Optimization of Sensor Network Topology Using Multiple Criteria

Pavol JURÍK, Peter SCHMIDT*, Jaroslav KULTAN

Department of Applied Informatics, Faculty of Economic Informatics,
University of Economics in Bratislava, Slovakia
pavol.jurik@euba.sk, peter.schmidt@euba.sk (**Corresponding author*), jaroslav.kultan@euba.sk

Abstract: A sensor network was created inside a building, as part of a project. The sensors monitored the room temperature, humidity, and CO_2. The location of the sensor nodes is fixed. A serious problem was to find out how to connect these sensors in order to ensure an optimal communication in the network. Another problem was the fact that the building was atypical, so the location of the sensors could not be applied to the grid. Two optimization criteria, namely the distance between the nodes and the quality of the working environment between them, limited the present research because in addition to reinforced concrete walls and foam concrete walls, other materials were also used in the building. The problem is described by a network graph and optimized by finding its spanning tree. A new algorithm was created for finding a multi-criteria spanning tree of a network graph taking into account the constraints for the weights of the edges. The algorithm has been programmed in C language and used to solve the task. The usage of this algorithm and the whole concept of finding a multi-criteria spanning tree of a sensor network graph is a contribution to sensor network topology optimization and it can be applied in similar projects as well.

Keywords: Sensor networks, Topology, Optimization, Graph theory, Spanning tree, Multiple criteria, Constraints.

1. Introduction

As part of the ongoing project, the task of this research was to find out indicators of air quality in the selected classrooms of a university. As there were more classrooms on different floors, only data acquisition could be considered, with the help of a sensor network. A sensor network consisting of sensor nodes that continuously monitored temperature, humidity and CO2 in the classrooms was created. The location of the sensors was given as each of them was powered directly from the existing power grid. Each sensor node was designed as a module that was plugged into an appropriate electrical outlet, because the wall panelling did not allow mounting of rails or other fastening material. From the architectural point of view the building could be characterized as atypical, because many rooms had non-standard floor plans and the sensors could not be applied to the grid, as many authors recommend (Liu et al., 2016). From the constructional point of view, it was a non-homogeneous building, too, because burnt bricks and plasterboard partitions have been used in addition to ferro-concrete walls and foam concrete walls.

The sensor node of this network consisted of a microcontroller board Arduino UNO R3 ATmega328P CH340 mini-USB to which 2 sensors were connected, namely temperature and humidity sensor DHT11 and CO_2 sensor MH-Z19B, as well as a communication module NRF24L01. NRF24L01 transmitter and receiver module uses the 2.4 GHz band and operates at transfer rates from 250 kbps to 2 Mbps. When applied outdoors and at lower transmission speeds, it can reach up to 100 meters. In a built-up environment, the range decreases significantly. The module might operate 125 different channels, enabling building a network, which consists of 125 independently operating modems in a simple place. Each channel might have up to 6 addresses, or each unit can communicate with up to 6 other units simultaneously (Rohidh et al., 2019). The operating voltage of the module is from 1.9 to 3.6 V, but the pins tolerate 5 V logic, so the module can be easily connected to Arduino without using any logic level converters. The sensor node was powered via the adapter. The data is transferred by hop to a special node called sink (Akyildiz et al., 2002). In total, 23 sensors were installed. Since the location of the sensor nodes was given in advance, the paths among the nodes had to be looked at, in order to ensure efficient wireless communication. Two criteria had to be taken into consideration: the distance among the nodes (a minimization criterion) and the quality of the environment among them (a maximization criterion).

With respect to the previous description, it was not a typical sensor network. Some differences should be highlighted. First of all, it was a sensor network where the sensors had a fixed location, and they were not battery powered, but mains-powered. In addition to it, the configuration of the sensor nodes and the neighbour search as well, were not automatic, unlike a typical sensor network. The present optimization approach is based on finding the optimal path among the nodes in terms of the reliability of data transmission. Transmission reliability is strongly dependent on node distance, quality of working environment and transmission speed. However, there was no need to consider an increased energy demand related to the node to transmit and receive data from neighbouring nodes. Attempts have been made to find a solution where the data transfer is relatively fast and the error rate is low.

Section 2 describes the minimum spanning tree problem, which was a basis for the research done. Section 3 presents a new solution for the multi-criteria minimum spanning tree problem and in section 4 there is an example of its use. Section 5 presents the conclusions and prospects for the development of the proposed research.

2. The Minimum Spanning Tree Problem

The problem of finding a way to interconnect a set of nodes on a network graph with a set of edges so that there are no cycles and no node is isolated, is called a spanning tree problem. According to Levitin (2007): "A spanning tree of a connected graph consists of a connected acyclic subgraph (i.e., a tree) that contains all the vertices (i.e., nodes) of graph." Every spanning tree of a network graph $G = (V, E)$, where V is a set of vertices (i. e. nodes) and E is a set of edges, consists of $|V|$ - 1 edges. For example, if the network graph G consists of 7 nodes, then the number of edges is 6 for each of its spanning trees.

