

Automated Graph-Based Methodology for Generating Specialized Cuts for the Long-term Transmission Network Expansion Planning Problem

Luis Fernando Mujica R.

Universidade Tecnológica de Pereira
Carrera 27 #10-02 Barrio Alamos - Colombia
luis.mujica@utp.edu.co

Juan Pablo Escobar V.

Universidade Tecnológica de Pereira
Carrera 27 #10-02 Barrio Alamos - Colombia
pablo.escobar@utp.edu.co

Laura Mónica Escobar V.

Universidade Tecnológica de Pereira
Carrera 27 #10-02 Barrio Alamos - Colombia
lamoescobar@utp.edu.co

ABSTRACT

This research focuses on implementing a graph-based methodology to identify specialized cuts for the optimization planning problem. The objective is to improve a lengthy and intricate process that requires a deep understanding of the test system and deep data analysis to identify cycles and possible repetitive behavior within the transmission power system. The development and implementation of valid inequalities typically require extensive analysis, which may not always significantly impact the selected system due to its strong dependence on time limitations. Therefore, this research aims to automate this process to quickly produce specialized cuts, enabling the resolution of long-term transmission network expansion problems in a shorter time frame, especially for highly complex systems.

KEYWORDS. Transmission expansion planning. Power system planning. Mathematical optimization. Data analysis. Valid inequalities. Graph.

Topics: Operations Research in the Energy, Oil, and Gas Area (EN&PG), Mathematical Programming (PM).

1. Introduction

The long-term transmission expansion problem aims primarily to establish the transmission network to be built for a future scenario of static generation and demand, minimizing both investment and future operation costs [Garver, 1970]. A deficient planning not only affects the future operation of the network but also has a significant impact on subsequent planning processes since the base network (the existing network) adds complexity when addressing an electrical expansion problem. In the specialized literature, different mathematical models are presented, such as linear and nonlinear models, which represent the behavior of the electrical network with greater simpler or greater accuracy [Gallego et al., 2010]. However, these models are difficult to solve and highly depend on different factors as topology and size, consequently requiring huge computational effort and memory capacity, when different optimization techniques are used to solve them [Escobar, 2008], [Escobar Vargas, 2018]. On the other hand, linear models, which have a lower degree of difficulty and, consequently, a lower degree of accuracy in representing the behavior of an electrical network. These models have proven significant advantage in addressing specifically long-term challenges. By providing only the necessary parameters for designing the future transmission network, they enable the exclusion of redundant parameters and behaviors, restructuring the calculation and identification process, and allowing the analyst to use linear techniques or exact optimization techniques, which enables the methodology to find the optimum solution. En works like [Haffner et al., 2000], strategies such as Branch and Bound are employed to solve the planning problem, aiming to ensure its optimality. For small-scale systems, other works like [Zhang et al., 2012] and [Alguacil et al., 2003] have addressed the planning problem mainly using heuristic [Monticelli et al., 1982], metaheuristic [Escobar Zuluaga et al., 2004], and other solution strategies. Specialized literature contains published works allowing for a more detailed review of the literature, particularly focused on transmission network expansion planning, such as [Latorre et al., 2003] and [Lumbreras and Ramos, 2016]. Since the precision of these models tends to be similar to nonlinear models when analyzing deterministic scenarios. Adequate planning of transmission expansion (TEP) not only jeopardizes the supply of electrical energy but also underscores safety and security concerns. Consequently, TEP needs periodic reassessment to adapt to evolving circumstances, such as new projects or changes in generation and demand forecasts. This requires the development of new strategies to ensure the system finds a good quality solution, y coherent periods of time, especially for NP-Hard type problems [Duque et al., 2014].

