Automatic Single-Line Diagram Generation for LV Networks from GIS Data

Pablo Bermejo Villaescusa
University of Strathclyde – Universidad Pontificia Comillas ICAI
Glasgow, Scotland

Abstract— **Low-voltage (LV) distribution networks are being reshaped by high penetrations of distributed energy resources, turning formerly passive feeders into active, bidirectional systems. Utilities already maintain rich network models in geographic information systems (GIS), yet raw geospatial representations do not conveniently convey connectivity and topology at a glance. For this purpose, single-line diagrams (SLDs) remain the most intelligible tool, but manual drafting does not scale well to utility-sized datasets. This thesis addresses that gap by automating the generation of standard, orthogonal SLDs directly from heterogeneous GIS models. The thesis develops a reproducible, modular, end-to-end pipeline that, given a GIS-based model of an LV distribution grid section, generates an SLD of the underlying network through four phases: (1) graph classification and simplification, producing a parameter-controlled simplified network graph that preserves key characteristics; (2) normalization to enforce a standard node pattern compatible with diagram rendering; (3) layered layout via a custom implementation of the Sugiyama framework for layer assignment and crossing minimization; and (4) plotting using standard single line diagram symbols and exporting in SVG format. The implementation is focused on ScottishPower Energy Networks (SPEN) data but can be generalized to other network datasets through minor adjustments to the first phase. Eight case studies with real LV network sections (increasing in size and meshedness) are used to demonstrate the method. The algorithm produced valid diagrams with no symbol overlap for six networks; and two invalid diagrams caused by non-level-planar substructures produced by fixed layer assignment. Runtime was ~0.1–1.6 s for smaller, radial sections and ~5–21 s for larger, more complex networks. One medium-voltage network outside the main LV scope was also laid out successfully, illustrating the versatility of the method. Limitations and trade-offs are also discussed, which include: (i) fixed layer assignment improves simplicity but can induce non-level-planarity; (ii) heuristic crossing minimization scales well but is non-deterministic; and (iii) input parameters (e.g., consumer grouping) offer adjustable detail but with imperfect control over the results.**

Keywords— SLDs, LV networks, Grid observability, DERs, Power system graphs, Graph layout.

I. INTRODUCTION

LV distribution networks are experiencing a significant shift due to the growing penetration of distributed energy resources (DERs), including rooftop PV, distribution-level storage and electric vehicles, which are turning traditionally passive LV feeders into active bidirectional systems. The resulting behaviors (voltage instability, protection coordination challenges, switching network topology) call for necessary increased observability and the development of tools that help network operators and planning engineers to quickly understand network connectivity, topology, and system status information at a glance [1].

Distribution system operators (DSOs) typically maintain comprehensive network databases based on geographical information systems (GIS) which contain information about network assets and their location. However, due to their scale and vast amount of data, representations of this raw data do not intuitively provide easily graspable connectivity and topology information. Single line diagrams are better suited for representing network connectivity and its underlying topology in an easier to process way for operators. However, the traditional approach of manually drafting the single line diagrams is slow and therefore impractical for visualizing distribution networks, which may contain millions of elements. This creates both a need and opportunity to develop automatic methods to generate single line diagrams leveraging the large GIS databases maintained by DSOs.

There is currently no open, reproducible end-to-end workflow that converts heterogeneous GIS-based LV network models into standard, orthogonal, single-line diagrams at scale. Existing approaches are either partial or proprietary; academic work often targets either radial trees or generating general graph drawings not focusing on single line diagram conventions. Crossing minimization on layered, meshed graphs is challenging; under fixed layer rankings, non-level-planar substructures make zero-crossing layouts impossible without re-ranking.

Therefore, the goal of the thesis is to implement a pipeline that generates standardized orthogonal single line diagrams from GIS data, ensuring that the algorithm:

- Preserves topology and elements connectivity information (switches, fuses, transformers, etc.).
- Minimizes visual clutter, grouping consumers and displaying only essential topological information.
- Runs fast enough for interactive or near interactive use on average personal computing hardware.

II. BACKGROUND AND LITERATURE REVIEW

A. *Visualization of Power System Data*

Power grid visualization has undergone a significant transformation over the past decades, evolving from static, paper-based schematic drawings to dynamic, interactive digital platforms that integrate real-time data.

Contemporary grid observability architectures for LV/MV distribution systems rely on two complementary spatial frameworks: geographical (map-based) and topological (connectivity-based). Each of these approaches is more suitable for different applications [2].

Geographical reference: based on geographical information systems (GIS), system elements and their associated data are displayed anchored to their physical location. This approach is useful when the location of the information shown is relevant, and it can help pinpoint the geographical coordinates of a source of information, which is especially useful for example when determining fault locations in transmission lines, or supporting street-level maintenance planning, easing the process of dispatching field crews. However, it might struggle to represent information in an organized and structured manner, failing to convey information effectively when a significant zoom is necessary in areas like dense urban networks, where information might be too cluttered otherwise, while the opposite occurs in rural areas, where information might be excessively sparse [2].

Topological reference: focuses on displaying electrical connectivity rather than physical geographical distance. This helps to identify the relationships between different elements of the network, like separating different voltage levels, and how system dynamics in an area of the grid might spread to neighboring grid elements, or how a local power outage could spread to adjacent areas and what switches could potentially be operated to isolate it. Single-line diagrams remain the industry standard for this purpose, thanks to their ability to express bus-branch relationships in a compact format, scaling from substations to interconnections [2].

