Optimization of the Algorithm for Finding Spanning Trees of a Graph

A.R. Dzhandigulov[1], D.S. Akhmetbaev[2,*], A.D. Akhmetbaev[3]

[1] L.N. Gumilyov Eurasian National University, Nur-Sultan, Republic of Kazakhstan
[2] S. Seifullin Kazakh Agro Technical University, Nur-Sultan, Republic of Kazakhstan
[3] Information Systems Directorate, «Kazakhtelecom» JSC, Almaty, Republic of Kazakhstan

***Abstract* — The paper presents one of the possible variants of optimization of the algorithm for finding all spanning trees of the graph from the point of view of the application of the results of calculations in the topological study of complex electric power systems.**
The first stage of the topological analysis of the network is the search and determination of the values of all possible trees of the graph corresponding to a given energy system. The computational complexity of the algorithm for finding and determining the weights of all spanning graphs increases as the branches and nodes of the power grid network increase.
The main idea of optimization is to build special classes of trees. At the same time, in the process of grouping, parts of future graphs that are "parent" for groups of graphs are allocated and corresponding graphs of significantly smaller dimension are constructed. If the analysis is time-consuming, the grouping process can be applied to each graph found.

***Index Terms* — network topology, graph tree.**

I. Introduction

The need to construct all spanning trees of a given graph appears in many cases:

- when searching for an optimal tree in some sense, when the optimality criterion is so complex that the construction of an optimal tree at once is impracticable [1];
- when finding the transfer function of the system [2];
- in the topological analysis of current distribution coefficients in electric networks [3], etc.

The problem of finding all spanning trees of a graph is computationally time-consuming, since their number grows exponentially depending on the number of edges of the graph under study. Therefore, there is a need to improve the search algorithms for possible graph trees, in relation to the problems of the electric power industry. For example, to find the current distribution coefficients of electric networks, the ratio of the sum of the weights of specific trees to the sum of the weights of various trees of the graph is calculated [4]. In this case, the weight of a tree is understood as a complex product of the conductivities of its branches, taken with a sign, depending on the orientation of the branch.

In [5,6] an effective algorithm of directed search and determination of weights of possible graph trees without involvement of previously defined trees is implemented. The main idea of this algorithm is as follows. Let be a connected graph with n-nodes and m-branches. We produce a directed (excluding repetition) selection of n-1 branches from m - given. For each sample, we check for the presence of a cycle. In this case, the check is carried out from two sides, that is, from the first and last branches of the sample. If any subset of branches from the sample forms a loop, then all variants containing this subset are excluded from the search. This algorithm works successfully in the case of a weakly filled vertex neighborhood matrix ($n \geq 0.8$ m) [7]. With a more complete matrix of incidents, the execution time of the program increases many times.

In this paper, we propose an optimal algorithm for finding graph trees based on partitioning the entire set of trees, into disjoint classes, by considering several generated graphs instead of the original graph. These graphs are obtained from the original graph, by "removing" the nodes and branches selected in a certain way. The graphs obtained in this way have significantly fewer spanning trees, and from the latter, the spanning trees of the original graph are obtained by the inverse addition of "remote" branches and nodes.

* Corresponding author.
E-mail: axmetbaev46@mail.ru

http://dx.doi.org/10.25729/esr.2019.03.0009
Received August 11, 2019. Revised October 26, 2019.
Accepted Novenber 3, 2019. Available online December 25, 2019.

II. Optimization of the algorithm

Let be a graph with n-nodes and m-branches. The optimization process consists of several steps.

Step 1. Selection and "removal" of hanging branches with terminal vertices (nodes), as they are included in any spanning tree.

Step 2. Selecting and" gluing " paired branches, that is, branches that connect the same nodes. Since, in the end, when calculating the current distribution coefficients, only the products of the complex resistances of the branches are involved, the resistances of the paired branches simply add up.

Step 3. Let n_1– be the number of remaining nodes, m_1– be the number of remaining branches. We allocate a set of V-nodes incident to only two branches. Let R be the set of edges incident to the set of nodes V. Let the powers of these sets be |V| = α, |R|=β. It is obvious that α ≤2β.

Step 4. From the set R we choose those branches that are "repeated" twice. We also set the condition that the repeating branch is not part of a triangular cycle, although this case can be considered, if necessary, for further splitting the set of spanning trees into classes. There are cases when such repetitions go one by one, forming a chain of branches. In this case, select only one of these branches. Let $R_1 = \{r_{i,1}, r_{i,2} \dots r_{i,k}\}$ be a subset of the set R. Now, instead of the original graph, consider the graphs in which the specified branches $r_{(i,j)}, j = 1 \dots k$ of the set R_1 is either exactly there (denoted by $r_{(i,j)} = 1$), or exactly not (similarly denoted by $r_{(i,j)} = 0$). Obviously, there will be 2^k such graphs.

