

Applications of Graph Theory and Minimum Spanning Trees in Architectural Wiring Diagram Design

Nathanael Shane Bennet – NIM 13524119

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

E-mail: shane150806@gmail.com , 13524119@std.stei.itb.ac.id

Abstract—Architectural wiring diagrams are pictures that show the approximate locations and interconnections of receptacles, lighting, and permanent electrical services in a building. Optimizing the layout of a wiring diagram can save time and money involved in the installation of said wiring. This paper examines the implementation of Prim's algorithm to find the minimum spanning tree for a weighted graph and investigates the use of restrictions to adjust the graph based on real-life situations.

Keywords—graph theory, minimum spanning tree, optimization, wiring diagram, tree, Prim's algorithm

I. INTRODUCTION

An architectural wiring diagram serves as a comprehensive visual representation or blueprint that details the entire electrical wiring system within a building or a specific room. Unlike schematic diagrams, which focus on the theoretical operation of circuits, architectural wiring diagrams emphasize the physical layout, showing the approximate locations and interconnections of electrical components such as receptacles, lighting fixtures, switches, circuit breakers, and permanent electrical services. These diagrams use standardized symbols to represent different types of devices, ensuring clarity and consistency for electricians, engineers, and inspectors involved in the design, installation, and maintenance processes.

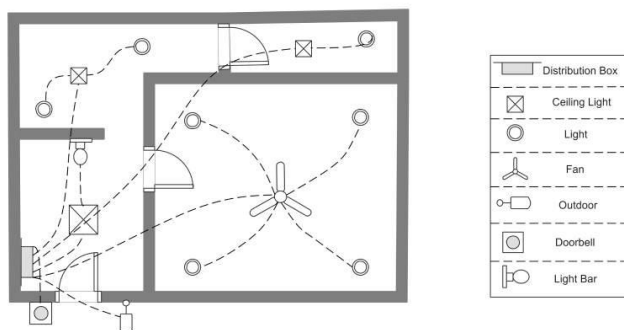


Fig.1. An example of an architectural wiring diagram. (Source: <https://www.edrawmax.com/house-wiring-diagram/>)

Wiring diagrams are indispensable tools in both the planning and execution phases of electrical installation. They not only guide the initial installation by illustrating how each component should be connected, but also serve as essential references for troubleshooting, upgrades, and repairs throughout the building's lifecycle. By providing a clear map of the wiring routes and device placements, these diagrams help ensure that electrical power is distributed efficiently and safely to operate a wide range of devices and appliances. Furthermore, accurate wiring diagrams are often required by regulatory authorities to verify compliance with safety standards and approve connections to the public electrical supply system.

Optimizing the design process of a wiring diagram can significantly reduce both the time and materials required for installation, while also minimizing the risk of workplace injuries associated with inefficient layouts or excessive wiring. One effective method for achieving such optimization is the application of graph theory, specifically by using a minimum spanning tree (MST) to visualize the shortest and most cost-effective set of paths between electrical fixtures. By minimizing the total length of wiring needed, designers can reduce material costs and simplify installation procedures.

This paper explores the implementation of Prim's algorithm in the context of architectural wiring diagram design. The study investigates how Prim's algorithm can be used to identify the optimal wiring paths that connect all necessary fixtures with the least total wire length. Additionally, the paper examines the impact of applying algebraic restrictions to the underlying graph, analyzing how these constraints affect the resulting network topology and overall efficiency.

II. THEORETICAL FRAMEWORK

A. Graph Theory

Graph theory is a branch of mathematics dedicated to studying graphs, which are abstract structures used to model

pairwise relationships between entities or objects. In this context, a graph is formally defined as a collection of vertices (also called nodes or points) and edges (also referred to as arcs, links, or lines) that connect pairs of vertices. The edges can be either undirected (signifying a two-way relationship) or directed, signifying a one-way relationship from one vertex to another.