B. *Modelling LV Networks as Graphs*

Due to their inherent network structure, electric networks can be naturally modelled as graphs. In this mathematical abstraction, network elements (substations, buses, switches, capacitors, loads, etc.) are represented as nodes, while electrical connections (transmission or distribution lines, cables, transformers) are represented as edges. This representation provides a powerful and flexible framework for analyzing and simulating the behavior of large power systems, enabling the application of a wide range of graph algorithms.

- Nodes: representing buses, substations, transformers, switches, loads and other electrical elements.
- Edges: representing transmission lines, distribution feeders, cables and other electrical connections between components.

Applications of graph theory in power systems include topology processing to identify connectivity, energized sections and traces; optimization and power-flow formulations that operate on the graph structure; and state estimation and fault localization. Graph modelling provides numerous benefits, like the possibility of applying well studied graph algorithms and mathematical frameworks; but several challenges must also be considered, including loss of electrical detail in pure topology views, dynamic behavior not captured by static graphs, and the scalability vs detail trade-off, which motivates network simplification approaches where intelligent graph reduction preserves essential connectivity while improving visualization clarity [3].

C. *Automatic Schematic Generation of Single Line Diagrams*

Power system visualization through diagrams has undergone fundamental transformations from manual paper-based and hand drawn diagrams, to sophisticated digital solutions. Various approaches have been developed to address the complexity of modern electrical networks. Several topological and aesthetic constraints and objectives must be achieved in the generation of diagrams for LV/MV grids including minimizing edge crossings, avoiding component overlap and maintaining layout clarity.

Some early approaches include pioneering work presented by Canales-Ruiz et al. in 1979 [4], proposed an automatic drawing algorithm for single-line diagrams. Their approach formulated the problem as a layered graph ordering algorithm, in which every node represents a bus in the diagram, and the layer for each bus is assigned based on a longest path algorithm. The algorithm then takes an iterative greedy switch approach to swap the edges in each layer of the diagram until it finds a solution with no crossings. This underlying graph is then represented graphically by assigning strictly vertical lines to buses and horizontal lines to connections between buses.

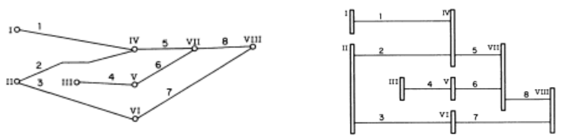

Figure 1: underlying graph (a), and diagram representation proposed by Ruiz et al. (right) [4]

Other approaches have built on this foundation, treating the diagram representation of power systems as a graph layout problem. Tree algorithms, like the Reingold Tilford algorithm, are used to represent graphs as rooted trees, useful if the underlying network is completely radial. A rooted tree layout was proposed by Rao et al. in 2003 [5], which generates a layered tree layout given a power system radial feeder, and then converts the layout into an orthogonal bus structure representation. Force directed and spring-based approaches model the graph as physical system in which nodes repel each other, and edges act as springs pulling nodes together. Although these approaches are intuitive and can handle diverse topologies, they do not generate the orthogonal and layered layouts often desired in one-line diagrams; nor do they yield geographically faithful representations. Mixed-Integer Linear Programming formulations have been proposed that guarantee optimal solutions for objectives such as crossing minimization or area minimization, but their computational complexity grows significantly with network size.

Although innovative solutions and useful applications have been developed over the past 4 decades, the field of automatic diagram generation is still largely undeveloped. Most contributions target very specific problems that cannot be generalized broadly to provide useful standardized single line diagrams. Some of the current obstacles include the lack of an end-to-end open-source tool chain, algorithmic bias towards radial or weakly meshed topologies, underdeveloped treatment of network simplification, and the lack of scalable heuristic implementations for meshed networks.

D. *Graph Layouts and Planarity*

A graph is planar if it can be laid out in a 2D plane such that no edges intersect each other except at their endpoints. There are several algorithms that can test for planarity, achieving linear time complexity, including the Hopcroft–Tarjan method or the Boyer–Myrvold method [5]. However, even though checking for planarity is simple, finding a layout for a planar graph with the minimal number of crossings is not, and there is not a definitive solution that works on every case. This is actually an NP-hard problem that is generally tried to be solved through heuristics.

Graph layout algorithms have the objective of assigning 2D positions to each node and routing edges so that the drawing conveys structure clearly. Common goals are to reduce visual clutter (minimizing edge crossings, edge bends and node overlap), emphasize structure (hierarchy, tree shape, flow direction, symmetry), satisfy aesthetic constraints (even spacing, alignment, balance), and represent data attributes. There is a vast amount of different layout algorithms, each serving their own specific purpose and best suited for a certain type of graph in particular, including hierarchical, tree, force-directed, circular, matrix-based or geographic. Two approaches in particular are relevant for this project: Reingold–Tilford (trees) and the Sugiyama framework (general directed graphs).

Reingold–Tilford produces tidy drawings of rooted trees, with even spacing, non-overlapping and aesthetically centered predecessors above their successors. It offers a computationally simple algorithm, which runs in linear time to the number of nodes and produces aesthetically pleasing results, but only works in acyclic rooted trees. The algorithm aims to satisfy: no node overlaps; children drawn on a level below their parent; parent centered over its children's span; uniform subtree separation; and symmetry for equivalent subtrees [6].

The Sugiyama framework allows hierarchical layouts using a 4-step framework: cycle removal, layer assignment, crossing minimization and coordinate assignment. Most layer generation algorithms generally do not incorporate the constraint that the resulting layout must be level-planar; generating a level-planar set of layers is an NP-hard problem, and detecting if a given graph is level-planar is not as trivial as detecting general planarity. Dummy node insertion helps by ensuring all edges span exactly one layer. For crossing minimization inside fixed layers, exact minimization is NP-complete, so heuristic methods are used—commonly the barycenter or median methods, with multiple top-down/bottom-up sweeps. A way to improve the results is to introduce naïve randomization of input orders, run the algorithm several times and keep the best solution; this increases the odds of escaping local minima at the cost of higher runtime. If the layered graph is non–level-planar, the crossing reduction cannot reach zero crossings; effectiveness is capped by the initial layering [7], [8].