Through traditional analysis, different characteristics and behaviors emerge, leading to the discovery and construction of new methods and methodologies that can improve computational efficiency and memory use reduction, in some instances, enable the model and technique to find the optimal solution. Data mining, as shown in specialized literature, plays an important role in finding various patrons and behavior related with complex optimization problems, often serving as an important foundation to identify the underlying factors contributing to create effective strategies [Florian and Kenneth, 2017], [Escobar Vargas, 2018]. Based on this analysis, normal system behavior can show us which path to follow to find these patterns, from which new specialized cuts or specialized techniques can be created and added to simplified mathematical model to ensure a better way to solve complex optimization problems. The topic of reducing computational effort remains a contentious issue in specialized literature. While companies typically prioritize obtaining optimal solutions over the time required to achieve them, this mindset often underestimates the significance of emerging methodologies and heuristics and their potential to enhance future solutions. Additionally, the ongoing restructuring of the electricity sector gives increases the complex characteristics related with a diverse and competitive framework of system. Consequently, network planning must

adapt to meet the future surge in demand, continuous changes related with generation and demand [Zolezzi and Rudnick, 2002]. Based on the above, this study proposes developing a graph-based methodology to replace the second Kirchhoff's law of the linear DC model, by automating the process of analyzing the selected system, its solution with different mathematical models, and the generation of directed graphs generated from the behaviors and patterns found in each solution. This strategy seeks to use this graphs to generate valid inequalities, a process that has in previous works demanded months or even years of analysis and construction time, now achieved within hours or minutes depending on the test system. Since the methodology is focused on NP-Hard transmission problems, the resulting valid inequalities created in the first part of the process, are introduced on the improved mathematical models created in previous investigations [Duque et al., 2014], [Escobar Vargas, 2018], [Skolfield et al., 2022], aiming to obtain the same optimal solution as the reduced linear DC model but with reduced computational effort.

2. Proposed methodology

The proposed methodology aims to automate the process of analyzing and constructing valid specialized inequalities used to solve the long-term transmission planning problem, previously applied by [Cárdenas Alzate, 2020] and [Escobar Vargas, 2018], in which the Brazil-southern and the Brazil-northeastern test systems were solved. In both studies, the authors developed specialized cuts for their respective long-term transmission expansion problems, with the aim of reducing the computational effort required for their solution. These cuts were the result of several years of detailed data analysis, and tests with different constraints combinations to ensure a better behavior. Therefore, the objective of this work is to develop a methodology that automates the generation of these specialized cuts based on cycles, like those implemented in [Cárdenas Alzate, 2020], using artificial intelligence and data science techniques to identify patterns that help determine which cuts should be incorporated into the model.

The proposal starts by applying graph theory techniques and heuristics to detect fundamental cycles in the transmission system. Identified cycles are then analyzed and determines if it's critical, and generates an associated constraint incorporated into the transportation model. This iterative process continues until no new corridors are added to the system. This approach aims to achieve solutions equivalent to the DC model without using the second Kirchhof law in a traditional manner. When discussing critical cycles in the transmission expansion planning problem, it refers to cycles that contain electrical corridors with power flows very close to their capacity limits. Additionally, the sum of the power/reactance product of the lines forming the cycle is small when compared with corridor with the same starting and ending bus on the test system, and based on these, the code generates the constraints that would prevent the model to calculate this behavior [Cárdenas Alzate, 2020], [Escobar Vargas, 2018]. The results of both studies demonstrated the effectiveness of this strategy.

As will be evidenced later in the document, the number of cycles in a graph increases exponentially with the increase in its size and complexity. In light of this, the graph was optimized by eliminating unattractive candidate lines. In [Escobar Vargas, 2018], lines with potential interest for a test system were identified by analyzing the characteristics of the optimal solution provided by various mathematical models, including:

- Flow Transport Model.
- Enhanced Transport Model.

- Linear Hybrid Model.
- DC Disjunctive Transport Model with minimal effort.