The origins of graph theory date back to 1735, when Leonhard Euler solved the famous Königsberg bridge problem, laying the foundation for the field. Since then, graph theory has become a fundamental area within discrete mathematics, with wide-ranging applications in computer science, engineering, social sciences, biology, and more. For example, graphs are used to model communication networks, transportation systems, social networks, and molecular structures.

Graphs can be further classified based on their properties. For instance, a graph is called Eulerian if it contains a circuit that traverses every edge exactly once, and weighted if each edge is assigned a numerical value, which is particularly useful in optimization problems such as finding the shortest path or minimum spanning tree. The study of graphs also includes concepts such as graph coloring, connectivity, and subgraphs, each providing tools for analyzing complex relational data.

B. Tree

Within graph theory, a tree is a special type of graph characterized by being connected and acyclic—that is, there is a path between every pair of vertices, and no cycles exist within the structure. More formally, a tree with n vertices always has exactly $n-1$ edges, and removing any edge from a tree would disconnect the graph. Another equivalent definition states that for any two vertices in a tree, there exists exactly one unique path connecting them.

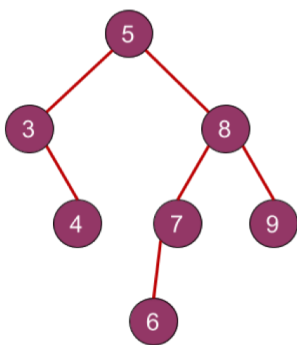


Fig. 2. An example of a tree. (Source: <https://www.shmoop.com/computer-science/graphs/trees.html>)

Trees are fundamental in both theoretical and applied contexts due to their strong structural properties. In computer science, trees serve as essential data structures for organizing and storing information efficiently, such as in file systems, database indexing, and hierarchical data representation. Additionally, trees are used in algorithms for searching, sorting, and parsing, as well as in network design and data compression techniques like Huffman coding.

A related concept is the forest, which is a disjoint union of trees—essentially, a graph with no cycles that may not be connected. In a tree, a vertex of degree one is called a leaf, representing an endpoint in the structure.

C. Minimum Spanning Tree

A minimum spanning tree (MST) is the subgraph of an edge-weighted graph that connects every vertex within the graph, is a tree (contains no cycles), and has the minimum combined edge-weight of all subgraphs that fulfill the previous two conditions.

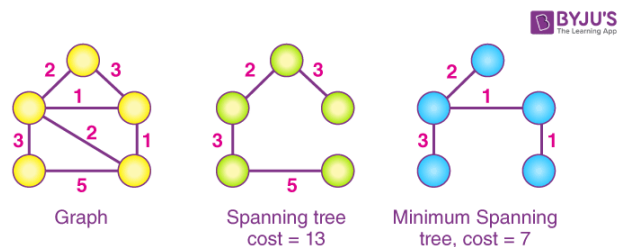


Fig. 3. An example of a minimum spanning tree. (Source: <https://byjus.com/gate/minimum-spanning-tree-notes/>)

Minimum spanning trees play a crucial role in network design and optimization. They are widely used in constructing efficient communication, transportation, and utility networks, ensuring that all points are connected with the least total cost. Applications include designing computer and telephone networks, road systems, and electrical grids, where minimizing the overall connection cost is essential for efficiency and resource management.

D. Prim's Algorithm

Prim's algorithm is a greedy method used to identify a minimum spanning tree within a weighted, undirected graph. It selects a group of edges that connect all vertices without forming any cycles, ensuring the sum of the edge weights is as low as possible. The process begins with any chosen vertex and progressively expands the tree by repeatedly adding the least expensive edge that links the existing tree to a new vertex until every vertex is connected.