III. Methodology

A. Formal Problem Definition And Objectives

The automatic generation of single-line diagrams from GIS data is formally defined as a graph transformation and visualization problem. The input network is defined as a spatial network graph $G = (V, E, P, A)$ where:

- $V = \{v_1, v_2, \dots, v_n\}$ is the set of nodes representing network components.
- E is a set of edges representing electrical connections between elements.
- $P: V \rightarrow \mathbb{R}^2$ is a position function mapping each node to geographic coordinates.
- $A: V$ is an attribute function mapping each node to a set of properties. Among these properties, is the component type, type_i. The set of all entity types in a graph is defined as T.

The output is a diagram representation $S = (G', L, R)$, where:

- $G' = (V', E')$ is a simplified directed acyclic graph derived from G.
- $L: V' \rightarrow \mathbb{Z}^2$ is a layout function mapping each node to a discrete grid position.
- $R: V' \rightarrow S$ is a rendering function mapping nodes to diagram symbols

The input graph is read from a GIS format data structure, which contains a set of nodes $\{v_i\}$ representing network elements and components with properties including a unique identifier: id_i, a component type: type_i, and other electrical properties: props_i; and a set of edges $\{e_j\}$, with properties source node: u_j and target node: v_j.

The algorithm also takes two additional inputs, which allow modifying the characteristics of the output diagram, and adjusting its resolution: *max_consumers_per_bus,* used to determine the maximum number of consumers inside each bus,which determines the vertical detail in the diagram by increasing or decreasing the number of buses, , based on the maximum number of consumers for each bus, lower values of *max_consumers_per_bus,* allow displaying more detail of the network's topology by creating more buses, and vice versa. The other parameter is max_consumers_per_group, which determines the maximum group size inside each bus, allowing to adjust granularity horizontally.

Internally, two additional inputs are also configured, *key_nodes,* which is used to identify important elements of the network that must be maintained in the final schematic, and *root_label*, which is used to identify the elements that must be placed on the first layer of the schematic, generally distribution transformers.

The algorithm produces an SVG diagram which meets the following objectives and constraints:

- Topology preservation, the connectivity for every element in the simplified graph reflects the connectivity in the original network.
- Hierarchical structure, the output is a layered representation with the distribution transformer(s) (or other configurable root element) placed on the first layer, and the rest of the elements in subsequent layers downstream.
- Key component preservation, all components identified as crucial are flagged and represented in the final schematic.
- Orthogonality: all connections are horizontal or vertical.
- Crossing-free: no elements overlap or cross with each other.
- Simplified representation, the algorithm groups consumers accordingly with the input parameters, and represents a simplified topological view of the network.

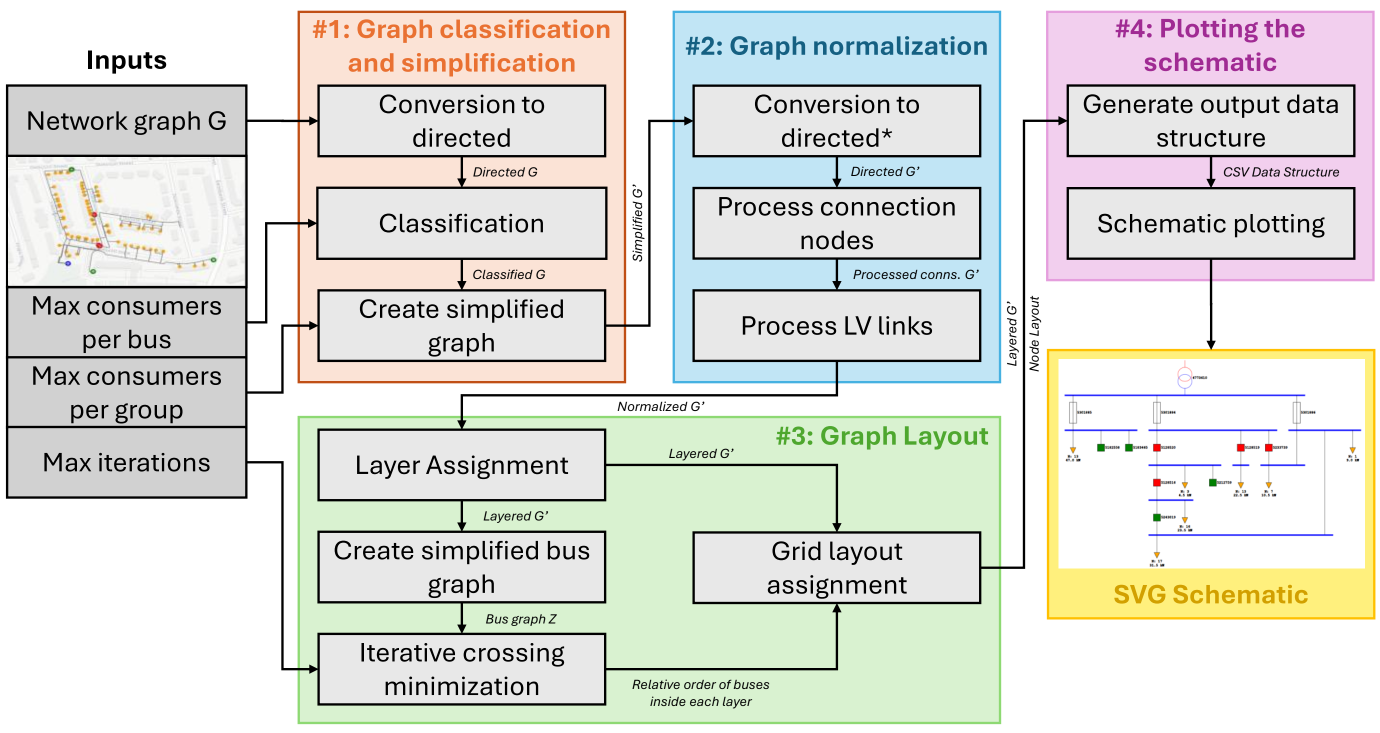

Figure 2: Methodology diagram.