Some numerical values might also be added to the edges of the network graph, which may represent, for example, mutual distances between two nodes, costs that are needed to create a direct connection between two nodes or transmission capacities of a path between two nodes.

A network graph with weighted edges is called "weighted network graph". If this graph is connected (i.e., it does not contain any isolated node), then it is possible to find a spanning tree with the minimum sum of weights for all included edges. This type of spanning tree is called "minimum spanning tree". Levitin defines a minimum spanning tree of a weighted connected graph as "its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges" (Levitin, 2007).

A detailed analysis of the minimum spanning tree problem history was done by Graham and Hell (1985), so the present paper will not discuss this analysis (Graham & Hell, 1985). There are a few algorithms that are suitable for solving the minimum spanning tree problem. Borůvka's algorithm was the first algorithm for spanning trees; however, it is applicable only under a special condition: every edge of the network graph has to have a unique weight (Borůvka, 1926). A better algorithm for solving the minimum spanning tree problem is Kruskal's algorithm because it always yields an optimal solution. The algorithm is building up a spanning tree as an expanding sequence of subgraphs, which are always acyclic but are not connected via intermediate stages of the algorithm (Kruskal, 1956). Another algorithm that guarantees the finding of an optimal solution of the minimum spanning tree problem of a connected graph is Prim's algorithm. This algorithm is building up the minimum spanning tree through a sequence of expanding subtrees. The initial subtree in such a sequence consists of a single vertex only, which is arbitrarily selected from the set V of the graph's vertices. At each iteration, the current tree is expanded in a greedy way so that the nearest vertex, which is not yet in the tree is attached to it (Prim, 1957).

3. The Multi-Criteria Spanning Tree Problem – Description of the Proposed Algorithm

If there is only a single optimization criterion, algorithms for finding a minimum spanning tree can be used. In the case of the project described in section 1, two different optimization criteria need to be applied at the same time. There are multiple algorithms that deal with the bi-criteria spanning tree problem (Clímaco & Pascoal, 2012; Ehrgott, 2005). One can cite also evolutionary algorithms (Moradkhan & Browne, 2006), genetic algorithms (Han & Wang, 2005; Sanger & Agrawal, 2010; Zhou & Gen, 1999), local search techniques (Andersen, Jörnsten & Lind, 1996) or ant colony techniques (Cardoso, 2006; Li et al., 2013). As far as it is known, there are only 3 exact algorithms for the bi-criteria spanning tree problem – an algorithm based on dynamic programming (Di Puglia Pugliese et al., 2015), exhaustive search (Ramos et al., 1998) and two-phase algorithm (Steiner & Radzik, 2008). Furthermore, only a few solutions for a multi-criteria spanning three problem with more than two criteria are known (Di Puglia Pugliese et al., 2015; Neumann & Witt, 2010).

This paper proposes a new algorithm, which is very simple because it transforms a multi-criteria spanning tree problem with constraints for the weights of the edges and any number of criteria into a single-criterion problem without any constraints. The constraints on the weights of the edges mean imply that, for example, the transmission capacity or the reliability of any edge in the graph or a subset of edges should be higher than or equal to the required constant. Or for example, the costs that are needed to create a direct connection between two nodes (i.e., sensors in a sensor network) should be lower than or equal to the specific constant. However, there can be also interval constraints (for example, a combination of the "higher or equal" and the "lower or equal" type of constraint) in order to achieve a balanced network.

The criteria that can be considered during the optimization of sensor network topology are, for example, the minimization of sum of distances among the nodes, minimization of total creation costs, maximization of transmission capacities or maximization of reliability of the individual edges. However, other criteria may be considered as well. Each criterion shall have its weight regarding its importance so there is an exact importance ratio. For example, some organizations may prefer transmission capacities over creation costs in a 2:1 ratio, while others may prefer creation costs over transmission capacities in 3:1 ratio.

The input data which are needed to use the proposed algorithm are the following:

- the number of the criteria,
- the importance ratio of the criteria (i.e., the weight of each criterion),
- for each criterion one needs to know if it is a minimization or a maximization criterion,
- evaluation matrices, where each evaluation matrix contains the weights of the edges corresponding to a single criterion.

Thus, the number of the evaluation matrices is equal to the number of criteria. For example, if an organization wishes to consider three different criteria, then it is necessary to have three different evaluation matrices because each of them is devoted to a single criterion only. All these evaluation matrices are square, i.e., the number of their rows is equal to the number of their columns and that is equal to the number of nodes (i.e., $|V|$) of the graph $G = (V, E)$. For example, if $|V| = 10$, then each matrix will have 10 rows and 10 columns. At the crossing points of the rows and the columns of the matrix, the numeric values represent the weights of the individual edges corresponding to a selected criterion. These values need to be given in specific measurement units corresponding to the criterion. For example, appropriate measurement units for creation costs are EUR or USD and appropriate units for measuring transmission capacities are bits per second (i.e., bps), kilobits per second (i.e., kbps), etc. On the main diagonal of these matrices there are no numeric values because it is not logical to connect a node of a sensor network to itself.

