sets $\mathcal{F}_k$ are commonly called 'sets of forbidden minors' or 'obstruction sets'. Theorem 8 presents forbidden minors for low values of k.

Theorem 8 ([15, Theorem 17]). *Let K_n be a complete graph (clique) with $n \in \mathbb{Z}_{\geq 1}$ vertices. Then:*

- $\mathcal{F}_1 = \{K_3\}$.
- $\mathcal{F}_2 = \{K_4\}$.
- $\mathcal{F}_3 = \{K_5, Y_5, M_8, K_{2,2,2}\}$ *(see Fig. 2).*

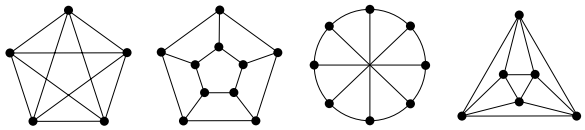

Fig. 2. The graphs from $\mathcal{F}_3$, left to right: K_5, Y_5, M_8, $K_{2,2,2}$.

The number of forbidden minors increases rapidly. While the set $\mathcal{F}_3$ includes only 4 elements, $\mathcal{F}_4$ contains more than 70 different graphs [18].

III. Distribution grids and DG-kernels

Let us now introduce the novel concept of DG-kernels intended to model the principal (in a certain sense) structure of *distribution grids* (hence the letters 'DG' in the name) on the MV and LV levels.

Generally speaking, distribution grids can be seen as an interconnected collection of *buses* representing *nodes* (i.e. points where electricity is consumed or injected into the grid) and *primary* or *secondary substations*. Each primary substation has a link to the transmission grid. It is also connected via *MV feeders* to a certain subset of secondary substations and/or MV nodes. Secondary substations, in turn, are connected via *LV feeders* to a certain subset of LV nodes.

In literature, 4 different structural configurations of feeders in distribution grids are distinguished [19]:

- *Radial configuration* – feeder has a tree structure.
- *Ring configuration without a counter substation* – feeder has both of its ends connected to the same substation.
- *Ring configuration with a counter substation* – feeder has its ends connected to different substations.
- *Meshed configuration* – feeder has more complex structure.

When a feeder has a ring or meshed configuration, a switch on at least one of its lines is usually opened during operation to divide it into several radial feeders. This is done to simplify power flow control and failure localisation.

We address this variety of topologies by defining a class of F-graphs. F-graphs will be the first level of abstraction in our framework that is intended to capture possible radial and ring configurations of *feeders* (hence the letter 'F' in the name) described above.

Definition 9 (F-graph). Let $n \in \mathbb{Z}_{\geq 1}$, $L_1, \ldots, L_n$ be undirected trees such that there are distinct vertices a_1, a_2 present in all $L_1, \ldots, L_n$ with the following properties:

- For any distinct $i, j \in \{1, \ldots, n\}$ $\mathrm{V}(L_i) \cap \mathrm{V}(L_j) = \{a_1, a_2\}$.
- For any distinct $i, j \in \{1, \ldots, n\}$ $\mathrm{E}(L_i) \cap \mathrm{E}(L_j) = \emptyset$.

Then graph $F = L_1 \cup \cdots \cup L_n$ is called an *F-graph with anchors* $\mathrm{A}(F) = \{a_1, a_2\}$.

Remark 10. Any tree with at least 1 edge is an F-graph. Its anchors can be any 2 vertices.

Let us also define different concepts relevant to F-graphs.

Definition 11 (Concepts relevant to F-graphs). Let F be an F-graph with anchors $\mathrm{A}(F) = \{a_1, a_2\}$.

- Any path subgraph of F connecting anchors a_1 and a_2 is called a *line*.
- We define the set of *waypoints* of F as the set $\mathrm{W}(F)$ of all vertices lying on any path between a_1 and a_2 including a_1 and a_2 themselves (these vertices are 'on the way' between the 2 anchors).
- For any $w \in \mathrm{W}(F)$ we call a connected component of F containing w that appears after removing all edges connecting w to other waypoints the *radial subnetwork* of w and denote it $\mathrm{RS}(F, w)$. $\mathrm{RS}(F, w)$ is called *degenerate* if $\mathrm{V}(\mathrm{RS}(F, w)) = \{w\}$.
- An F-graph with 1 line is called *radial*.

F-graphs and their components can have different semantics depending on the context. For example, an F-graph with 2 lines can represent a ring feeder without a counter substation and, at the same time, it can model 2 parallel feeders with the same counter substation. Radial F-graphs can represent 1 or more radial feeders connected to the same substation.