III. IMPLEMENTATION

This paper conducts a comprehensive examination of Prim's algorithm implementation for identifying minimum spanning trees (MSTs) in wiring diagram optimization. The methodology employs C programming language with strategic library integration to achieve robust functionality. String.h and stdlib.h enable advanced string parsing for user-defined parameters, facilitate dynamic memory allocation for graph structures, and provide pseudorandom number generation via rand(). Time.h seeds random number generation using srand(time(NULL)) to ensure non-repetitive coordinate sequences.

```
srand(time(NULL));
for (int i = 0; i < n; i++) {
    points[i].x = (float)(rand() % 1000)/500 - 1;
    points[i].y = (float)(rand() % 1000)/500 - 1;
    points[i].z = (float)(rand() % 1000)/500 - 1;
    explored[i] = 0;
}
```

Next, we create a complete graph from every vertex within the diagram as the basis for our MST. We do this by iterating through every combination of vertices, creating an edge between them, and storing the edges in another array.

```
for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
        segments[c].idx1 = i;
        segments[c].idx2 = j;
        segments[c].dist = dist(points[i], points[j]);
        c++;
    }
}
```

Using this complete graph, we create a minimum spanning tree using Prim's algorithm, using the length of the edge as its weight.

```

sort(segments, c, sizeof(segment), comp);
tree[0] = segments[0];
explored[segments[0].idx1] = 1;
explored[segments[0].idx2] = 1;
while (iter < c && finalCount < n - 1) {
    if (!(explored[segments[iter].idx1] == 1 && explored[segments[iter].idx2] == 1) &&
        !(explored[segments[iter].idx1] == 0 && explored[segments[iter].idx2] == 0)) {
        tree[finalCount] = segments[iter];
        explored[segments[iter].idx1] = 1;
        explored[segments[iter].idx2] = 1;
        finalCount++;
        iter = 0;
    }
    iter++;
}
}

```

The results of the implementation are as thus:

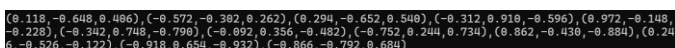


Fig. 8. Same series of coordinates, visualized in desmos 3D.



Fig. 10. Same list of edges, visualized in desmos 3D.

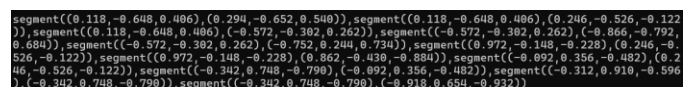


Fig. 11. Resulting MST from Prim's algorithm.

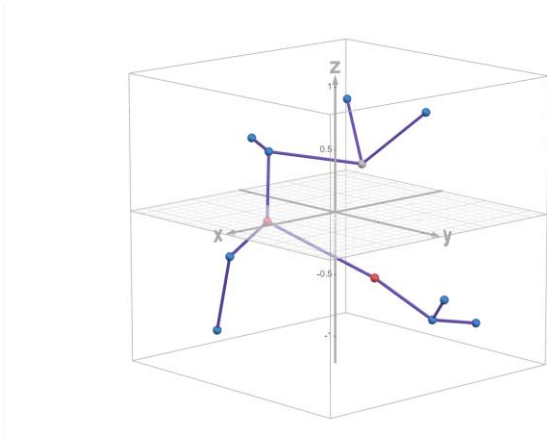


Fig. 12. The MST, visualized in desmos 3D.

From the results, we can see that Prim's algorithm is indeed successful in creating a MST from the set of vertices. Checking the distances between the two points that seem closest together (marked in red and white) show that the chosen edge is the shortest way to reach the right red point.

3D Distance Calculator	3D Distance Calculator
$(X_1, Y_1, Z_1) = [0.246, -0.526, -0.122]$ $(X_2, Y_2, Z_2) = [-0.092, 0.356, -0.482]$	$(X_1, Y_1, Z_1) = [-0.572, -0.302, 0.262]$ $(X_2, Y_2, Z_2) = [-0.092, 0.356, -0.482]$
Clear Calculate Answer: $d = 1.010825$ For: $(X_1, Y_1, Z_1) = (0.246, -0.526, -0.122)$ $(X_2, Y_2, Z_2) = (-0.092, 0.356, -0.482)$ Distance Equation Solution: $d = \sqrt{(-0.092 - (0.246))^2 + (0.356 - (-0.526))^2 + (-0.482 - (-0.122))^2}$ $d = \sqrt{(-0.338)^2 + (0.882)^2 + (-0.36)^2}$ $d = \sqrt{0.114244 + 0.777924 + 0.1296}$ $d = \sqrt{1.021768}$ $d = 1.010825$	Clear Calculate Answer: $d = 1.103132$ For: $(X_1, Y_1, Z_1) = (-0.572, -0.302, 0.262)$ $(X_2, Y_2, Z_2) = (-0.092, 0.356, -0.482)$ Distance Equation Solution: $d = \sqrt{(-0.092 - (-0.572))^2 + (0.356 - (-0.302))^2 + (-0.482 - (0.262))^2}$ $d = \sqrt{(0.48)^2 + (0.658)^2 + (-0.744)^2}$ $d = \sqrt{0.2304 + 0.432964 + 0.553536}$ $d = \sqrt{1.2169}$ $d = 1.103132$