As it has been mentioned earlier, the multi-criteria spanning tree problem may be extended by specific constraints (i.e., limitations) on the edges of the constructed spanning tree. If such constraints are established, then it is necessary to identify all the edges that do not meet any of these constraints for any criterion. After the identification of all inappropriate edges, a subgraph of graph $G = (V, E)$ is created, which may be denoted as $G' = (V', E')$. This subgraph must contain only the edges of G that meet all the constraints. After the creation of the subgraph G', it is necessary to verify if it is still possible to construct the required multi-criteria spanning tree of G using the edges in E'. If there is no acceptable spanning tree meeting all the required criteria, then there is no optimal solution for the task. Verification of the existence of at least one feasible solution can be done through an analysis of the subgraph G'. If this subgraph contains all the vertices of the graph G and it is connected at the same time, then it is possible to find at least one feasible spanning tree of G for sure. These two conditions need to be met at the same time. A spanning tree of G' is a connected acyclic subgraph of G' containing all vertices of G and logically, it is not possible to find a connected subgraph of G' containing all its vertices if G' itself is not connected. It is also obvious that if the subgraph G' does not contain all the vertices of G, then it is not possible to create a feasible spanning tree of G using the edges of G'.

After the verification that the subgraph G' is connected and contains all the vertices of the original graph G, the optimal solution can be found. The next problem that should be solved is the mutual incomparability of the evaluation matrices due to different measurement units used in each matrix. This problem may be solved by normalization of the matrices. Normalization annuls the relation of the matrices to specific measurement units and makes them mutually comparable. It can be done using formula (3.1). With the help of this formula a normalized matrix **C'**= $\{c'_{ij}\}$ can be calculated, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$. It means that the original values of the matrix **C** can be transformed into dimensionless numbers at a closed interval [0;1] and these values can be stored in matrix **C'**.

If some of the criteria that are set for the multi-criteria spanning tree are maximization criteria, then it is necessary to convert all these maximization criteria into an opposite form, which is intended for minimization. However, this conversion is done only from a computational point of view. It means that a maximization criterion (the higher the value, the better) is still a maximization criterion, however, from the computational point of view it appears to be a minimization criterion (the lower the value, the better) after the conversion. The conversion of a normalized matrix **C'** = $\{c'_{ij}\}$, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$ to a normalized matrix **C''** = $\{c''_{ij}\}$, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$ corresponding to the opposite type of extreme can be performed using formula (3.3).

After the normalization of all the matrices and the conversion of all the matrices corresponding to maximization criterion to an opposite form intended for minimization, a final aggregation matrix **K** = $\{k_{ij}\}$ can be created, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$. This matrix is used with the intention to find the final spanning tree of *G,* taking into account all the required criteria and their importance ratio. The elements of this matrix represent a weighted sum of corresponding elements in all the normalized matrices (regarding minimization criteria) and all the transformed matrices that have already been normalized (regarding maximization criteria) taking into account the importance ratio of all the criteria. Thus, if there are, for example, three optimization criteria, then the matrix **K** is calculated using three evaluation matrices and the importance ratio (i.e., the weights) of these criteria. For example, if there are two minimization criteria and the third criterion is a maximization criterion, then the values in all three matrices are normalized using formula (3.1) and after that the normalized matrix for the third criterion needs to be transformed into an opposite form intended for minimization using formula (3.3). After that the values of the aggregation matrix **K** can be calculated using the two normalized matrices corresponding to the minimization criteria and the transformed matrix that was normalized before and corresponds to the maximization criterion. The elements of the matrix **K** can be calculated using formula (3.4 or 3.5). These are different ways of writing the same formula.

After the creation of the aggregation matrix **K**, it is possible to continue the calculation using standard algorithms for the minimum spanning tree problem, for example, Kruskal's algorithm or Prim's algorithm. Thus, the aim of the whole procedure described above was to convert a multi-criteria spanning tree problem considering the importance ratio of the criteria and the constraints on the weights of the edges into a standard single-criterion minimization spanning tree problem without any constraints. However, to interpret the final result (i.e., the final spanning tree) in a correct way, it is necessary to use the original weights of all edges regarding all required criteria, not the normalized or the transformed weights.

The whole algorithm described above consists of a sequence of steps as follows:

1. Identify all the edges that do not meet any constraint;
2. Create *G*' = (*V*', *E*') – a subgraph of the original network graph *G* consisting of all the edges that meet all the constraints;
3. $E_T \leftarrow \emptyset$ (E_T is a set of edges that has already been included in the constructed spanning tree);
4. **If** *G'* doesn't contain all the nodes of *G*, **then** the output is E_T (the optimal solution of the task doesn't exist);

   else

   if *G'* is not a connected graph (i.e., if an arbitrary vertex of *G'* which is not reachable from all of the other vertices of *G'* using the edges in *E'* can be selected), **then** the output is E_T (the optimal solution of the task doesn't exist);

   else go to step 5.