B. *Solution Overview*

The solution takes a 4 step process to generate the final diagram representation from the raw GIS data, the detailed process can be observed in Figure 2.

This approach allows to ensure that the objectives are met by targeting them sequentially, and enables the replacement and improvement of individual modules while maintaining an end-to-end functional pipeline. It also improves compatibility, by maintaining the core algorithmic process intact, but making slight modifications to the input and output modules, as desired. The individual stages of the process are explained in the subsequent sections.

C. *Graph Classification and Simplification*

Inputs: raw graph G and parameters max_consumers_per_bus, max_consumers_per_group; internal sets *key_labels* and *root_label*. Output: classified and simplified graph G'.

The goal of the classification and simplification phase is to create a simplified subgraph G′ from the input graph G, reducing the number of nodes and grouping consumer nodes so that the resulting single line diagram representation contains only the key topological information of the underlying network. This process allows to reduce visual clutter, while also ensuring that the key connectivity information between all the elements in the network is preserved.

Converting the graph to directed: the algorithm takes the root nodes (distribution transformers in LV grids) and traverses the whole graph through breadth first search (BFS), converting edges to directed in traversal order. This generates a directed G to which the classification rules can be applied.

Node classification: the algorithm assigns each node a role that determines how it is simplified and laid out. Categories include consumer nodes; bus nodes (electrically equivalent points grouped later); connection nodes (boundaries for electrically equivalent zones and preservation of main joints); and key nodes (transformers, switches, fuses, LV links) that must be preserved. Whether adjacent nodes form the same bus is determined by *consumer_count*, the number of consumers a set of adjacent nodes are connected to, and the *max_consumers_per_bus* threshold.

Creating the simplified graph: connection and key nodes are added directly; adjacent bus nodes are grouped into a single node and reconnected to their original boundary connection nodes; consumer nodes adjacent to grouped buses are grouped in sizes up to *max_consumers_per_group* (and consumers directly attached to connection nodes are grouped and attached accordingly). *max_consumers_per_group* and *max_consumers_per_bus* together tune the level of detail displayed in the final diagram. Edges in G′ are created as undirected for efficiency; direction is restored later after simplification.

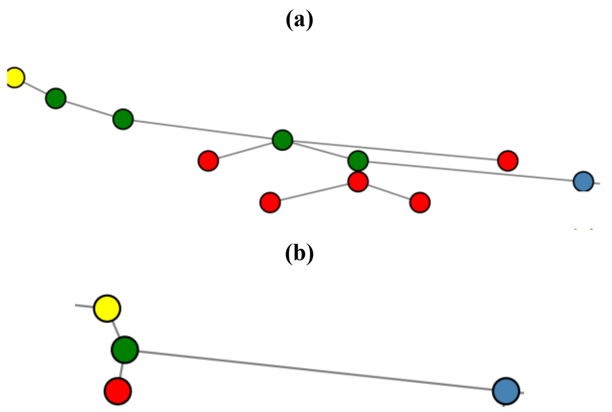

Figure 3: Sample graph section before (a) and after (b) the simplification algorithm is applied. Nodes: green-bus, red-consumer, blue-connection, yellow-connection key node.

D. Graph normalization

Input: simplified graph G'. Output: normalized graph G' with bus-connection-bus structure.

The objective of this phase is to ensure the simplified graph follows the normalized bus-connection-bus node structure that is required by the layout algorithm and to plot the final diagram. The post-processing transforms the undirected simplified graph into a DAG that meets:

- Connection nodes have only one predecessor and one successor, both bus nodes.
- There are no bus nodes adjacent to each other.
- Demand and key nodes are successors to bus nodes; connection-key nodes have one predecessor bus and one successor bus.

To maintain topology while achieving this structure, adjacent connection nodes are removed, and connection nodes with n successors are split into n copies with one successor and one predecessor each. Additional synthetic bus nodes are inserted in two scenarios: (i) adjacent key nodes; and (ii) connection nodes with a consumer or single-degree key successor. Single-degree bus nodes created as artifacts are removed, along with any non-key connection nodes linking them.

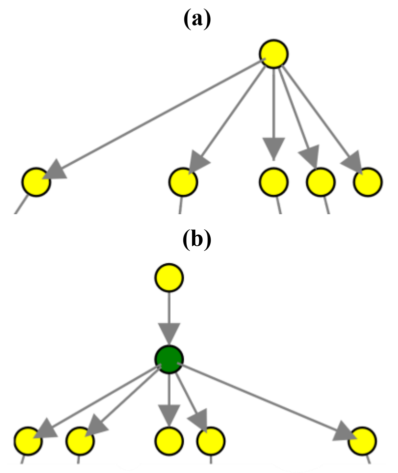

Figure 4: Insertion of bus nodes in the normalization phase, adjacent key nodes (a), bus node inserted (b)

E. Graph Layout Generation

Inputs: normalized graph G', max_iterations. Output: G' with assigned layers and coordinates.