In the variant, where of k branches there are no *l*-branches and *k-l* branches there are, we will have 2^l hanging branches. And after applying step 1 to such a graph, we get a graph whose number of nodes is $n^1\text{-}2^l\text{-}(k\text{-}l) = n^l - (k+l)$ nodes, and the number of branches $m_1\text{-}3l - (k\text{-}l) = m_1 - (k+2l)$.

Step 5. If in some graph there is a branch $r_{(i,j)}$, which is incident to two nodes v_{j1}, v_{j2}, then instead of this graph, we can consider a graph, which instead of such two nodes will have one node, and the number of branches respectively decrease by one.

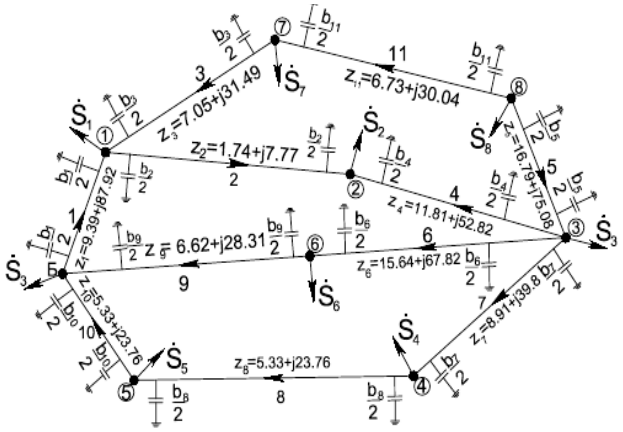

Fig. 1. Scheme of 220 kv network.

Table 1. A set of nodes incident to only two branches.

Nodes -v_i	2	4	5	6	7	8
1st branch $r_{1,i}$	2 (1)	7(3)	8(4)	6(3)	3(1)	11(7)
2nd branch $r_{2,i}$	4(3)	8(5)	10(9)	9(9)	11(8)	5(3)

As a result of constructing the subtrees of the graph with such steps we get:

1. Reducing the time spent on the study of such graphs.
2. The possibility of parallelization of the calculations of the graphs.
3. All "deleted" branches are necessarily included in the spanning tree of the original graph.
4. Adding "deleted" branches back to the spanning trees is necessary to find possible trees of the original graph.
5. The set of spanning trees corresponding to the graphs in question do not intersect and their Union yields all possible trees of the original graph.

III. An example application of the optimization process

Let's describe the optimization process on the example of the scheme (Fig. 1), is given in [7].

In this work the number of possible trees equal to 85 is obtained. In this scheme, the number of nodes $n = 9$, the number of branches $m = 11$.

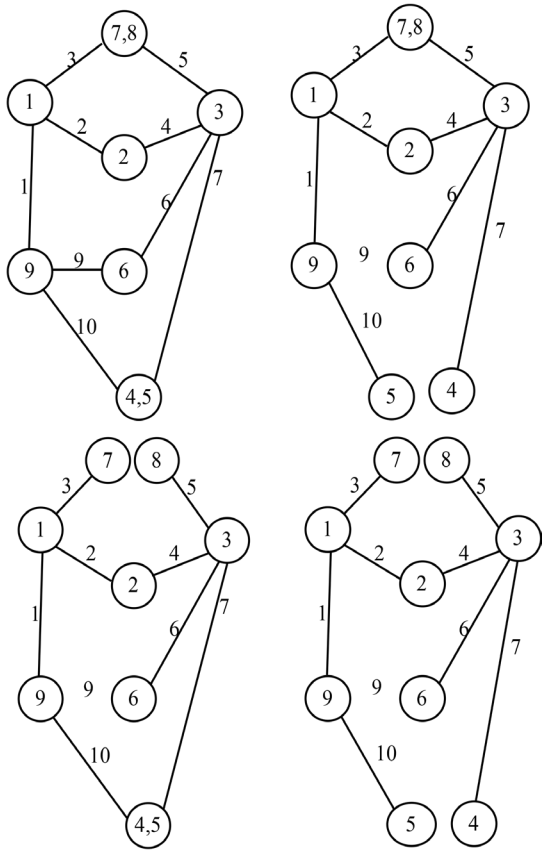

Fig. 2. Splitting the original graph into 4 graphs.

Step 1. In the scheme under consideration there are no hanging branches with terminal vertices (nodes).

Step 2. In the scheme under consideration, there are no branches linking the same nodes.

Step 3. In the scheme under consideration, $n_1 = n = 9$, $m_1 = m = 11$. For the considered scheme we have the following data given in table. 1.

The nodes associated with this branch are indicated in parentheses.

Step 4. From the set R we choose those branches that are "repeated" twice. In the above table. 1 these are branches 8 and 11. In our case $k = 2$. Thus, the original graph corresponding to the scheme shown in Fig. 1, is divided into 4 graphs (fig. 2).