5. For every evaluation matrix **C** = $\{c_{ij}\}$, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$ create a normalized evaluation matrix **C'** = $\{c'_{ij}\}$, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$ using the following formula (3.1):

$$c_{ij}' = \frac{c_{ij} - c^{\min}}{c^{\max} - c^{\min}} \tag{3.1}$$

$$c^{\min} = \min_{ij}\{c_{ij}\}, c^{\max} = \max_{ij}\{c_{ij}\} \tag{3.2}$$

6. For every normalized matrix **C'** that corresponds to a maximization criterion, create a transformed matrix **C"** = $\{c''_{ij}\}$, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$. This matrix represents, from the computational point of view, a conversion of the maximization criterion into an opposite form intended for minimization using the following formula (3.3):

$$c_{ij}'' = 1 - c_{ij}' \tag{3.3}$$

7. Create an aggregation matrix **K** = $\{k_{ij}\}$, where $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$. This matrix aggregates all the normalized matrices (for all the minimization criteria) and all the transformed matrices that were normalized before (for all the maximization criteria) using the following formula (3.4 or 3.5, which are different ways of writing the same formula):

$$k_{ij} = v_1 * k_{1ij} + v_2 * k_{2ij} + \dots + v_r * k_{rij} \tag{3.4}$$

$$K = v_1 * K_1 + v_2 * K_2 + \dots + v_r * K_r \tag{3.5}$$

where k_{ij} are the values of the aggregation matrix **K**; r is the number of the criteria; $v_1, v_2, \dots, v_r$ are the weights representing the importance ratio of the individual criteria and $k_{1ij}, k_{2ij}, \dots, k_{rij}$ for $i = 1,\dots, |V|$ and $j = 1,\dots, |V|$ represent the elements of the matrices $\mathbf{K_1}$, $\mathbf{K_2}$, ..., $\mathbf{K_r}$ that are necessary to calculate the elements of the aggregation matrix **K.**

For example, let's consider that there are two criteria. The first criterion is a minimization criterion and the edge weights corresponding to this criterion are given in matrix **C**. The values in matrix **C** are normalized using formula (3.1) and a normalized matrix **C'** is obtained. In this research, matrix **C'** represents matrix $\mathbf{K_1}$ (i. e., **C'** is denoted as $\mathbf{K_1}$). Let's consider that the second criterion is a maximization one and the edge weights corresponding to this criterion are given in matrix **D**. The values in matrix **D** are normalized using formula (3.1) and a normalized matrix **D'** is obtained. However, since it is a maximization criterion formula (3.3) must also be applied and matrix **D"** must be calculated using the values in matrix **D'**. In this research, matrix **D"** represents matrix $\mathbf{K_2}$ (i.e., **D"** is denoted as $\mathbf{K_2}$). Since only two criteria are considered in this case, matrices $\mathbf{K_1}$ and $\mathbf{K_2}$ are the only ones needed to calculate the values of the aggregation matrix **K** using formula (3.4 or 3.5). Thus, if the first criterion is a minimization criterion, formula (3.1) is used for the corresponding matrix and the resulting matrix is denoted as matrix $\mathbf{K_1}$. If it is a maximization criterion, formula (3.1) is firstly used and then formula (3.3) and the resulting matrix is denoted as matrix $\mathbf{K_1}$. If the second criterion is a minimization criterion, formula (3.1) is used for the corresponding matrix and the resulting matrix is denoted as matrix $\mathbf{K_2}$. If it is a maximization criterion, formula (3.1) is firstly used and then formula (3.3) and the resulting matrix is denoted as matrix $\mathbf{K_2}$. This principle shall be applied for each of the *r* criteria. The matrices $\mathbf{K_1}$, $\mathbf{K_2}$, ..., $\mathbf{K_r}$ that are needed to calculate the aggregation matrix **K** are obtained using formula (3.4 or 3.5).

8. Calculate the minimum spanning tree of *G'* using the edge weights in matrix **K** with the help of Kruskal's algorithm or Prim's algorithm. If the Kruskal's algorithm is selected, then the following steps are required:

```
a) Sort the edges in E' in a non-
   descending order according to their
   weight in matrix K;
b) ecounter ← 0 (ecounter is a variable
   indicating the number of edges in $E_T$);
c) k ← 0 (k is a variable representing an
   index of an edge on the sorted list of
   edges in E' that is currently considered
   to be or not to be included in the set $E_T$);
d) Until (ecounter < |V| - 1) {
   k ← k + 1;
   if $E_T \cup \{e_k\}$ is acyclic, then
   ($e_k$ is an edge on the sorted list of edges
   and k is the variable described in step
   8c)
   {
   $E_T \leftarrow E_T \cup \{e_k\}$;
     ecounter ← ecounter + 1;
     } };
e) The output is $E_T$.
```