An example of an F-graph can be found in Fig. 3.

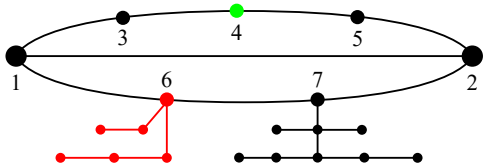

Fig. 3. An example F-graph F with anchors $\mathrm{A}(F) = \{1, 2\}$, waypoints $\mathrm{W}(F) = \{1, 2, 3, 4, 5, 6, 7\}$ and 3 lines: $(1, 3, 4, 5, 2)$, $(1, 2)$ and $(1, 6, 7, 2)$. $\mathrm{RS}(F, 4)$ is degenerate and coloured green, $\mathrm{RS}(F, 6)$ is coloured red.

We now define DG-kernels, the second level of abstraction in our framework, with the help of edge replacement sequences.

Definition 12 (Edge replacement sequence). Let $n \in \mathbb{Z}_{\geq 1}$, $G_1, \ldots, G_n$ be graphs, $e_1, \ldots, e_{n-1}$ be edges from $G_1, \ldots, G_{n-1}$ respectively, $P_1, \ldots, P_{n-1}$ be graphs such that:

- For any $i \in \{1, \ldots, n-1\}$ $\mathrm{V}(P_i) \cap \mathrm{V}(G_i) = e_i$.
- For any $i \in \{1, \ldots, n-1\}$ $(\mathrm{E}(P_i) \cap \mathrm{E}(G_i)) \setminus \{e_i\} = \emptyset$.

Then we say that

$$G_1 \xrightarrow{e_1, P_1} G_2 \xrightarrow{e_2, P_2} \cdots G_{n-1} \xrightarrow{e_{n-1}, P_{n-1}} G_n$$

is an *edge replacement sequence* if for each $i \in \{1, \dots, n-1\}$ G_{i+1} can be obtained from G_i by replacing e_i with P_i.

Definition 13 (DG-kernel). Let G, D be graphs. We say that D is a *DG-kernel* of G if there is an edge replacement sequence

$$(D = G_1) \xrightarrow{e_1, F_1} G_2 \xrightarrow{e_2, F_2} \cdots G_{n-1} \xrightarrow{e_{n-1}, F_{n-1}} (G_n = G)$$

where each F_i is an F-graph with $\mathrm{A}(F_i) = e_i$.

Remark 14. Each graph is its own DG-kernel.

Remark 15. If D is a DG-kernel of G and there is an edge replacement sequence $(D = G_1) \xrightarrow{e_1, F_1} G_2 \xrightarrow{e_2, F_2} \cdots G_{n-1} \xrightarrow{e_{n-1}, F_{n-1}} (G_n = G)$, then each intermediate G_i is also a DG-kernel of G.

Any graph G has a minimum DG-kernel D^*. In general, nothing guarantees the uniqueness of D^*: say, G is a tree with $|\mathrm{V}(G)| > 2$ and let $v_1, v_2 \in \mathrm{V}(G)$ be 2 distinct vertices. By Definition 13, graph D^* with $\mathrm{V}(D^*) = \mathrm{E}(D^*) = \{v_1, v_2\}$ is its DG-kernel. Note that there is no DG-kernel of G with less than 2 vertices, since there should be a non-trivial edge replacement sequence to construct G from such a kernel and, hence, there should be at least 1 edge in it. Thus, D^* is minimum. However, the choice of v_1 and v_2 is arbitrary, therefore, there will be multiple (isomorphic) minimum DG-kernels in this case.

Examples of DG-kernels for SimBench test grid `1-MV-rural--0-sw`[2] can be found in Fig. 4.

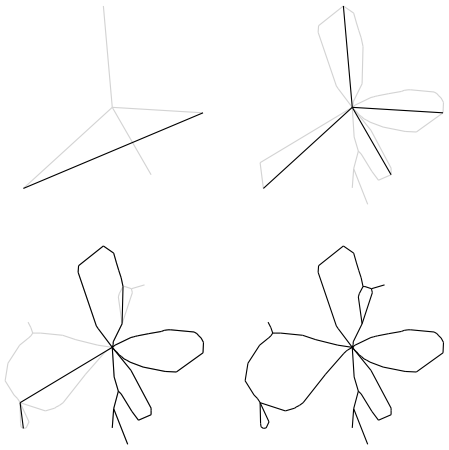

Fig. 4. A minimum DG-kernel (top left) for SimBench test grid `1-MV-rural--0-sw` (bottom right) and a corresponding edge replacement sequence (top left → top right → bottom left → bottom right). The F-graph that a black edge has to be replaced with is shown in grey next to it.