The transport model, by excluding the second Kirchhof law, require less computational effort and reach the optimal solution in short times. In contrast, the hybrid and minimal effort disjunctive models, although they incorporate this law, have distinctive characteristics that influence their execution time. The Hybrid Model applies the second law only to new lines, while the DC disjunctive model with minimal effort significantly expands the maximum flow limit allowed for all lines, thus facilitating a more agile resolution. The purpose of using these models is that, by offering solutions comparable to the DC disjunctive model, their results serve as a reference to discern which candidate lines could be part of the solution of the disjunctive model and which are considered unattractive because they do not appear in any solution. To achieve this, an interface for communication with Cplex was integrated into the developed program, which allows resolving the test scenario with each of the mentioned mathematical models. The responses obtained are used for the program to automatically identify the unattractive lines, thus excluding them in the search for base cycles.

It is crucial to note that this is the starting point of a training process, which will be improve with other strategies in the future, since this is a investigation in progress. Metrics such as the time needed to solve the problem, computational load, cost of the response, and identification of corridors at capacity limits will be considered to evaluate the strategy's performance, generating reward or penalty signals that will guide the automated process in the right direction.

2.1. Cycle search

In the literature, a cycle is defined as a subgraph that forms a sequence of vertices where the first vertex is the same as the last, and each vertex in the sequence is connected to the next by an edge in the graph. However, for this work, the cycles of interest are called simple or elementary cycles, which are a type of cycle that does not contain repeated vertices except for the first.

The proposed algorithm uses a modified Depth-First Search (DFS), which originates from graph theory and is used to discover paths connecting all nodes in a graph. Is well known technique that involves traversing a path from a starting node to an end node, making it an algorithm capable of finding paths from start to finish, Which, like the original algorithm, expands one path at a time from the current vertex while recording the path taken so far. To identify cycles, it checks that the current vertex does not have a path to a vertex that is part of the existing path. If such a path exists, it means a cycle has been found ; thus, the portion of the path from the visited node to the end of the path is taken. To prevent the program from entering infinite loops, each expanded node is marked as visited to avoid traversing the same path again. Once all nodes are marked as visited, the execution ends.

However, although marking nodes as visited prevents infinite loops, it causes the search to be non-exhaustive, so not all cycles are found in each execution. To maximize the number of cycles found, the cycle search is repeated from each node in the graph as the starting node, and each time the search begins, all nodes are marked as unvisited. It is important to note that the DFS algorithm typically expands the current vertex by taking the first available edge. Nevertheless, the graph's topology can lead to some cycles not being detected. To avoid this, controlled randomness was introduced in selecting the edge by which the current vertex is expanded. Each time an edge is

selected, it has a lower probability of being selected again when this vertex is visited from another starting node. The idea behind this controlled randomness is to encourage expansion through less-explored paths, thus promoting exploration in the graph. Considering the above, it is concluded that a complete search ends once all nodes in the graph have been used as starting nodes. However, as the graph's size and complexity increase, a single complete search is insufficient to find all cycles. Therefore, several complete searches are conducted to find all cycles, keeping track of the times each edge has been used and thus continuing to favor the exploration of new paths.

Then to improve the analysis it is added to the process techniques such as node clustering, dominance theory, and node coordinates for base cycle detection. Subsequently, a Python code will be implemented to adapt the analysis and the solutions found in the mathematical models discussed before to compare the power flow direction in each solution, as has been done in previous works [Escobar Vargas, 2018]. This code will solve the models with AMPL during an iterative process. The solutions will be returned to the algorithm, and an analysis will determine which identified base cycles are present in the solution. These cycles will be passed through an analysis based on an overload heuristic, which will classify the cycle as critical or not. Based on this classification, a new constraint based on the cycle will be generated and added to an improved transportation model for resolution with AMPL and subsequent evaluation. It is important to note that the DFS search will not be part of the iterative process; this approach is taken mainly to avoid repeated execution of this subroutine and thereby expedite the process.