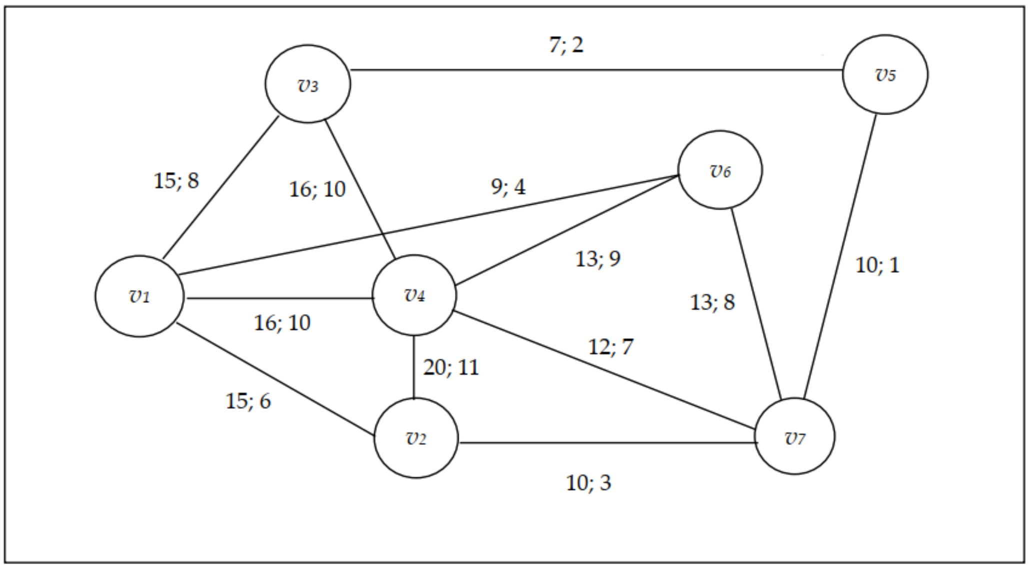

Figure 1. Graph $G = (V, E)$

4. How to Use the Algorithm

Let's consider a situation (displayed in Figure 1) in which the task is to connect 7 network nodes (i.e., sensors) with each other in such a way that the final result shall be a connected acyclic network graph that will contain all the nodes while taking into account two criteria – minimization of distances among the nodes and maximization of quality of working environment among the nodes. The importance ratio of these criteria is set to be 1:2 (i.e., the maximization of working environment quality is twice as important as the minimization of distances between the nodes in this particular situation). The weights of the edges representing mutual distances between the nodes are presented in Table 1 (matrix **D**).

Table 1. Weights of edges representing the distance in meters (matrix **D**)

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	-	15	15	16	-	9	-
v_2	15	-	-	20	-	-	10
v_3	15	-	-	16	7	-	-
v_4	16	20	16	-	-	13	12
v_5	-	-	7	-	-	-	10
v_6	9	-	-	13	-	-	13
v_7	-	10	-	12	10	13	-

The values of transfer rates were obtained by experimental measurement. Since the transfer rate was mainly influenced by the thickness of the walls, the walls were distinguished according to materials. Based on the wall thickness, intervals in increments of 150 kbps have been created. For greater clarity, dimensionless weight numbers to each interval were assigned. These weights are presented in Table 2.

Table 2. Dependency of transfer rate on environment quality

Weight	**Max. transfer rates (kbps)**	**Quality of the environment**
1	250	ferro-concrete
2	400	ferro-concrete
3	550	ferro-concrete
4	700	burnt bricks
5	850	burnt bricks
6	1000	burnt bricks
7	1150	foam concrete
8	1300	foam concrete
9	1450	plasterboard
10	1600	plasterboard
11	1750	no obstacle up to 20 m
12	1900	no obstacle up to 10 m

The measured values have been replaced with the corresponding weight value according to the interval. These weights are introduced into matrix **E** which is shown in Table 3. A constraint was also established, namely that the transfer rate of every edge included in the final solution should be higher than 250 kbps. This means that every edge with weight equal to 1 in terms of quality of working environment should be considered to be inappropriate and should not be included in the final spanning tree.

Table 3. Weights of edges representing the quality of working environment (matrix **E**)

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	-	6	8	10	-	4	-
v_2	6	-	-	11	-	-	3
v_3	8	-	-	10	2	-	-
v_4	10	11	10	-	-	9	7
v_5	-	-	2	-	-	-	1
v_6	4	-	-	9	-	-	8
v_7	-	3	-	7	1	8	-

The calculation using the algorithm described in section 3 goes as follows:

1. All the edges that do not meet the constraint regarding transfer rates are identified. The only edge that does not meet this constraint is the edge (v_5, v_7), because its weight in terms of the quality of working environment is equal to 1.

2. $G' = (V', E')$, a subgraph of the original graph G, is created, where $V' = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and $E' = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_1, v_6), (v_2, v_4), (v_2, v_7), (v_3, v_4), (v_3, v_5), (v_4, v_6), (v_4, v_7), (v_6, v_7)\}$.

3. $E_T \leftarrow \emptyset$;

4. It is apparent that G' contains all the vertices of G. The connectedness of G' must be verified. If one arbitrary vertex is selected, it has to be reachable from all the other vertices of G' using the edges in E'. For example, if vertex v_1 is selected, it is observed that it is connected to:

   - v_2 using the edge (v_1, v_2),
   - v_3 using the edge (v_1, v_3),
   - v_4 using the edge (v_1, v_4),
   - v_5 using the edges (v_1, v_3), (v_3, v_5),
   - v_6 using the edge (v_1, v_6),
   - v_7 using the edges (v_1, v_2), (v_2, v_7).