The concept of DG-kernel is defined based on the assumption that F-graph-like structures with a relatively large number of vertices are a recurring pattern in distribution grids. If this assumption holds, it should be possible to find DG-kernels for real distribution grids that are significantly smaller than the original grids.

[2]https://simbench.de/en/

IV. Main theoretical results

In Theorems 16 and 17 we present our theoretical findings on the computation of treewidth using the framework we have developed. Afterwards, we shortly discuss the results and present our hypothesis about the treewidth of distribution grids.

Theorem 16 (Treewidth of F-graphs). *Let F be an F-graph. Then $\mathrm{tw}(F) = 1$ if F is radial and $\mathrm{tw}(F) = 2$ otherwise.*

Proof. In the case of a radial F-graph the statement is trivial: F is a tree and, hence, due to Theorem 4, $\mathrm{tw}(F) = 1$.
Consider now the case when there is more than one line and let L_1 and L_2 be then two different lines of F. It follows from Definition 9 that $C = L_1 \cup L_2$ is a cycle subgraph of F. Every subgraph of a graph is its minor because it can be obtained by removing vertices and edges of the original graph, therefore, $C \sqsubseteq F$. After contracting an appropriate number of edges of C one can see that $K_3 \sqsubseteq C$ and, hence, $K_3 \sqsubseteq F$. Thanks to Theorem 8, we conclude that $\mathrm{tw}(F) \geq 2$.
We will now show that $\mathrm{tw}(F) \leq 2$ by explicitly constructing a tree decomposition (T, X) of F of width 2. Let $\mathrm{A}(F) = \{a_1, a_2\}$ and $L_1, \dots, L_n$ with $n \in \mathbb{Z}_{\geq 2}$ be the lines of F such that every $L_i = (a_1, w_{i,1}, \dots, w_{i,m_i}, a_2)$ for some $m_i \in \mathbb{Z}_{\geq 0}$. For each line L_i construct $(T_i, X^{[i]})$ as follows: $\mathrm{V}(T_i) = \{(i,j)\}_{j=1}^{m_i}$, $\mathrm{E}(T_i) = \{\{(i,j),(i,j+1)\}\}_{j=1}^{m_i-1}$, $X^{[i]}_{(i,1)} = \{a_1, a_2, w_{i,1}\}$, $X^{[i]}_{(i,j)} = \{a_2, w_{i,j-1}, w_{i,j}\}$ for $j \in \{2, \dots, m_i\}$.
Notice that $\mathrm{RS}(F, a_1)$, $\mathrm{RS}(F, a_2)$ and all $\mathrm{RS}(F, w_{i,j})$ are trees, hence, due to Theorem 4, there exist corresponding tree decompositions $(H_1, Y^{[1]})$, $(H_2, Y^{[2]})$ and $(H_{i,j}, Y^{[i,j]})$ for them of width at most 1. For each $w_{i,j}$, choose $b \in \mathrm{V}(H_{i,j})$ such that $w_{i,j} \in Y^{[i,j]}_b$. Define then $H^*_{i,j}$ by adding (i,j) to $H_{i,j}$: $\mathrm{V}(H^*_{i,j}) = \mathrm{V}(H_{i,j}) \cup \{(i,j)\}$, $\mathrm{E}(H^*_{i,j}) = \mathrm{E}(H_{i,j}) \cup \{\{b,(i,j)\}\}$. Radial subnetworks of a_1 and a_2 can be processed similarly.
On the last stage, construct graph P: $\mathrm{V}(P) = \{(i,1)\}_{i=1}^{n}$, $\mathrm{E}(P) = \{\{(i,1),(i+1,1)\}\}_{i=1}^{n-1}$. Finally, define (T, X) as

$$T = P \cup \left(\bigcup_{i=1}^{n} T_i \right) \cup \left(\bigcup_{i=1}^{n} \bigcup_{j=1}^{m_i} H^*_{i,j} \right),$$

$$X = \left(\bigsqcup_{i=1}^{n} X^{[i]} \right) \sqcup \left(\bigsqcup_{i=1}^{n} \bigsqcup_{j=1}^{m_i} Y^{[i,j]} \right) \sqcup Y^{[1]} \sqcup Y^{[2]}.$$

Verification of Definiton 1 for (T, X) is purely technical.
Thus, we constructed a tree decomposition of F where each bag contains at most 3 vertices of F, hence, by Definition 2, $\mathrm{w}(T, X) = 2$ and, therefore, $\mathrm{tw}(F) \leq 2$. Knowing that $\mathrm{tw}(F) \geq 2$ and that $\mathrm{tw}(F) \leq 2$, we conclude that $\mathrm{tw}(F) = 2$. □

A tree decomposition constructed for the F-graph from Fig. 3 with the method used in the proof of Theorem 16 can be found in Fig. 5.