Fig. 13. Distance calculations for the original edge (left) and the proposed edge (right, showing that the original edge is part of the MST).

B. Restrictions

In real-world spatial applications—such as plumbing layouts, architectural planning, or circuit design—physical barriers often prevent direct connections between vertices. Examples include:

- Structural obstacles like walls, columns, or load-bearing beams
- Functional constraints (e.g., preserving living spaces between fixtures)
- Safety regulations requiring clearance zones

These limitations necessitate modifying graphs to exclude impractical edges before applying algorithms like Prim's MST. Without this step, the resulting connections would violate real-world feasibility.

Any restriction that is composed of straight, orthogonal edges (aligned with Cartesian axes) can be modeled with a series of linear inequalities, like thus:

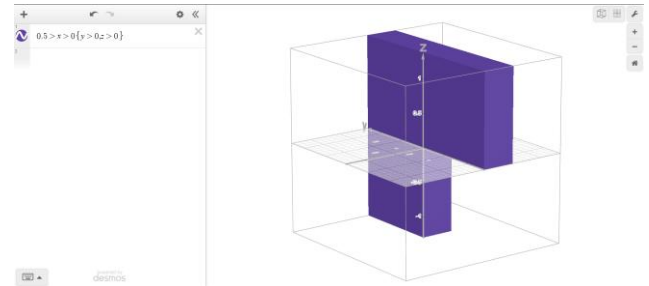


Fig. 14. Example set of restrictions and the visualization in desmos 3D.

We can see that the set of restrictions is composed of several linear inequalities joined by OR (commas) and AND (curly brackets) operands. This structure can be modeled with a tree.

- Root node checks edge viability
- Branches split into sub-conditions (AND/OR layers)
- Leaf nodes represent atomic inequalities

This structure enables efficient edge validation through recursive tree traversal.

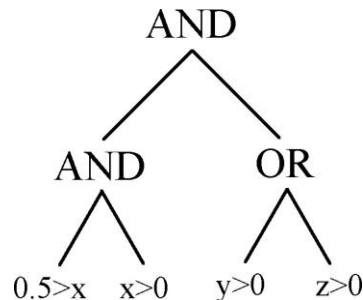


Fig. 15. Tree representation of the set of restrictions from Fig. 14.

To implement these restrictions, we check every edge to see if at least part of the edge fulfills the restrictions. If part of the edge does fulfill all the restrictions, we don't add it to the interim graph.

```

for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
        processTree(root, points[i], points[j]);
        if (root->Neff == 0) {
            segments[c].idx1 = i;
            segments[c].idx2 = j;
            segments[c].dist = dist(points[i], points[j]);
            c++;
        }
        resetTree(&root);
    }
}
  
```

Fig. 16. Updated edge generation snippet, now checking for restrictions before adding the edge to the array.