2.2. Determining base cycles

In this work, base cycles are considered as simple cycles, which are cycles that do not contain an internal path connecting two vertices of the cycle. This definition makes the detection of base cycles challenging because it depends on the spatial location of each node, as it is necessary to differentiate between internal and external paths of the cycle. By considering this characteristic, it is possible to assert that if a path connecting two or more nodes exists within the cycle, this path can only contain vertices located inside the cycle's perimeter. This characteristic was addressed by implementing the Ray Casting (RC) algorithm to the proposed final algorithm, which identifies whether a point is inside or outside a given perimeter. The RC algorithm is based on defining a perimeter and projecting a ray in any direction. If the ray crosses a perimeter edge zero or an even number of times, it indicates that the point is outside the perimeter; if the number of crossings is odd, it indicates that the point is inside the perimeter.

Thus, for each cycle found, a perimeter is constructed, and it identifies which nodes are inside it. Once the internal nodes of the cycle are identified, a DFS algorithm is used to search for all possible paths through the internal nodes that connect two nodes belonging to the cycle. However, it is possible to find paths that directly connect two nodes, as shown in Figure 1. To detect this type of path, an auxiliary point is calculated in the middle between both nodes, and using the RC algorithm, it is identified whether this point is inside or outside the cycle. Knowing where this point is located, it can be determined whether the path crosses the interior or exterior of the cycle. Figures 2a and 2b show an internal and external direct path, respectively.

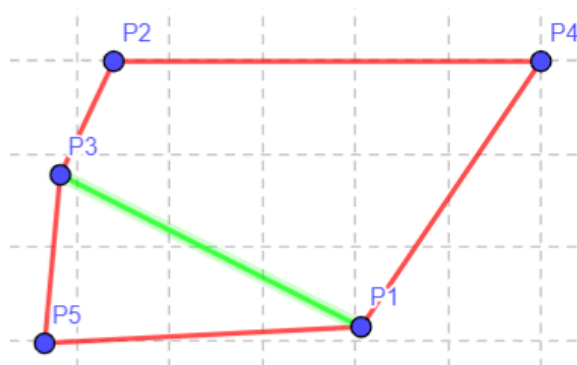


Figure 1: Graphic representation of a cycle containing a direct internal path between two nodes.