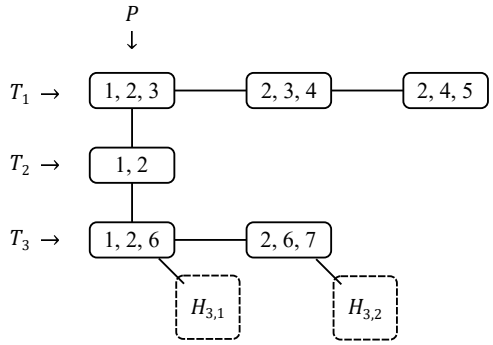

Fig. 5. A tree decomposition of the F-graph from Fig. 3 of width 2 built using the method from the proof of Theorem 16. Bags containing only 1 vertex as well as subtrees $H_{3,1}$ and $H_{3,2}$ corresponding respectively to the decompositions of $\mathrm{RS}(F, 6)$ and $\mathrm{RS}(F, 7)$ are hidden.

Theorem 17 (Treewidth via DG-kernels). *Let G be a graph with DG-kernel D and edge relacement sequence $(D = G_1) \xrightarrow{e_1,F_1} G_2 \xrightarrow{e_2,F_2} \cdots G_{n-1} \xrightarrow{e_{n-1},F_{n-1}} (G_n = G)$. Then*

$$\mathrm{tw}(G) = \max\{\mathrm{tw}(D), z\}$$

where $z \in \{1, 2\}$ and $z = 1$ if and only if all F-graphs $F_1, \ldots, F_{n-1}$ are radial.

Proof. First, note that any F-graph F can be reduced to a graph with a single edge connecting $\mathrm{A}(F)$ by a series of vertex deletions, edge deletions and edge contractions. To see this, it is sufficient to remove all radial subnetworks as well as all but one of the lines of F. The edges of the remaining line can be contracted into a single edge. Thus, $D \sqsubseteq G$ by construction and $\mathrm{tw}(G) \geq \mathrm{tw}(D)$ by Theorem 6. By the same token, all $F_1, \ldots, F_{n-1}$ are also minors of G, hence $\mathrm{tw}(D) \geq \mathrm{tw}(F_i)$ for all $i \in \{1, \ldots, n-1\}$. Thanks to Theorem 16 we conclude that $\mathrm{tw}(G) \geq \max\{\mathrm{tw}(D), \max\{\mathrm{tw}(F_i)\}_{i=1}^{n-1}\} = \max\{\mathrm{tw}(D), z\}$. Construct now a tree decomposition of D of width $\mathrm{tw}(D)$ and a tree decomposition of each F_i of width $\mathrm{tw}(F_i)$. Again, it is purely technical to check that if we merge these tree decompositions and add appropriate edges between their bags, we get a tree decomposition of G of width $\max\{\mathrm{tw}(D), z\}$, thus showing that $\mathrm{tw}(G) \leq \max\{\mathrm{tw}(D), z\}$. Knowing that $\mathrm{tw}(G) \geq \max\{\mathrm{tw}(D), z\}$ and that $\mathrm{tw}(G) \leq \max\{\mathrm{tw}(D), z\}$, we state that $\mathrm{tw}(G) = \max\{\mathrm{tw}(D), z\}$ and the theorem is proven. □

Theorem 17 carries several important practical implications.

To start, the proof of the theorem hints on a simple algorithm for computing a minimum DG-kernel of any graph G. Recall Definition 13 and Remark 15. From these statements, it is clear that replacing any F-subgraph of G with a single edge yields a DG-kernel of G. As the proof says, any F-subgraph F of G can be reduced to a single edge by removing its radial subnetworks and all but one lines, then gradually merging the remaining vertices, except $\mathrm{A}(F)$, with any vertex in $\mathrm{A}(F)$.

By exhaustively applying this procedure, we can derive a minimum DG-kernel of G. Algorithmically, this involves iteratively removing vertices of degree 1 and merging vertices of degree 2 with any of their neighbours until no such vertices remain, or until we obtain a graph with a single edge. This process necessitates at most $O(|\,\mathrm{V}(G)|^2)$ iterations.