Laying out the graph nodes is an essential step in producing a topological representation of the network. Generating a visually pleasing orthogonal single-line diagram requires no crossings in the underlying graph; and discrete x and y positions to allow orthogonality, and uniform spacing between elements. Positions are therefore assigned on a grid with y-layers based on longest distance from roots (incrementing only for bus nodes) and x-positions chosen to avoid crossings.

Creating a layout that minimizes crossings is NP-hard; although all graphs here are planar, fixed layer assignment can yield non-level-planar cases where a 0-crossing layout is impossible. The layout algorithm is a custom implementation of the Sugiyama framework with: layer assignment; crossing minimization; and coordinate assignment. The algorithm runs on a subgraph Z containing only bus nodes to improve performance, then propagates coordinates to all nodes. Edges are not rendered; connectivity is conveyed implicitly by bus length in x (to farthest successor) and by placing successors along the same y as their parent bus, using the node position as the top-most reference point to draw the symbol. Elements connecting two buses and connection nodes are represented vertically, with a length that is calculated based on the layers of the two buses they connect.

Crossing minimization uses a barycenter heuristic with dummy nodes inserted in multi-layer edges, ensuring every edge connects a node from one layer to a node in the following layer. Because a single run is prone to get stuck in local minima, the algorithm is run iteratively with randomized input edge orders up to a maximum number of runs; it stops early on a zero-crossings solution and otherwise returns the best-found (least crossings) layout.

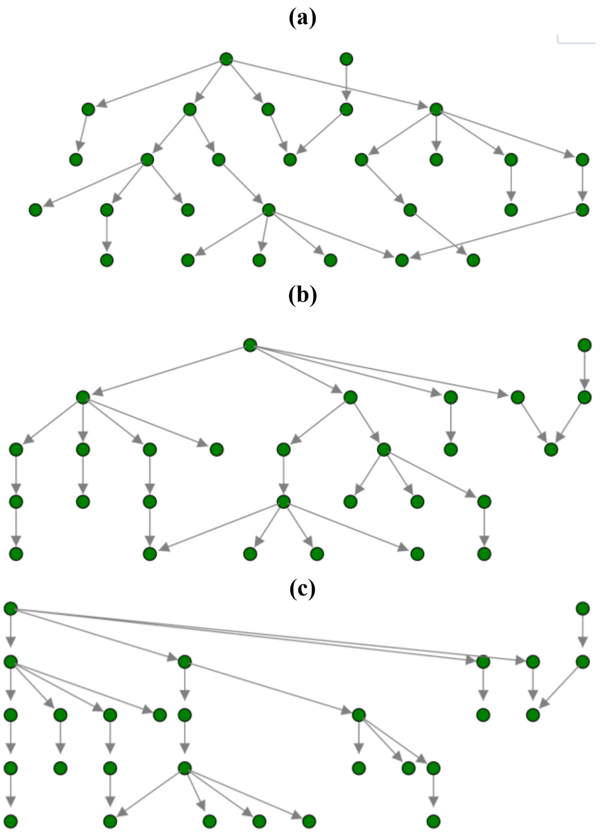

Figure 5: Example bus graph before (a) and after applying the crossing minimization algorithm (b); and after the assignment of discrete x positions (c)

F. Schematic Plotting and Output

Once node positions are determined, the size to plot buses and connection nodes is calculated. A plotting script assigns custom symbols to each node and generates a self-contained SVG. The SVG format is chosen for versatility, scalability, and fidelity to the geometric structure of the data, enabling efficient storage and resolution-independent scaling.

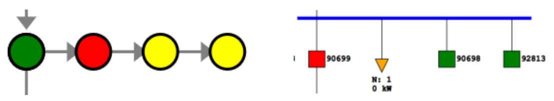

Figure 6: Symbol assignment for a bus with 4 elements, two open LV links (yellow nodes), a consumer (red node) and a closed LV link (not visible, below the green, bus node)

IV. Case studies

A. *Experimental setup*

To validate the results of the automatic diagram generation algorithm, a case study approach is used, in which several individual samples of LV networks of a wide variety in size and structure are represented as diagrams using the explained methodology, measuring several metrics to gauge performance and determine validity. Eight case studies are performed, with eight different networks with characteristics shown in Table 1. Their characteristics are compared by their size (number of nodes) and meshedness (measured by M, number of nodes with two predecessors in the underlying graph). A diagram is considered valid if the laid-out graph contains no crossings.

Table 1: Case studies input networks characteristics

Case Study	N_{NODES}	M
#1	69	0
#2	145	1
#3	239	2
#4	611	2
#5	1126	5
#6	1233	4
#7	2282	11
#8	1810	7

The objective of the first three case studies, which are performed on simpler networks, is validating if the resulting schematics faithfully represent the actual topology of the network, and to test whether the input parameters can be used to effectively adjust the characteristics and detail of the diagram as desired. On the other hand, case studies 4 to 7 focus on testing the performance of the layout algorithm on more complex networks. Case study 8 is performed on an MV network, outside of the formal scope of the project, but that allows to test the versatility of the algorithm, to generate schematics for data with different characteristics compared to the main project's scope.Figure 8 contains a representation of four of the networks used in the case studies. Orange points represent consumers, blue circles transformers and red/green circles closed/open LV links.