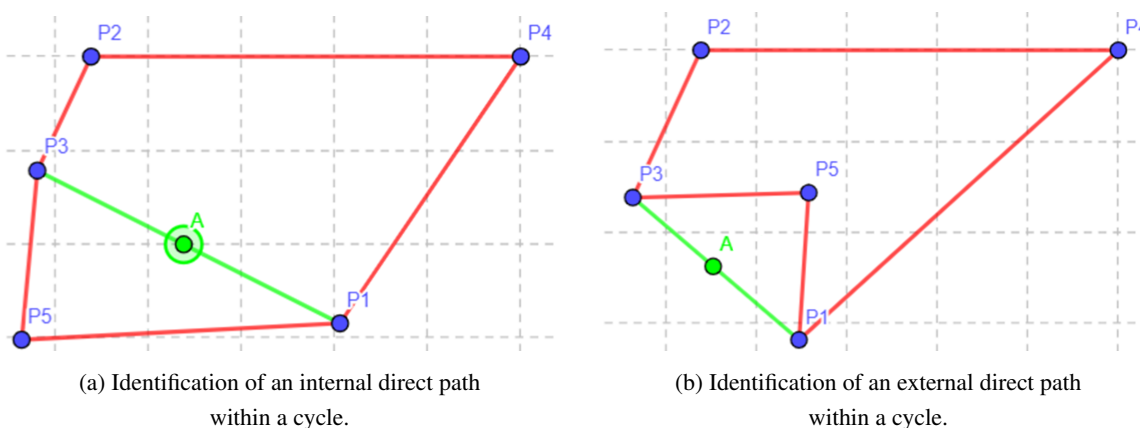


Figure 2: Identification of internal (a) and external (b) graph

3. Proposed algorithm for the transmission expansion planning problem

Since this algorithm is focused on the problem transmission expansion planning problem, the general case to be analyzed includes both existing lines and candidate lines. The former are the lines that already exist in the system, while the latter are lines that can be constructed. Therefore, the objective is to determine which candidate lines should be built so that the electrical system meets a future deterministic scenario of generation and demand. This means that our graph (the transmission system) transforms into a different graph depending on the found solution. Consequently, the lines that appear in the solution can cause cycles to appear or disappear, and cycles that were previously base cycles may no longer be. This is why, when working with electrical systems, all candidate lines are initially treated as existing lines, and the cycle search in this graph is conducted as previously described.

However, now that the cycles and internal paths can be composed of a combination of base and candidate lines, it is necessary to identify which candidate lines must be active for each cycle to exist, and if this cycle has internal paths, which candidate lines must be inactive for an internal path not to form and for it to be considered a base cycle. To achieve this, once all internal paths of a cycle are found, the vertices connected via a candidate line are identified. After identification, a vector is generated indicating which connections must be active for the cycle to exist. For the internal paths, a similar procedure is carried out: for each path, the vertices connected via a candidate line are

identified, and a vector is created indicating the lines that must be active for the path to exist. Once the analysis of all cycles is complete, the system will have:

- Potential base cycles: are all cycles that, depending on the response, can become base cycles.
- Active cycles: are all the cycles that are active based on the input response.
- Base cycles: These are the cycles that, among the active cycles, are considered base cycles because they do not contain an internal path, or no internal path has been created based on the response.

The corresponding algorithms related to the proposed methodology are shown below:

Algorithm 1 Cycle Detection using Depth-First Search (DFS)

```

1: for all vertices  $v$  do
2:   Initialize  $path = [v]$ 
3:   Initialize  $visited = [v]$ 
4:   while path is not empty do
5:     current = last element of path
6:     new = choose a neighbor with a probability proportional to its weight
7:     if new is in visited then
8:       cycle = sublist of path from new to current
9:       if cycle is not in cycles then
10:        Add cycle to cycles
11:      end if
12:    end if
13:    if all neighbors of current are in visited then
14:      Remove current from path
15:    else
16:      Recalculate the weight of new
17:      Add new to path
18:    end if
19:  end while
20: end for
  
```

Algorithm 2 Find Dependencies

```

for cycle in cycles do
  for  $v$  in cycle[0 : len(cycle) - 1] do
    if  $airs(v, v + 1)$  is a candidate then
      dependency  $\leftarrow$  dependency +  $(v, v + 1)$ 
    end if
  end for
  for internalPath in cycleinternalpath do
    for  $v$  in internalPath[0 : len(internalPath) - 1] do
      if  $aris(v, v + 1)$  is a candidate then
        dependencyPath  $\leftarrow$  dependencyPath +  $(v, v + 1)$ 
      end if
    end for
  end for
end for
  
```

Algorithm 3 Finding Internal Paths in a Cycle

```

1: for all vertex in cycle do
2:   Perimeter  $\leftarrow$  perimeter with the vertices of the cycle
3:   for all  $v$  in vertex not in cycle do
4:     Raycast algorithm( $v$ )
5:     if  $v$  is inside Perimeter then
6:       Internals  $\leftarrow$  Internals  $+$   $v$ 
7:     end if
8:   end for
9:   newGraph  $\leftarrow$  Graph  $\cap$  cycle  $\cap$  Internals
10:  cycleInternalPaths  $\leftarrow$  DFS algorithm(newGraph)
11: end for
  
```

Once the cycles and dependencies have been determined, both for being active and for being existing lines, they are simply compared with the answer, and it is determined which cycles are active and which are base.