Besides this, let us examine the formula proven in Theorem 17. When constructing a distribution grid, it is highly likely that some redundancy exists in the structure. Therefore, if the formula is applied to a distribution grid, the treewidth can generally be expected to depend predominantly on the treewidth of a DG-kernel. Recall Theorem 8 and in particular the set $\mathcal{F}_3$. We argue that it is improbable for buses in a distribution grid to be interconnected in more complex ways than suggested by the forbidden minors in this set. Thus, we hypothesise that the treewidth of distribution grids will be generally low and probably bounded by 4 for many cases.

In order to provide a first basis for our claim, we tested this hypothesis for MV distribution grids of 4 French cities[3] with population between 200,000 and 500,000 people. Each distribution grid was taken within the bounds defined by a major ring road in the corresponding metropolitan area. For each such grid G, where each vertex is a primary or a secondary substation, we constructed its minimum DG-kernel D^* and used Theorem 17 to compute the treewidth of G. Our results can be found in Table I. An example of a minimum DG-kernel for one of the studied grids can be found in Fig. 6.

TABLE I
TREEWIDTH OF FRENCH DISTRIBUTION GRIDS.

Metropolitan area	$\lvert\mathrm{V}(G)\rvert$	$\lvert\mathrm{V}(D^*)\rvert$	$\mathrm{tw}(G)$
Bordeaux	3305	21	3
Rennes	1406	53	4
Nantes	2307	77	4
Toulouse	1723	10	3

Table I shows that even big cities can have distribution grids with low treewidth. Furthermore, the size of D^* for all cases is notably smaller than that of the corresponding G (less than 4% of the original vertex count). Recall that the construction of D^* is polynomial-time feasible and, as per Theorem 4, determining treewidth is NP-complete. Therefore, calculation of $\mathrm{tw}(G)$ with a strategy involving constructing D^*, computing $\mathrm{tw}(D^*)$ and then applying Theorem 17 can yield substantial performance gains compared to direct computation of $\mathrm{tw}(G)$.

V. EXAMPLE

In this section, we show how tree decompositions can be used in practice to construct efficient dynamic programming algorithms for distribution grids with low treewidth. We also evaluate their computational performance in comparison to a mixed-integer linear programming (MILP) baseline.

A. Problem statement

The example problem we consider focuses on determining an optimal operating state for a MV distribution grid. For

[3]Data provided by Enedis: https://data.enedis.fr/pages/cartographie-des-reseaux-contenu/

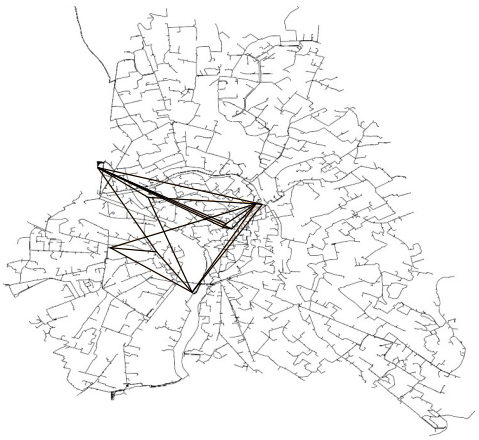

Fig. 6. The MV distribution grid of Toulouse (thin) with its minimum DG-kernel (thick).

simplicity, we assume that the distribution grid is represented by a graph denoted as G, which has a DG-kernel D' and an edge replacement sequence $(D' = G_1) \xrightarrow{e_1,F_1} G_2 \xrightarrow{e_2,F_2} \cdots G_{n-1} \xrightarrow{e_{n-1},F_{n-1}} (G_n = G)$ with the following properties: (P1) $\mathrm{V}(D')$ is the set of all primary substations of G, (P2) all e_i in the edge replacement sequence belong to $\mathrm{E}(D')$, (P3) all radial subnetworks of G are degenerate.

In this scenario, we aim (1) to select switches along the feeders whose opening divides the network into a set of trees, each tree containing exactly one primary substation. Given the active and reactive power injections at the secondary substations and the line characteristics, we also aim (2) to ensure that the voltage remains within the acceptable limits of 1 ± 0.1 pu at all locations. To this end, we assume that the supply transformers at all primary substations have 21 tap positions from $\{-10, \ldots, 10\}$, where each tap position $v(w) \in \{-10, \ldots, 10\}$ sets the base voltage of w to $1 + 0.01v(w)$ pu. To ensure the maximum reliability of the discovered operating state, we aim (3) to provide as many options as possible for adjusting each tap in either direction. Formally, we can define our objective as the minimisation of $\max_{w \in \mathrm{V}(D')} |v(w)|$.