C. Results of Investigations on Restrictions

After generating another set of random coordinates, the result of the updated implementation is as thus:

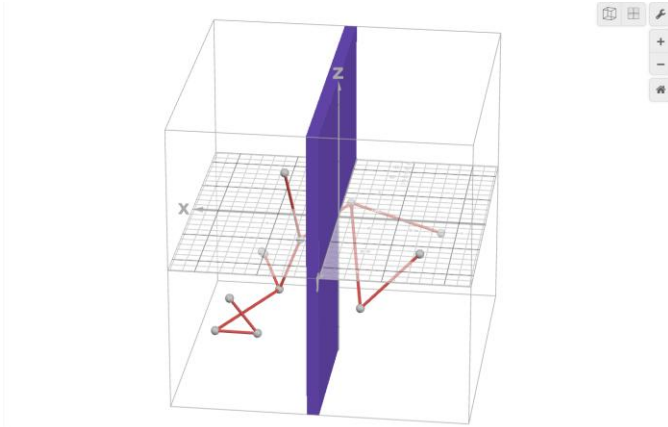


Fig. 17. MST generated from new set of coordinates.

From the results, we can see that the MST does follow the restrictions inputted into the program, validating our investigation.

IV. CONCLUSION

This study demonstrates that Prim's algorithm is an effective tool for optimizing architectural wiring diagrams by generating a minimum spanning tree (MST) that connects all electrical fixtures with the least total wire length. The implementation successfully models wiring layouts as weighted graphs, allowing the algorithm to identify the most efficient paths and thereby reduce material costs and installation time. Furthermore, the introduction of real-world restrictions—such as physical barriers or inaccessible areas—can be incorporated into the graph model using algebraic constraints, ensuring that the generated MST adheres to practical limitations commonly encountered in building design. The results confirm that Prim's algorithm not only produces optimal wiring layouts in idealized conditions but also remains robust and adaptable when faced with complex, real-life constraints. This approach offers a systematic and scalable method for improving the efficiency and safety of electrical installations in architectural projects.

V. FURTHER WORK

While the current approach effectively models spatial restrictions using linear inequalities and logical operators to exclude impractical edges, further progress can be achieved by incorporating non-linear restrictions. Many real-world constraints—such as curved architectural features, irregular safety zones, or complex functional boundaries—cannot be accurately represented by linear inequalities alone. By extending the model to support non-linear restrictions, such as quadratic or polynomial inequalities, circular clearance zones, or spline-defined obstacles, the graph pruning process can better reflect realistic environments. This advancement would require adapting the edge validation algorithm to handle more complex geometric checks, potentially leveraging

computational geometry techniques or numerical solvers. Consequently, the resulting minimum spanning trees (MSTs) would not only respect orthogonal and linear constraints but also conform to intricate spatial limitations, enhancing the applicability and precision of the algorithm in diverse real-world scenarios.

LINKS

Link to mega folder containing video:

<https://mega.nz/folder/bYwggLrK#jhJgcQYydN-cc353raSyEw>

Link to github repo:

<https://github.com/kalkabena/MST>

ACKNOWLEDGMENT

The author of this paper wishes to sincerely express heartfelt gratitude to God for His continuous blessings, unwavering guidance, and profound support throughout every stage of the experimental work and the entire writing process of this research paper. Without His divine presence, the successful completion of this study would not have been possible.

In addition, the author would like to extend deep appreciation to the lecturers of the IF1220 course, with special thanks directed to Dr. Ir. Rinaldi Munir, M.T. His dedication to teaching, generosity in providing extensive learning materials, and the creation of numerous valuable learning opportunities have greatly enriched the educational experience not only for the author but for all students enrolled in the course. The knowledge and insights gained under his mentorship have been instrumental in shaping the author's understanding and approach to the subject matter.

Finally, the author is profoundly grateful to his family for their unwavering love, patience, and constant encouragement throughout his academic journey. Their moral support and belief in his abilities have served as a vital source of motivation and strength, enabling him to overcome challenges and persist in his pursuit of academic excellence.

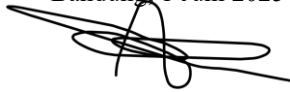
REFERENCES

- [1] Discrete Mathematics, R. Johnsonbaugh, 5th ed., 2001.
- [2] <https://www.edrawmax.com/house-wiring-diagram/>
- [3] <https://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php>

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Juni 2025



Nathanael Shane Bennet, 13524119