4. Test and results

To test the cycle-finding capability of the implemented methodology, we used reduced-size graphs in which we identified all cycles and subsequently determined whether they were base cycles. Additionally, planar graphs were chosen to facilitate the observation of base cycles. Four graphs with diverse topology and characteristics were implemented to observe the algorithm's behavior with each of them. It is important to highlight that the obtained results were compared with those obtained using the networkx library. This library is capable of finding all simple cycles in a graph, allowing this response to be taken as a reference point to assess the performance of the implemented technique. It is crucial to emphasize that although networkx has the ability to efficiently find all simple cycles in a graph, finding all cycles within a large graph already constitutes a complex problem. For the purpose of obtaining base cycles, it is inefficient since most simple cycles will be discarded as they do not fit the definition of base cycles. This assertion is evidenced in subsequent test.

The data for the Brazil-southern and Brazil-northeastern test systems can be found in [Cárdenas Alzate, 2020] and [Escobar Vargas, 2018], along with the optimal solution for Brazil-southern and the best-known solution for Brazil-northeastern, both addressed using various mathematical models.

4.1. Considerations

To confirm the performance of the proposed methodology, tests were conducted using two test systems related to the long-term transmission planning problem: the Brazilian southern test system and the Brazilian northeastern test system. To contrast the results, a performance comparison was made between the number of cycles found by the Networkx library and the proposed methodology. There for these tests involved two analysis scenarios each:

- Scenario 1: The quantity of simple cycles, active cycles, and base cycles is analyzed without considering the candidate lines.
- Scenario 2: The quantity of simple cycles, possible base cycles, active cycles, and base cycles is analyzed when activating all active candidate lines.

4.2. Brazilian southern test system

The Brazilian-southern test system consists of 79 lines (edges) and 46 buses (nodes), in Figure 3, all the existing lines (Black) and candidate lines (Blue) are shown, the graph is given by the code as part of the analysis:

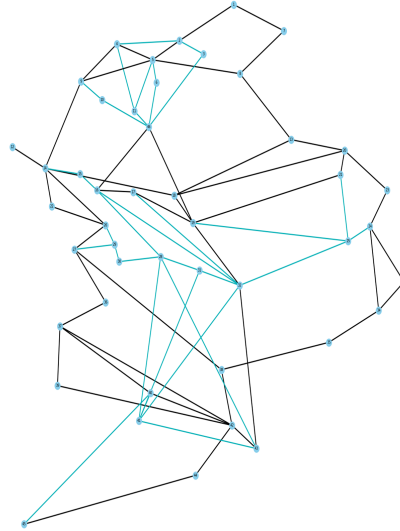


Figure 3: Brazilian southern test system with all active candidate lines

4.2.1. Case 1

In this scenario, both the implemented methodology and the Networkx library manage to find the same number of simple and base cycles, which are 447 and 14, respectively. However, the difference lies in the execution time, where the proposed algorithm implemented outperforms the Networkx library, being 14.5 times faster. Table 1 shows the comparison of both results. Once the cycles are found, the valid inequalities related to each cycle are automatically generated by the code, then this specialized cuts will be added to the enhanced model to continue the solution process of this research. In this document, only the results obtained with the developed algorithm will be presented since the generated cuts have been presented in previous works [Escobar Vargas, 2018], [Cárdenas Alzate, 2020], [Skolfield et al., 2022].

4.3. Case 2

For this analysis, all candidate lines were activated, as shown in Figure 3. Since the system is still relatively small, finding simple cycles with the Networkx library is feasible, managing to find 10,771,083 cycles. However, due to the large number of cycles, the required memory becomes limiting to analyze them all. Nevertheless, despite this limitation, the library can conclude that the search for all cycles is possible, allowing them to be analyzed in parts. It was determined that 12,545 are possible base cycles and 111 are bases, while the rest of the cycles are not of interest. The methodology, was limited to 20 minutes of calculations and search, during which only 2,907,179 simple cycles were found. Despite representing only 26.9% of the total number of simple cycles, 11,814 possible base cycles were found, representing 94% of the real analysis interest. Additionally, the program manages to find the 111 base cycles, as shown on table 1.