To calculate voltage drops, we use the common LinDistFlow model [20]. Namely, if $s_1, s_2 \in \mathrm{V}(G)$ are adjacent substations in some grid G after the selected switches are opened, r, x are the resistance and reactance of the connection between them, f_a, f_r are the active and reactive power flows from s_1 to s_2, then given the voltage $u(s_1)$ we compute the voltage $u(s_2)$ as $u^2(s_2) = u^2(s_1) - r \cdot f_\mathrm{a} + x \cdot f_\mathrm{r}$.

For convenience, we will further address this problem as SWITCHSELECTION. Its summary can be found in Table II.

SWITCHSELECTION is a non-trivial, discrete optimisation problem, as illustrated for a minimal distribution grid in Fig. 7.

B. Dynamic programming algorithm

We approach SWITCHSELECTION using a classical method of dynamic programming with memoisation, which allows us to recover a solution (or enumerate all solutions) to the problem with the help of simple backtracking.

TABLE II
SWITCHSELECTION PROBLEM.

Input	• G – a MV distribution grid with DG-kernel D' that satisfies properties (P1), (P2) and (P3). • $p : \mathrm{V}(G) \setminus \mathrm{V}(D') \to \mathbb{Q}$ – active power at each secondary substation. • $q : \mathrm{V}(G) \setminus \mathrm{V}(D') \to \mathbb{Q}$ – reactive power at each secondary substation. • $r : \mathrm{E}(G) \to \mathbb{Q}$ – resistance of each edge. • $x : \mathrm{E}(G) \to \mathbb{Q}$ – reactance of each edge.
Output	• $v : \mathrm{V}(D') \to \{-10, \ldots, 10\}$ – an optimal tap position for each primary substation. • $S \subseteq \mathrm{E}(G)$ – set of edges where switches should be opened.

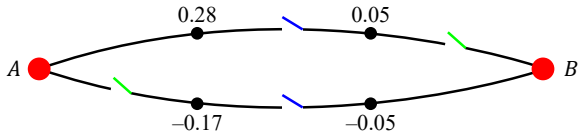

Fig. 7. Minimal example instance of the SWITCHSELECTION problem. Two primary substations (red circles) supply four secondary substations (black circles) with active power in-feeds shown alongside. For simplicity, reactive powers are set to 0 and all line segments are assumed to have a resistance of 1 pu. The trivial division of the feeders in the middle (blue switches) is infeasible. In contrast, the optimal split (green switches) requires tap positions $v(A) = 10$ and $v(B) = -10$.

Given the input grid G, our algorithm starts by constructing D' and generating a directed tree decomposition (T, X) for it. One can use any existing tree decomposition algorithm to generate an undirected tree decomposition as in Definition 1 and then select any of its vertices as the root.

After constructing (T, X), we start a bottom-up traversal of $\mathrm{V}(T)$ in a depth-first search post-ordering. For each $b \in \mathrm{V}(T)$, we use brute force to find locally feasible tap positions for all primary substations contained in X_b, i.e. we find all combinations of tap positions of X_b for which there is at least one edge on each feeder connected to any two primary substations from X_b where a switch can be opened to keep all voltages on the feeder within the limits of 1 ± 0.1 pu.

Each $b \in \mathrm{V}(T)$ stores a memo in which we record all locally feasible tap positions for b, and the minimum of the maximum value of $|v(w)|$ we have discovered so far for each set of locally feasible tap positions. We require that if b has a child b', then any locally feasible set of tap positions for b stored in its memo must have at least one corresponding locally feasible set of tap positions for b' stored in its memo, such that the tap positions for primary substations from $X_b \cap X_{b'}$ coincide.

We then backtrack the vertices of T in a breadth-first search pre-ordering and use the stored memos to construct an optimal solution to the problem.

C. Theoretical evaluation

Let us assess the complexity of the given approach. In terms of the time complexity, the algorithm has to construct DG-kernel D', which can be done in $O(|\mathrm{V}(G)|^2)$, then it has to build a tree decomposition of D', which can be done in $O(2^{\mathrm{tw}(G)^3} \cdot |\mathrm{V}(G)|)$ due to Theorem 4, and then it has to

traverse $O(|\,\mathrm{V}(G)|)$ bags of the tree decomposition, where each bag is processed in $O(21^{\mathrm{tw}(G)+1} \cdot |\,\mathrm{E}(G)|)$ time. Thus, the total time complexity is $O(21^{\mathrm{tw}(G)+1} \cdot |\,\mathrm{E}(G)| \cdot |\,\mathrm{V}(G)|)$, which is parameterised quadratic in the size of the graph. As for the space complexity, in addition to the input data, the algorithm has to store the tree decomposition and a memo for each bag. The tree decomposition has $O(|\,\mathrm{V}(G)|)$ bags, each containing at most $\mathrm{tw}(G)+1$ vertices. Each memo has at most $21^{\mathrm{tw}(G)+1}$ entries with at most $\mathrm{tw}(G)+1$ tap positions each. Thus, the algorithm requires $O(21^{\mathrm{tw}(G)+1} \cdot \mathrm{tw}(G) \cdot |\,\mathrm{V}(G)|)$ of additional space, which is parameterised linear.