In the first graph, instead of two pairs of nodes 4-5 and 7-8, there will be two nodes. Thus, the number of nodes and the number of edges will decrease by two, compared to the original graph. In the second graph, instead of a pair of nodes 4-5, there will be one node, and the 11th branch will not be, since branches 3 and 5 will become "hanging". In the third column, instead of a pair of nodes 7-8, there will be one node, and the 8th branch will not be, since branches 7 and 10 will become "hanging". In the second and third graphs, after applying step 1, the number of nodes will decrease by three, and the number of branches will decrease by 4 compared to the original graph. In the fourth graph, the graph removes branches 8 and 11, resulting in 4 hanging branches. Thus, after applying step 1, the number of nodes will decrease by 4, and the number of branches by 6 compared to the original graph.

If we calculate the number of possible spanning trees of the original graph by a known method, we get a number equal to 85. And the distribution of the number of spanning trees in the resulting four graphs will be as follows: 1 graph-48, 2 and 3 graphs - 16, the 4th graph has only 5 spanning graphs.

Similar calculations for the test 14-node scheme (IEEE) yield the following results: only one Union of nodes 10 and 11, with branch 18, is Possible. The number of trees with Union is 2182, without this branch 927. In sum, we get the known number 3909. It should be noted that, if on a simple laptop Intel Core I3-6006UCPU @ 2.00 GHZ 1.99 GHz And 4.00 GB of RAM to calculate the number of branches of the original graph 14-node scheme took 1438 milliseconds, the split for the first class took 432 milliseconds, and the second subclass 97 milliseconds.

The efficiency of this partition will be significantly manifested when considering schemes with more connected graphs. In particular, if we consider the test IEEE 30-node scheme with 41 branches, which has 7 824 000 spanning trees. If you consider that each tree has 29 branches, you need to remember the database of 453 792 000 at least two-digit numbers, and 900 times to search for the desired branches, to calculate the current distribution coefficients. Firstly, it will require large machine-time resources, and secondly, the rounding error will accumulate.

However, many trees have significant identical parts, respectively, the contribution of these parts in the calculation of current distribution coefficients is also the same, these contributions are simply summed. Therefore, the same parts can be placed outside the brackets, which significantly reduces the required number of calculations.

When splitting, parts of future trees are immediately allocated, so in the remaining parts, which are a graph with a smaller dimension, the number of search options is significantly reduced.

Note that for the specified IEEE test 30-node scheme, in which the number of possible spanning trees is 7 824 000, $k = 4$. That is, the set is divided into 16 classes. In the largest class there are 2,500,749 trees, that is 31.9% of the total number of graphs. In this case, the number of nodes of the resulting graph is 23, and the number of branches is 34. And in the most recent small class, the graph has 18 nodes and 25 branches, the number of spanning trees is 25,191 (0.3% of the total number of trees).

IV. Conclusion

1. The optimal algorithm search all possible spanning trees of a given graph based on the partitioning of many kinds of trees on disjoint classes, while additional branches of the spanning tree.

2. It is shown that such a partition significantly reduces the required number of calculations.

3. As a result of the proposed partition, the tasks are divided into several subtasks, each of which can be calculated in parallel and independently of each other.

4. Optimization efficiency is found to be essential for schemes with more filled connections.

References

[1] Christofides N. Graph Theory. Algorithmic approach. M .: Mir, 1978, - 432 p.20.

[2] Mayeda W. Graph Theory. Wiley- *Interscience*, New York, 1972.

[3] Geraskin O.T. Topological content of nodes and circuits of the electrical network and the calculation of their size with the help of a digital computer, Izvestia ANSSR. Energy and transport, 1966. №2. - pp. 59-70, (in Russian.)

[4] David Avis, Komei Fukuda. Reverse search for enumeration. *Discrete applied mathematics* 65, pp. 21-46, 1996.

[5] Akhmetbayev D.S., Aubakir D.A., Sarsikeyev Y.Zh., Bainiyazov B.A., Surkov M.A., Rozhkov V., Ansabekova G.N., Yerbolova A.S., Suleimenov A.T., Tokasheva M.S. Development of Topological Method for Calculating Current Distribution Coefficients, *Complex Power, Networks, Results in Physics*, vol. 7, pp. 1644–1649, 2017.

[6] A. R. Dzhandigulov, D. S. Akhmetbaev Finding all spanning graphs of a given graph. computer program, 2019.

[7] Akhmetbayev D., Dzhandigulov A. Development of algorithms for a new topological method for calculating current distribution coefficients in complex electrical networks, *Eurasian Journal of Mathematical and Computer Applications,* ISSN 2306–6172 Vol. 7, Issue 3, pp. 4–12, 2019.