It is important to note that although the NetworkX library may be limited in finding a certain number of cycles, the search technique it uses tends to favor the identification of the longest cycles (big size cycles) at the beginning of the search. This means that the shorter cycles, which are more likely to belong to possible bases, are found towards the end of the search. Therefore, it is necessary to allow the technique to complete the process to locate a greater number of cycles of interest, which is feasible for small graphs.

However, when dealing with larger systems, the exhaustive search for all cycles becomes computationally expensive, which can cause the program to terminate due to lack of memory. The NetworkX library was limited to finding the first 3 million simple cycles. Of all these cycles, only 1,234 possible base cycles and 7 base cycles were found, representing 9.8% and 6.3% respectively of the total cycles of interest to the valid inequalities generation methodology. On the other hand, by limiting the methodology to only 20 minutes of time, 515,944 simple cycles were found, of which 8,709 are possible base cycles and 111 are base cycles, representing 69.4% and 100% respectively of the total cycles of interest. This comparison can be found on Table 1 on the limited analysis.

	Proposed methodology					Networkx library				
	Total time (s)	Found cycles	Potential cycles	Base cycles	Limit time (s)	Total time (s)	Found cycles	Potential cycles	Base cycles	Limit time (s)
Case 1	1.1	474	14	14	1.200	0.08	474	14	14	1.200
Case 2	1.200	2.907.179	11.814	111	1200	1.986	10.771.083	12.545	111	1.200
limited	500	515.944	8.709	111	500	500	3.000.000	1.234	7	500

Table 1: Brazilian southern test system comparative summary of the library and the methodology

4.4. Brazilian northeastern test system

The Northeast-Brazil test system consists of 183 lines (edges) and 87 buses (nodes) as shown in Figure 4, all the existing lines (Black) and candidate lines (Blue) are shown, the graph is given by the code as part of the analysis. The Brazilian north-northeastern test system is a NP-Hard problem and highly complex even for the initial part of the algorithm which only focus is the cycle identification problem.



Figure 4: Brazilian north-northeastern test system with all active candidate lines

4.5. Case 1

Similarly to the previous test system, in this scenario, the implemented methodology manages to find the same amount of 6355 simple cycles as the library, with 85 base cycles and possible bases. The difference lies in the execution time, where the algorithm implemented by the library is 237.5 times faster. The comparison is shown in the Table 2.

4.6. case 2

For this analysis, all candidate lines were activated as shown in Figure 4. In this case scenario, due to the complexity of the graph and the limitations of the equipment, the NetworkX library cannot find all possible simple cycles without running out of memory. On the other hand, the implemented methodology continues to be able to find 1,800,082 simple cycles and analyze them without running out of memory. Of these, 489,620 are possible base cycles, and 3181 are actual base cycles. For comparison, the NetworkX library was limited to a time where it could find the same number of simple cycles. Despite finding the same number of cycles in less time, NetworkX only managed to identify 1 possible base cycle and 0 base cycles. The table shows the comparison of the results obtained when searching for cycles with the implemented technique and with NetworkX, respectively.

	Proposed methodology					Networkx library				
	Total time (s)	Found cycles	Potential cycles	Base cycles	Limit time (s)	Total time (s)	Found cycles	Potential cycles	Base cycles	Limit time (s)
Case 1	44	6355	85	85	1.200	0.2	6355	85	85	1.200
Case 2	900	1.800.082	489.620	3181	900	900	-	-	-	900
limited	700	1.478.820	436.830	3097	700	700	1.800.000	1	0	700

Table 2: Brazilian north-northeastern test system comparative summary of the library and the methodology

5. Acknowledgements

This work was supported by the Master's Program in Electrical Engineering, the Electrical Technology Program, and the Planning Group at the Universidad Tecnológica de Pereira (Colombia).