These observations allow us to conclude that, at some point, this algorithm can be expected to outperform traditional combinatorial methods for solving the discrete optimisation problems, which are essentially exponential in their time complexity.

D. Computational evaluation

We now evaluate our proposed algorithm on generated MV distribution grids of treewidth 2, comparing the computation time and solution quality to a MILP.

1) Random test grid generation: Test grids of treewidth 2 are generated in two steps: (1) the generation of a DG-kernel and (2) the generation of feeders. DG-kernel D' with n vertices is constructed using the following procedure: a) Define $D' \leftarrow K_3$. b) Add a new vertex to D' and connect it with edges to any existing pair of adjacent vertices. c) Repeat step b) until $|\,\mathrm{V}(D')| = n$. d) Take any connected subgraph of D' containing at least one cycle. Graph D' produced with these steps is guaranteed to have treewidth 2 [15]. After constructing the DG-kernel, we replace each edge of it with an F-graph. The number of lines for each F-graph is chosen randomly from $\{2, \ldots, 5\}$. The number of secondary substations on each line is chosen randomly from $\{5, \ldots, 10\}$. Finally, each secondary substation and each edge of the resulting grid is randomly assigned corresponding power injections, each from $[-0.5, 0.5)$, and line characteristics, each from $[0.01, 0.05)$. All distributions are uniform.

2) Baseline & implementation: As a baseline, we formulate the SwitchSelection in the form of a MILP and solve it using CPLEX™ version 22.1.1 with the default configuration. The details of this are found together with the implementation of our algorithm in our public repository[4].

All random values are generated using the xoroshiro128++ pseudorandom number generator [21] with the seed value of 13374. Tree decompositions are constructed with the algorithm by Tamaki [22].

3) Execution environment: Our algorithm (hereafter referred to as TreeDecompositionSolver) and the baseline algorithm (hereafter referred to as CPLEXSolver) were both run on an Intel® Core™ i7-10750H CPU @ 2.60GHz with 6 physical and 12 logical cores. Both algorithms used a maximum of 12 threads during their execution.

[4]https://github.com/EINS-TUDa/PSCC2024-SwitchSelection

4) Results: The TreeDecompositionSolver yields the optimal switch locations for the instance in Fig. 7 and the same objective and feasibility value as CPLEXSolver for all the tested instances.

In Fig. 8, we examine the running time of both algorithms for increasing size of the graphs.

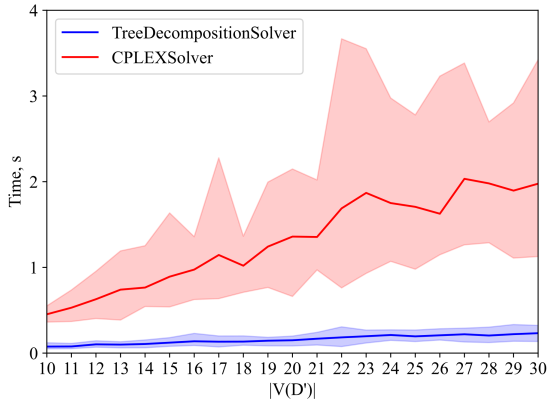

Fig. 8. The average running time in seconds over 20 random instances of SwitchSelection for each $|\,\mathrm{V}(D')|$. Transparent shapes show the intervals between the minimum and the maximum observed computation time.

The plot shows that for all test instances, the TreeDecompositionSolver had a better performance than the CPLEXSolver. However, it must be noted that the absolute differences are modest for the given size of distribution grids. For larger grids, the variance of the computation times for the CPLEXSolver visibly increases, whereas the variance for the TreeDecompositionSolver remains virtually constant. A linear fit of a power function $a|\,\mathrm{V}(D')|^b$ to the timing results of the TreeDecompositionSolver yields values $a = 0.008$ and $b = 0.991$. This indicates a linear growth on the given sample size, which is below the theoretical limit.