   Thus, vertex v_1 is connected to all the other vertices and we can go to step 5.

5. For matrix **D** = $\{(v_1, v_2) = 15, (v_1, v_3) = 15, (v_1, v_4) = 16, (v_1, v_6) = 9, (v_2, v_4) = 20, (v_2, v_7) = 10, (v_3, v_4) = 16, (v_3, v_5) = 7, (v_4, v_6) = 13, (v_4, v_7) = 12, (v_6, v_7) = 13\}$ (described in Table 1) a normalized matrix **D’** is calculated using the following formula (4.1):

$$d_{ij}' = \frac{d_{ij} - d^{\min}}{d^{\max} - d^{\min}} \qquad (4.1)$$

   where $i = 1, \dots, 7$ and $j = 1, \dots, 7$

$$d^{\min} = \min_{ij}\{d_{ij}\}, d^{\max} = \max_{ij}\{d_{ij}\} \qquad (4.2)$$

   Matrix **D’** is described in Table 4. The lowest value of the entries of matrix **D** is $\min_{ij}\{d_{ij}\} = 7$ and the highest value of the entries of matrix **D** is $\max_{ij}\{d_{ij}\} = 20$. The elements of a normalized matrix E’ corresponding to matrix **E** are also calculated. Matrix **E’** is depicted in Table 5.

Table 4. *Matrix* **D’**

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	-	0.615	0.615	0.692	-	0.154	-
v_2	0.615	-	-	1	-	-	0.231
v_3	0.615	-	-	0.692	0	-	-
v_4	0.692	1	0.692	-	-	0.462	0.385
v_5	-	-	0	-	-	-	-
v_6	0.154	-	-	0.462	-	-	0.462
v_7	-	0.231	-	0.385	-	0.462	-

Table 5. *Matrix* **E’**

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	-	0.444	0.667	0.889	-	0.222	-
v_2	0.444	-	-	1	-	-	0.111
v_3	0.667	-	-	0.889	0	-	-
v_4	0.889	1	0.889	-	-	0.778	0.556
v_5	-	-	0	-	-	-	-
v_6	0.222	-	-	0.778	-	-	0.667
v_7	-	0.111	-	0.556	-	0.667	-

6. The only maximization criterion is the maximization of the quality of the working environment. Thus, from the computational point of view, this criterion is transformed into an opposite form intended for minimization.

The values of a transformed matrix **E**" (written in Table 6) have to be calculated using the values in the normalized matrix **E**' and the following formula (4.3):

$$e_{ij}'' = 1 - e_{ij}' \quad (4.3)$$

where $i = 1, \ldots, 7$ and $j = 1, \ldots, 7$

Table 6. *Matrix* **E"**

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	-	0.556	0.333	0.111	-	0.778	-
v_2	0.556	-	-	0	-	-	0.889
v_3	0.333	-	-	0.111	1	-	-
v_4	0.111	0	0.111	-	-	0.778	0.444
v_5	-	-	1	-	-	-	-
v_6	0.778	-	-	0.778	-	-	0.333
v_7	-	0.889	-	0.444	-	0.333	-

7. We create an aggregation matrix **K** = $\{k_{ij}\}$, where $i = 1,\ldots, 7$ and $j = 1,\ldots, 7$ that aggregates the normalized matrix **D'** and the transformed matrix **E"** using formula (4.4):

$$k_{ij} = v_1 * k_{1ij} + v_2 * k_{2ij} + \ldots + v_r * k_{rij} \quad (4.4)$$

where $i, j = 1, \ldots, 7$, $r = 2$, $v_1 = 1$, $v_2 = 2$

In the present case, it is $k_{ij} = 1*\mathbf{K_1} + 2*\mathbf{K_2}$ for $i, j = 1, \ldots, 7$. After the substitution of matrices $\mathbf{K_1}$ and $\mathbf{K_2}$ by real matrices **D'** and **E"** formulas (4.5) and (4.6) are obtained:

$$k_{ij} = 1 * d_{ij}' + 2 * e_{ij}'' \quad (4.5)$$

$$K = 1 * D' + 2 * E'' \quad (4.6)$$

The calculated values of matrix **K** are displayed in Table 7.