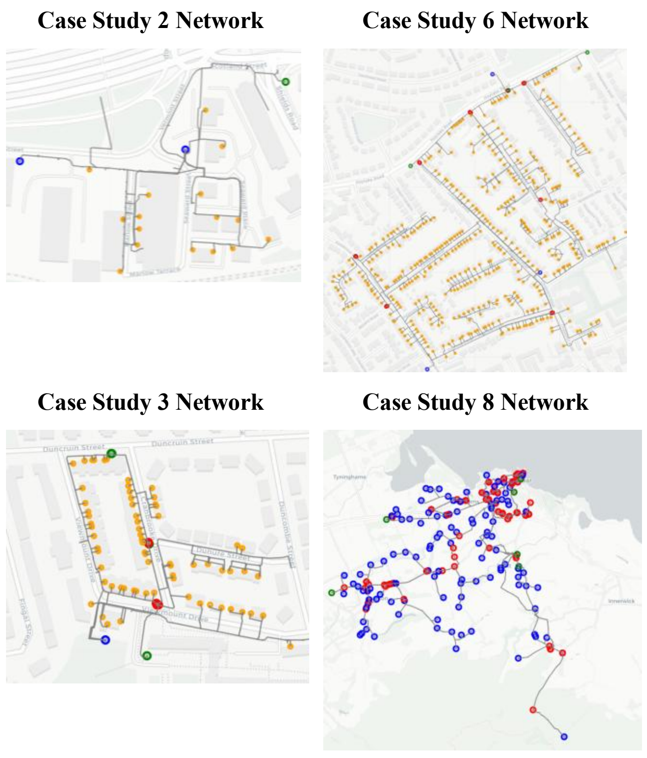

Figure 8: Sample of case study input networks

B. *Outputs for the case studies*

Figure 7 contains the single line diagrams generated for the four networks shown in Figure 8. The generated diagrams scale in complexity along the size of the input networks, and for the MV network in case study 8, the complexity of the diagram remains similar to the pre-simplification network, as

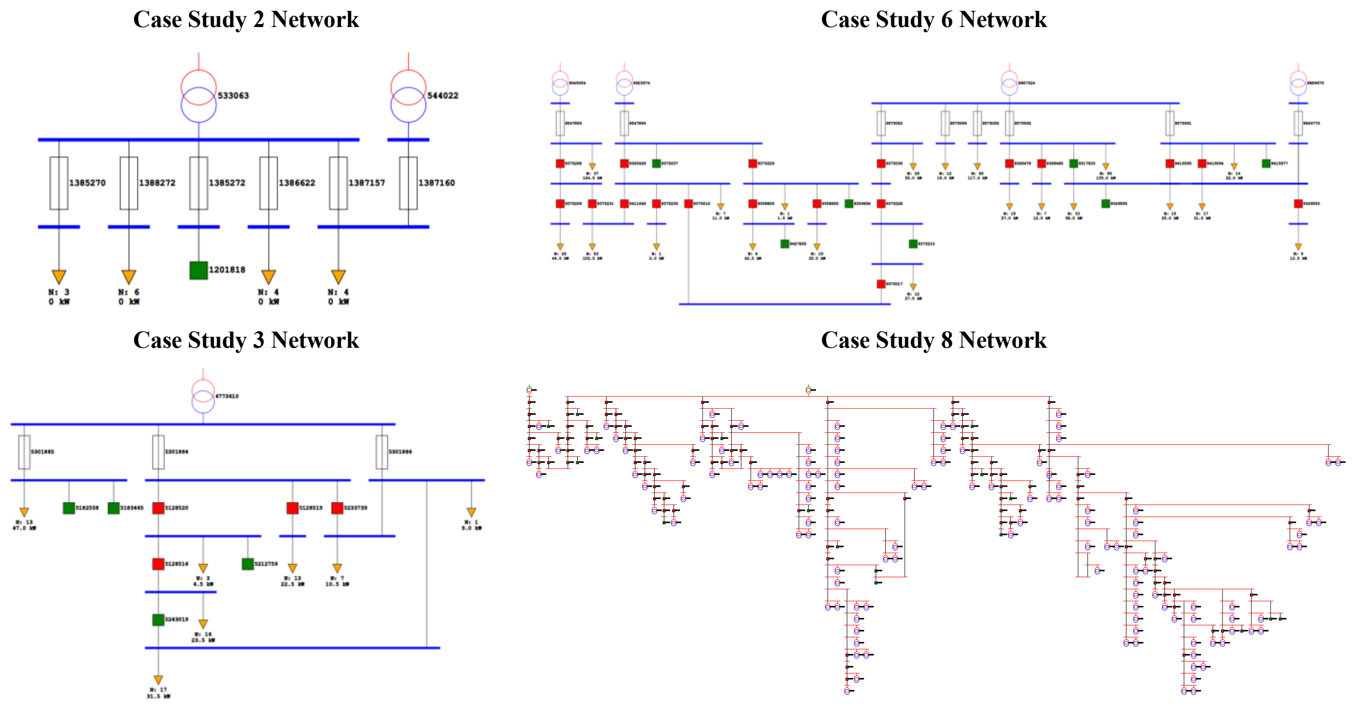

Figure 7: output diagrams for the networks shown in Figure 8

no consumer clusters were generated due to the network having none.

V. Results

A. Summary of Results and Key Findings

Table 2 contains a summary of each case studies' results, along with the characteristics of the input networks.

Table 2: Summary of results

Case Study	Results			
	k	**N_{iter}**	**N_L**	**Runtime (s)**
#1	0	1	13	0.1
#2	0	1	15	0.13
#3	0	1	36	0.58
#4	0	1	67	1.57
#5	0	4	64	5.07
#6	2	20	90	6.43
#7	8	77	127	21.18
#8	0	27	643	17.87

The algorithm generated valid diagrams for 6 of the 8 networks studied, as it was able to find zero crossing layouts for their underlying graphs. For case studies 6 and 7 the algorithm was not able to find a zero crossings solution for their underlying graphs and therefore generated invalid diagrams. Runtime increased significantly for larger and more meshed graphs.