VI. Conclusion

In this work, we introduced the novel conceptual framework featuring F-graphs and DG-kernels, studied their properties and argued that they can be used to analyse the topology of distribution grids, in particular their treewidth and tree decompositions. We demonstrated that minimum DG-kernels can be built in polynomial time. Our hypothesis that the treewidth of distribution grids tends to be low lays the foundation for the development of efficient dynamic programming algorithms for many applications, an example of which was presented.

Looking ahead, it is pivotal to test our hypothesis on a broader spectrum of distribution grids. Moreover, we encourage further exploration into the application of tree decompositions and, more generally, dynamic programming to a broader array of combinatorial problems arising in power systems.

References

[1] Y. Hu, A. Kuh, T. Yang, and A. Kavcic, "A belief propagation based power distribution system state estimator," *IEEE Computational Intelligence Magazine*, vol. 6, no. 3, pp. 36–46, 2011.

[2] A. Minot, Y. M. Lu, and N. Li, "A distributed gauss-newton method for power system state estimation," *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3804–3815, 2015.
[3] W. Jiang, J. Chen, H. Tang, S. Cheng, Q. Hu, M. Cai, and S. Rahman, "A physical probabilistic network model for distribution network topology recognition using smart meter data," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6965–6973, 2019.
[4] L. Zdeborová, A. Decelle, and M. Chertkov, "Message passing for optimization and control of a power grid: Model of a distribution system with redundancy," *Physical Review E*, vol. 80, no. 4, p. 046112, 2009.
[5] E. Kellerer and F. Steinke, "Scalable economic dispatch for smart distribution networks," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 1739–1746, 2014.
[6] Y. Jiang, D. Kouzoupis, H. Yin, M. Diehl, and B. Houska, "Decentralized optimization over tree graphs," *Journal of Optimization Theory and Applications*, vol. 189, pp. 384–407, 2021.
[7] A. Engelmann, M. B. Bandeira, and T. Faulwasser, "Approximate dynamic programming with feasibility guarantees," *arXiv preprint arXiv:2306.06201*, 2023.
[8] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," in *IEEE Global Communications Conference*, 2013, pp. 2164–2169.
[9] A. Agarwal, "A novel polynomial time heuristic algorithm for minimal pmu allocation in the grid," in *Soft Computing: Theories and Applications*, R. Kumar, C. W. Ahn, T. K. Sharma, O. P. Verma, and A. Agarwal, Eds. Springer Nature Singapore, 2022, pp. 661–671.
[10] M. Cygan, F. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parametrized Algorithms*. Springer Cham, 2016.
[11] K. Atkins, J. Chen, V. Anil Kumar, and A. Marathe, "The structure of electrical networks: a graph theory based analysis," *International Journal of Critical Infrastructures*, vol. 5, no. 3, pp. 265–284, 2009.
[12] J. Valdes, R. E. Tarjan, and E. L. Lawler, "The recognition of series parallel digraphs," in *ACM Symposium on Theory of Computing*, 1979, pp. 1–12.
[13] S. Arnborg, J. Lagergren, and D. Seese, "Easy problems for tree-decomposable graphs," *Journal of Algorithms*, vol. 12, no. 2, pp. 308–340, 1991.
[14] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
[15] H. L. Bodlaender, "A partial k-arboretum of graphs with bounded treewidth," *Theoretical Computer Science*, vol. 209, no. 1, pp. 1–45, 1998.
[16] S. Arnborg, D. G. Corneil, and A. Proskurowski, "Complexity of finding embeddings in a k-tree," *SIAM J. Algebraic Discrete Methods*, vol. 8, no. 2, p. 277–284, apr 1987.
[17] H. L. Bodlaender, "A linear time algorithm for finding tree-decompositions of small treewidth," in *ACM Symposium on Theory of Computing*, 1993, pp. 226–234.
[18] D. P. Sanders, "Linear algorithms for graphs of tree-width at most four," Ph.D. dissertation, Georgia Institute of Technology, 1993.
[19] D. Wolter, "Neue Topologiekonzepte für moderne Mittelspannungsnetze," Ph.D. dissertation, Universitätsbibliothek Wuppertal, 2019.
[20] M. E. Baran and F. F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on Power Delivery*, vol. 4, no. 1, pp. 725–734, 1989.
[21] G. Marsaglia, "Xorshift rngs," *Journal of Statistical Software*, vol. 8, pp. 1–6, 2003.
[22] H. Tamaki, "Positive-instance driven dynamic programming for treewidth," *Journal of Combinatorial Optimization*, vol. 37, no. 4, pp. 1283–1311, 2019.