Table 7. *Matrix* **K**

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	-	1.556	1.333	1.111	-	1.778	-
v_2	1.556	-	-	1	-	-	1.889
v_3	1.333	-	-	1.111	2	-	-
v_4	1.111	1	1.111	-	-	2.018	1.444
v_5	-	-	2	-	-	-	-
v_6	1.778	-	-	2.018	-	-	1.333
v_7	-	1.889	-	1.444	-	1.333	-

8. The minimum spanning tree of *G'* is calculated using the weights in the aggregation matrix **K** by help of Kruskal's algorithm:

a) The edges in **E'** are sorted in a non-descending order according to their weight in matrix **K**:
$e_1 = (v_2, v_4) = 1$, $e_2 = (v_1, v_4) = 1.111$, $e_3 = (v_3, v_4) = 1.111$, $e_4 = (v_1, v_3) = 1.333$,
$e_5 = (v_6, v_7) = 1.333$, $e_6 = (v_4, v_7) = 1.444$,
$e_7 = (v_1, v_2) = 1.556$, $e_8 = (v_1, v_6) = 1.778$,
$e_9 = (v_2, v_7) = 1.889$, $e_{10} = (v_3, v_5) = 2$, $e_{11} = (v_4, v_6) = 2{,}018$.

b) *ecounter* ← 0;

c) *k* ← 0;

d) 1st iteration: 0 < (7 - 1) => The following cycle is entered:
$k \leftarrow 1$; $E_T \cup \{e_1\}$ is acyclic => $E_T \leftarrow E_T \cup e_1$; *ecounter* ← 1;
2nd iteration: 1 < (7 - 1) => The following cycle is entered:
$k \leftarrow 2$; $E_T \cup \{e_2\}$ is acyclic => $E_T \leftarrow E_T \cup e_2$; *ecounter* ← 2;
3rd iteration: 2 < (7 - 1) => The following cycle is entered:
$k \leftarrow 3$; $E_T \cup \{e_3\}$ is acyclic => $E_T \leftarrow E_T \cup e_3$; *ecounter* ← 3;
4th iteration: 3 < (7 - 1) => The following cycle is entered:
$k \leftarrow 4$; $E_T \cup \{e_4\}$ is cyclic.
5th iteration: 3 < (7 - 1) => The following cycle is entered:
$k \leftarrow 5$; $E_T \cup \{e_5\}$ is acyclic => $E_T \leftarrow E_T \cup e_5$; *ecounter* ← 4;
6th iteration: 4 < (7 - 1) => The following cycle is entered:
$k \leftarrow 6$; $E_T \cup \{e_6\}$ is acyclic=> $E_T \leftarrow E_T \cup e_6$; *ecounter* ← 5;
7th iteration: 5 < (7 - 1) => The following cycle is entered:
$k \leftarrow 7$; $E_T \cup \{e_7\}$ is cyclic.
8th. iteration: 5 < (7 - 1) => The following cycle is entered:
$k \leftarrow 8$; $E_T \cup \{e_8\}$ is cyclic.
9th iteration: 5 < (7 - 1) => The following cycle is entered:
$k \leftarrow 9$; $E_T \cup \{e_9\}$ is cyclic.
10th iteration: 5 < (7 - 1) => The following cycle is entered:
$k \leftarrow 10$; $E_T \cup \{e_{10}\}$ is acyclic => $E_T \leftarrow E_T \cup e_{10}$; *ecounter* ← 6;
11th iteration: 6 < (7 - 1) => This is not true, so no cycle is entered anymore.

e) The output is E_T;

Thus, the final multi-criteria spanning tree which is depicted in Figure 2 consists of the following edges with the following weights: $(v_2, v_4) = 20; 11$, $(v_1, v_4) = 16; 10$, $(v_3, v_4) = 16; 10$, $(v_6, v_7) = 13; 8$, $(v_4, v_7) = 12; 7$, $(v_3, v_5) = 7; 2$. The first weight of an edge is related to the mutual distance of the nodes in meters. The second weight is related to the quality of working environment and it is given in special weights according to Table 2.