The obtained results validate the following key findings:

Valid diagrams for most cases, failures align with expectations. The first 4 case studies produced a successful layout with zero crossings on the first iteration, reflecting the algorithm's effectiveness in small-sized, low-meshed networks. For case study 5, a significantly more meshed network, the algorithm also produced a valid diagram, although it required several layout iterations. The algorithm was unable to generate a zero-crossings solution for case studies 6 and 7, and therefore, a valid final diagram. However, this aligns with the expected result, as both networks have an underlying non-level planar graph which is generated by the fixed layer assignment process.

Runtime scales with size and meshedness. Runtime increased significantly in relation to the number of nodes in the graph, remaining almost linear for less complex networks (only ~0.1 seconds for the simplest cases), but becoming exponential as the network complexity, both in size and meshedness, increased, up to 21.18 s for the most complex network. This is caused by the longer graph traversals required for several algorithms, as well as the larger number of iterations of the layout algorithms. A smaller number of traversals (by for example integrating several algorithms together) could potentially lower the runtime for larger networks.

Input parameter effects are predictable and controllable. Consumer grouping parameters tune horizontal and vertical detail as intended in the final diagram. Decreasing max_consumers_per_bus increases vertical resolution, showing more detail into the network's topology by displaying more buses, while decreasing max_consumers_per_group allows to show more horizontal grouping granularity, showing higher numbers of consumer groups connected at the same network level.

MV network case study validates generality. The algorithm was also successful in producing a valid diagram for a MV network, out of the formal project's scope, demonstrating the proposed pipeline is generalizable to successfully generate valid diagrams for networks with significantly different topological characteristics than the LV networks on which it is mainly implemented.

Randomized naïve iterations of the layout algorithm improve results. Re-running the crossing minimization section of the layout algorithm with randomized inputs allowed the algorithm to escape local minima and find optimal solutions for case studies 5 and 8, whose solution was not found on the first algorithm iteration. For case studies 6 and 7, even though it could not find a solution with no crossings due to the non-level-planarity of the graphs, later iterations also successfully reduced the number of crossings. This performance is further analyzed in the following section.

B. Layout Algorithm Benchmark

To visualize and benchmark the performance of the layout algorithm, a simulation for each of the three most demanding case studies performed has been conducted, in which the layout algorithm exclusively was run 30 times with different values for the number of random naïve iterations, to test how increasing the maximum number of iterations can lead the algorithm to find more optimal solutions, and how the execution time increases in relation to the number of iterations.

Each test was conducted using incremental values of maximum iterations, for the networks in case studies 6, 7 and 8. The results can be observed in Figure 9:

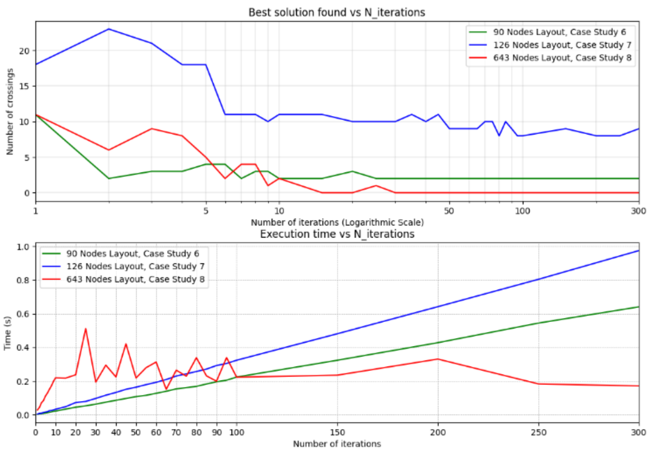

Figure 9: Layout algorithm performance, case studies 6, 7 and 8

As the figures illustrate, the tests for each network produced significantly different results. Starting with the network used in case study 6, the smallest out of the ones tested, the algorithm found the best solution (2 crossings) in only two iterations. However, due to the random nature of the algorithm, in the following tests it did not converge to that optimal solution until the 7th test, when 7 max iterations were used. As expected, for higher iteration tests, the algorithm always managed to find the best solution, which is the corresponding to two crossings, as explained in the case study. Execution time increased linearly, with the number of iterations, as expected, but was always shorter than 1 second, indicating that the layout optimization phase represents only a small fraction of total time to run the algorithm, which in case of case study 6 was 6.42 seconds for 100 max iterations.

In contrast, the results for the tests done on the larger and more complex case study 7 network, showed that the algorithm required a significantly larger number of max iterations to find the best solution it found among all tests (8 crossings), around 80. It also shows a greater variance in the results for higher max iteration values, which highlights that the randomness of the algorithm is correlated with the complexity of the network, suggesting that a higher number of maximum iterations should be used to maximize the odds of obtaining the optimal solution. Runtime also scaled linearly, with similar values to the case study 6 network, but a higher slope which is likely caused by the increased complexity and size of the network.

Finally, the results for the MV network used in case study 8 show significant differences, as in this case the algorithm did find a zero crossings solution. This crossing-free solution was first found in the 15 max iterations test, and the algorithm managed to consistently find it when the number of max iterations was higher than 30. This proves that when the underlying network contains no non-level-planar substructures, the algorithm can effectively find the optimal solution within a small number of iterations. The execution time figure shows a significant variability after 10 max iterations because the algorithm stops after finding the 0 crossings solution, leading to random total execution times for runs that are able to find that solution, and therefore uncorrelated with the number of max iterations. For tests that did not converge to the optimal solution (max iterations ≤ 10 and $= 25$), the algorithm did have a runtime proportional to max iterations, and its slope can be observed to be significantly higher than in the two previous simulations, further proving that the slope is correlated with the size of the network laid out.