6. Conclusions

Based on the conducted tests, it can be concluded that despite NetworkX being more efficient in terms of time when finding cycles, its search strategy may limit performance in identifying possible base cycles. This is because it requires a more exhaustive search, which is a drawback when analyzing complex systems and can even prevent the completion of the search due to lack of resources.

On the other hand, the technique implemented in this work, despite being less efficient in identifying cycles, tends to locate those relevant cycles more easily. This enables the technique to find a greater number of relevant cycles without the need to identify all cycles, thus saving computational resources. This approach allows for the identification of a considerable number of cycles of interest without the necessity of finding all cycles, which can lead to resource savings in computational tasks. Regarding the advantages and limitations of NetworkX versus the implemented technique. While NetworkX may offer faster results, its efficiency in identifying relevant cycles, especially base cycles, can be compromised. In consequence, the proposed methodology and algorithm may take longer in specific cases, but has a higher chance of identifying the cycles of interest efficiently, which are the necessary cycles to create the valid inequalities.

References

- Alguacil, N., Motto, A., and Conejo, A. (2003). Transmission expansion planning: a mixed-integer lp approach. 18(3):1070–1077.
- Cárdenas Alzate, P. P. (2020). *Planeamiento de la expansión de redes de transmisión a largo plazo usando una formulación basada en ciclos*. Universidad tecnológica de Pereira.
- Duque, A., Escobar, A., and Gallego, R. (2014). Multistage transmission expansion planning via network partitioning and principal variables identification. Medellin. IEEE.
- Escobar, A. H. (2008). *Análise crítica de aspectos de modelagem matemática no planejamento da expansão a longo prazo de sistemas de transmissão*. Ilha Solteira.
- Escobar Vargas, L. M. (2018). *Specialized models for the long-term transmission network expansion planning problem*. Ilha Solteira.
- Escobar Zuluaga, A., Gallego Rendón, R., and Romero, R. (2004). Multistage and coordinated planning of the expansion of transmission systems. 19:735–744.
- Florian, A. and Kenneth, S. (2017). What makes a solution good? the generation of problem-specific knowledge for heuristics. Antwerpen.
- Gallego, R. A., Romero, R., and Escobar, A. (2010). *Modelos usados en el planeamiento de la expansión a largo plazo de sistemas de transmisión de energía eléctrica*. Pereira.

- Garver, L. L. (1970). Transmission network estimation using linear programming. PAS-89(7): 1688–1697.
- Haffner, S., Monticelli, A., Garcia, A., Mantovani, J., and Romero, R. (2000). Branch and bound algorithm for transmission system expansion planning using a transportation model. 147:149–156.
- Latorre, G., Cruz, R., Areiza, J., and Villegas, A. (2003). Classification of publications and models on transmission expansion planning. 18(2):938–946.
- Lumbreras, S. and Ramos, A. (2016). The new challenges to transmission expansion planning. survey of recent practice and literature review. 134:19–29.
- Monticelli, A., Santos, A., Pereira, M. F., Cunha, S. H. F., Parker, B. J., and Praca, J. G. (1982). Interactive transmission network planning using a least-effort criterion. PAS-101:3919–3925.
- Skolfield, J. K., Escobar, L. M., and Escobedo, A. R. (2022). Derivation and generation of path-based valid inequalities for transmission expansion planning. 312(2):1031–1049. ISSN 1572-9338. URL <https://doi.org/10.1007/s10479-022-04643-1>.
- Zhang, H., Vittal, V., Heydt, G., and Quintero, J. (2012). A mixed-integer linear programming approach for multistage security-constrained transmission expansion planning. 27:1125–1133.
- Zolezzi, J. M. and Rudnick, H. (2002). Transmission cost allocation by cooperative games and coalition formation. 17(4):1008–1015.