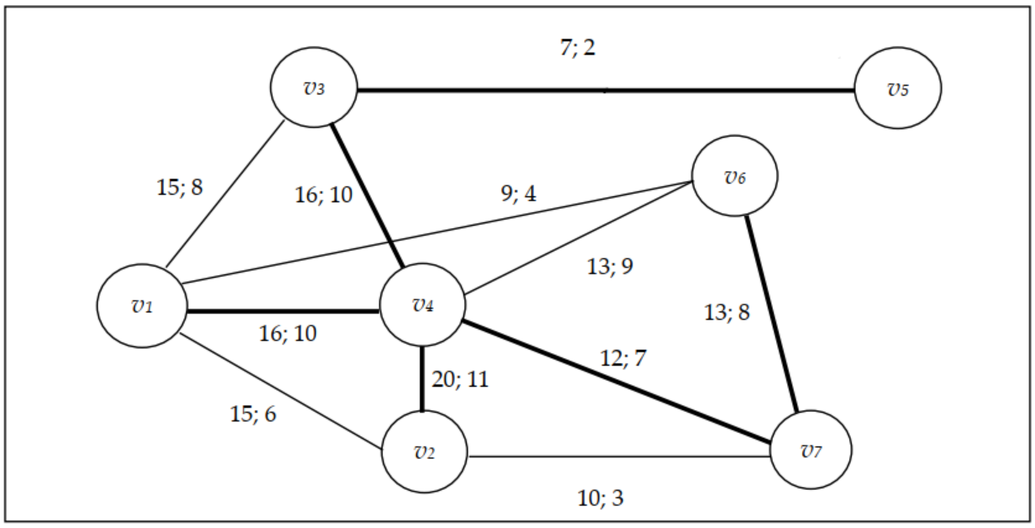

Figure 2. The multi-criteria spanning tree of the graph $G = (V, E)$

5. Conclusion

A simple algorithm, which is able to find a multi-criteria spanning tree of a network graph, was created, taking into account also the constraints on the weights of the edges if these constraints are needed. In smaller tasks the whole computation can be done by hand, so there is no need to use genetic algorithms, ant colonies and other complicated approximation techniques. It is hoped that this algorithm and the whole approach to sensor network topology optimization using multi-criteria spanning trees is a piece of contribution to sensor networking theory as well as to the field of operational research and network analysis. It may help to carry out many useful experiments and projects in the future.

Acknowledgements

This work was supported by a project KEGA No. 019EU-4/2020 entitled “Support of distance education through a virtual department” by the Ministry of Education, Science, Research and Sport of the Slovak Republic.

REFERENCES

Akyildiz, I. F., Weilian Su, Sankarasubramaniam, Y & Cayirci, E. (2002). A survey on sensor networks, *IEEE Communications Magazine*, *40*(8), 102-114. DOI: 10.1109/MCOM.2002.1024422

Andersen, K. A., Jörnsten, K. & Lind, M. (1996). On bicriterion minimal spanning trees: An approximation, *Computers & Operations Research*, *23*(12), 1171–1182.

Borůvka, O. (1926). *O jistém problému minimálním* (*On a certain minimal problem*). Práce Moravské Přírodovědecké společnosti v Brně, *3*, 37-58.

Cardoso, P. J. S. (2006). *Ant Colony Algorithms for Multiple Objective Combinatorial Optimization: Applications to the Minimum Spanning Trees Problem*. PhD thesis, University of Seville.

Clímaco, J. & Pascoal, M. (2012). Multicriteria path and tree problems: discussion on exact algorithms and applications, *International Transactions in Operational Research*, *19*(1-2), 63-98.

Di Puglia Pugliese, L., Guerriero, F. & Santos, J. L. (2015). Dynamic programming for spanning tree problems: application to the multi-objective case, *Optimization Letters*, *9*(3), 437-450.

Ehrgott, M. (2005). *Multicriteria optimization*. Springer-Verlag, Berlin, Heidelberg.

Graham, R. L. & Hell, P. (1985). On the History of the Minimum Spanning Tree Problem, *Annals of the History of Computing*, *7*(1), 43-57.

Han, L. & Wang, Y. (2005). A Novel Genetic Algorithm for Multi-criteria Minimum Spanning Tree Problem, *Computational Intelligence and Security*, 297-302.

Kruskal, J. B. (1956). *On the shortest spanning tree problem of a graph and the travelling salesman problem*. In *Proceedings of the American Mathematical Society*, *7*(1), (pp. 48-50).

Levitin, A. (2007). *Introduction to The Design and Analysis of Algorithms*, 316-344.United States of America: Pearson Education, Inc.

Li, Y., Zou, C. Y., Zhang, S. & Vai, M. I. (2013). Research on multi-objective minimum spanning tree algorithm based on ant algorithm, *Research Journal of Applied Sciences, Engineering and Technology*, *5*(21), 5051-5056.

Liu, B.-H., Nguyen, N.-T., Pham, V.-T. & Wang, V.-S. (2016). Constrained node-weighted Steiner tree based algorithms for constructing a wireless sensor network to cover maximum weighted critical square grids, *Computer Communications*, *81*, 52-60.

Moradkhan, M. D. & Browne, W. N. (2006). A knowledge-based evolution strategy for the multiobjective minimum spanning tree problem. In *2006 IEEE International Conference on Evolutionary Computation* (pp. 1391-1398).

Neumann, F. & Witt, C. (2010). Multi-objective Minimum Spanning Trees, *Bioinspired Computation in Combinatorial Optimization*, 149-159. Natural Computing Series. Springer, Berlin, Heidelberg.

Prim, R. C. (1957) Shortest connection networks and some generalizations, *Bell System Technical Journal, 36*, 1389-1401.

Ramos, R., Alonso, S., Sicilia, J. & Gonzalez, C. (1998). The problem of the optimal biobjective spanning tree, *European Journal of Operational Research*, *111*(3), 617-628.

Rohidh, V., Ranjith, G., Revathi, G. & Balaji, G. 2019) Centralised Status Alert System for Industrial Machines, *International Research Journal of Engineering and Technology, 6*(3), 3011-3015.

Sanger, A. K. & Agrawal, A. K. (2010). Comparison of tree encoding schemes for biobjective minimum spanning tree problem. In *2010 2nd IEEE International Conference on Information and Financial Engineering* (pp. 233-236).

Steiner, S. & Radzik, T. (2008). Computing all efficient solutions of the biobjective minimum spanning tree problem, *Computers & Operations Research*, *35*(1), 198–211.

Zhou, G. & Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem, *European Journal of Operational Research*, *114*(1), 141-152.