C. Method Limitations and Design Trade-offs

The results of the case studies also highlight the following limitations and trade-offs

Fixed layer assignment vs level planarity. The algorithm assigns node layers based solely on distance from the roots, before crossing minimization. This fixed layer assignment provides simplicity and creates intuitive layers, that allow to easily represent topological distance from the transformer. This however causes a significant limitation, that the creation of non-level-planar structures is not avoided. An algorithm that dynamically checks level planarity and adjusts the layers to ensure it would be necessary to solve this, but at the cost of increased complexity and without certain success, as it is another NP-hard problem.

Naïve heuristic approach vs deterministic solution. The success of the crossing minimization algorithm relies on the naïve heuristic algorithm being able to escape local optima to find the optimal solution, which is increasingly unlikely as network size increases and requires more iterations. The approach is effective for simple networks, but more complex cases might benefit from a more deterministic approach, through MILP, for example, although this would likely increase runtime significantly.

Limited parametric tuning. The input parameters allow to adjust the maximum number of consumers in each bus and group, up to a certain degree. The number of consumers on each bus in some cases can be larger than the specified threshold due to several consumers being connected at the same topological point, or due to the graph simplification and normalization process, limiting the capability of tuning the final diagram through changing the input parameters. Additionally, when several groups of consumers are created in the same bus node, the groups generally do not reflect clusters consumers close to each other, which could be implemented through a more complex algorithm that takes this into consideration when creating clusters.

Exponential time complexity for complex networks. Although for the majority of cases studied, the algorithm was able to generate the single line diagrams in reasonably short times (less than 5 seconds), it has been shown that for more complex networks runtime increases exponentially. This might make the algorithm unsuitable for increasingly meshed and complex graphs.

VI. Conclusions and Future Work

This thesis demonstrates and delivers a reproducible pipeline that can be used to translate LV network GIS data into standard, readable, single-line diagrams that faithfully represent the underlying topology of the network. The separation of graph simplification, layout and symbol plotting enables a modular approach that can be tuned via several inputs to fit the data characteristics and to adjust characteristics of the final diagram, which contains a simplified view of the underlying network topology and key characteristics. The method produced six valid diagrams for eight diverse real networks studied, with the two unsuccessful cases explained by level-planarity violations caused by the fixed layer assignment algorithm. The runtime for these cases increases linearly with the number of nodes for the least complex networks but rises exponentially for larger and more meshed networks, although remaining on a timescale suitable for interactive or near interactive use on personal computing hardware.

Some suggested areas for further development and potential result improvement include:

- Substitute the fixed layer assignment and heuristic crossing minimization for a MILP approach that aims to solve a two-objective problem, jointly optimizing the layer assignment and relative position of nodes inside each layer to reliably produce valid zero-crossings layouts, not constrained by fixed layer assignments which might be non-level-planar. However, this approach, if feasible, might have an excessive time complexity that could make it impractical.

- Heuristic approaches to adjust the layer assignment. Detection of level planarity is complex, however, iteratively detecting smaller non-level planar structures inside a larger graph might be feasible, and changing the layer of some nodes in the structure while checking level planarity iteratively might be a valid heuristic approach for creating level planar graphs that can be laid out using the current crossing minimization and layout algorithms.

- Pre-computing layouts for different values of max_consumers_per_bus. As changing the value of max consumers can change the resolution into network details, effectively changing the number of bus nodes in some network sections, this can potentially cause non-level planar structures to disappear as the layers change. Although this is not an optimal solution as it removes the capability of tuning the input parameters, it might allow to generate single line diagrams of some networks that could not be generated using a different max_consumers_per_bus parameter. Pre-computing the layouts with several parameter values could allow storing what values produce suitable diagrams.

- Interactive final diagram. From the final layout, an interactive implementation of the diagram could be created, through a web-based application for example, that allows features like selecting elements to view more detailed information, selecting two points to create traces between elements in the network, or visualizing with more detail certain consumer clusters that the operator selects, for example.

- Partitioning the layout. For larger networks for which it might not be feasible to generate a single line diagram without components overlapping (due to complexity and non-level-planarity, a possible solution could be to compute several smaller layouts independently and later generating the complete layout by unifying the smaller sections.

VII. References

[1] S. Gordon, C. McGarry, and K. Bell, “The growth of distributed generation and associated challenges: A Great Britain case study,” *IET Renewable Power Generation*, vol. 16, no. 9, pp. 1827–1840, Jul. 2022, doi: 10.1049/rpg2.12416.

[2] T. J. Overbye and J. D. Weber, “Visualization of Power System Data,” 2000.

[3] M. Zhou, J. Yan, and Q. Wu, “Graph Computing and Its Application in Power Grid Analysis,” *CSEE Journal of Power and Energy Systems*, vol. 8, no. 6, pp. 1550–1557, Nov. 2022, doi: 10.17775/CSEEJPES.2021.00430.

[4] D. Toral and G. A. Alonso-Concheiro, “OPTIMAL AUITI4TIC DRAWING OF ONE-LINE DIAGRAMS Canales-Ruiz, fMrber ITEE.”

[5] Robin J. Wilson, *Introduction to Graph Theory*, Fourth edition. Longman Group Ltd, 1972.

[6] E. M. Reingold and J. S. Tilford, “Tidier Drawings of Trees,” 1981.

[7] KOZO SUGIYAMA, SHOJIRO TAGAWA, and MITSUHIKO TODA, “Methods for Visual Understanding of Hierarchical System Structures,” 1981.

[8] G. Brückner and I. Rutter, “Partial and constrained level planarity,” *Theor Comput Sci*, vol. 1045, p. 115291, Aug. 2025, doi: 10.1016/J.TCS.2025.115291.