CHAPTER 4

MAXIMUM SPANNING TREE MODELING

4.1 INTRODUCTION

A spanning tree of a graph is a subgraph that contains all the vertices and is generally represented as a tree. A graph may have many spanning trees. Spanning trees find their applications in laying of telephone cables from the telephone office in which one vertex is designated as the starting point of laying cable. The remaining nodes represent the houses to which the cable lines have to be installed. There may be more than one cabling topology from the telephone office as there may be more than one for a given graph.

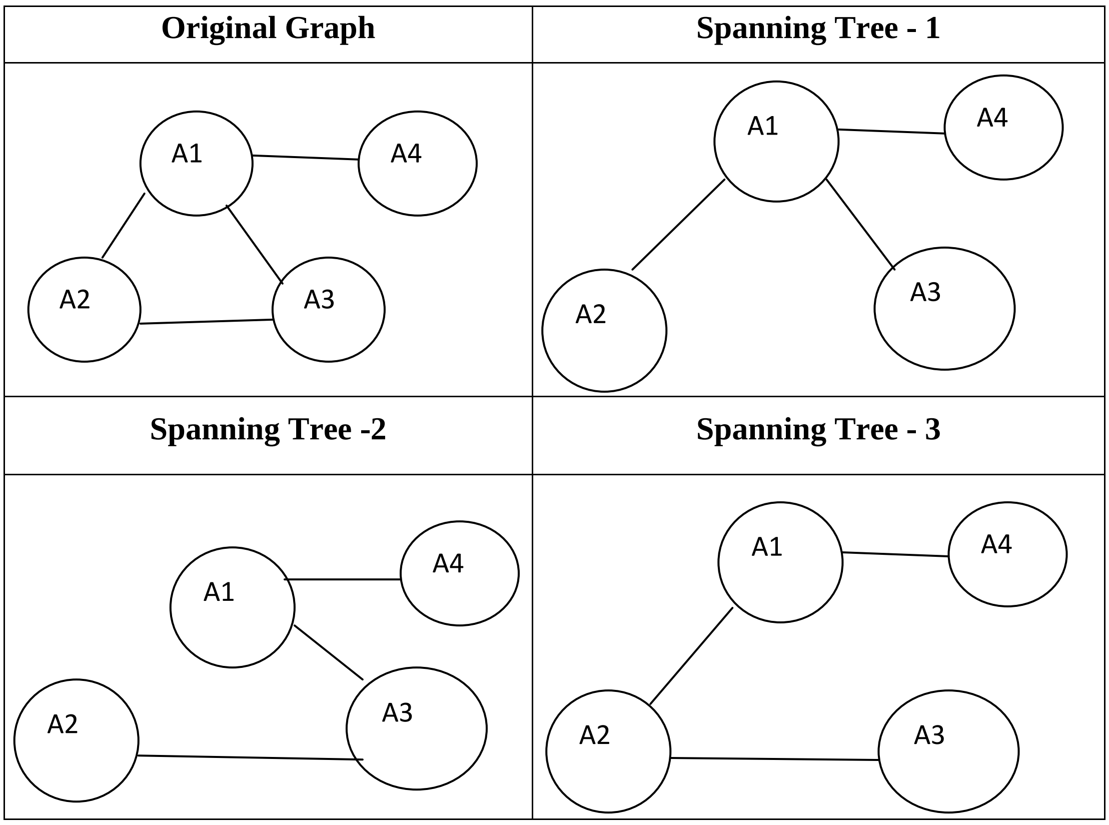


Figure 4.1 Different Spanning Trees for a graph with 4 vertices and 4 edges

Figure 4.1 shows the 4 different spanning trees obtained for a graph with 4 vertices and 4 edges.

In a connected undirected graph with weight associated with each edge, the cost of the spanning tree is the sum of the cost of the edges in the spanning tree . Practically, in Laying of cables scenario, the cable cost should be minimized by utilization of only limited cable for laying lines. The minimum cost of the spanning tree is the lowest cost obtained among the costs computed for all spanning trees obtained from the graph. If s1,s2,…sk are the spanning trees associated with a given graph G and c1, c2,….ck are the costs associated with the spanning tree's then minimum spanning tree sj =spanning tree corresponding to (min(c1,…ck)). If the edge cost in the above telephone cabling problem is the distance , the minimum cost in laying cable in above scenario is obtained by choosing the minimum spanning tree which results in minimum usage of cable to connect telephone office and houses.

On the contrary, the maximum cost of the spanning tree is the highest cost obtained among the costs computed for all spanning trees obtained from the graph. If s1,s2,…sk are the spanning trees associated with a given graph G and c1, c2,….ck are the costs associated with the spanning tree's then maximum spanning tree sj =spanning tree corresponding to (max(c1,…ck)). Consider the case in which the edges denote the noiseless effectiveness of voice at homes which are denoted by vertices. In this example where the noiselessness is given a greater weightage effectiveness should be maximized in which maximum spanning tree is used.

The Traditional algorithms designed for Minimum spanning tree in which minimum cost is chosen at each stage is reversed to get Maximum spanning Tree in which the maximum cost is chosen at each stage.

4.2 BASIC DEFINITIONS

Graph

A graph is a collection of nodes called vertices, and the connections between them, called edges. When the edges in a graph have a direction, the graph is called a directed graph or digraph, and the edges are called directed edges or arcs.

A graph can also be stated as an ordered pair, G = <V, A>, where V is the set of vertices, and A, the set of arcs, is itself a set of ordered pairs of vertices.

Graph theory is used in many applications like road networks, design of routers in computer networks, burglar alarm design , time tabling problems etc.

Tree

A tree is a finite set of one or more nodes such that: There is a specially designated node called the root. The remaining nodes are partitioned into n>=0 disjoint sets T1, ..., Tn, where each of these sets is a tree. We call T1, ..., Tn the subtrees of the root.

4.3 MAXIMUM SPANNING TREE – DIFFERENT APPROACHES

An approach to combine the two basic problems Knapsack and minimum spanning tree which are the basic problems of Resource Management Techniques and computer science were done. Knapsack problem of fixed capacity and undirected graph are considered. The problem solved in this approach is filling the knapsack problem with a feasible spanning tree such that the profit is maximized. (Yamada et al, 2005) (Ahuja et al, 1993) (Busacker and Saaty, 1965) (Kruskal 1956) (Prim, 1957)

Minimum spanning tree is used to identify clusters with irregular boundaries. Two minimum spanning tree cluster algorithm was proposed in which the first algorithm uses a divisive approach and the second algorithm uses agglomerative approach to find informative meta similarity clusters. One algorithm produces k clusters with center and guaranteed intra cluster similarity and the second algorithm is proposed to create a dendogram using k clusters as objects with guaranteed intra cluster similarity. (John et al, 2010) (Asano et al, 1988) (Gabow et al, 1986) (Gower and Ross, 1969) (Prim, 1957).

A MST based approach to facilitate easy web navigation by utilizing the concept of clustering. It constructs a minimum spanning tree of a point set and removes edges that satisfy a predefined criteria. It is being necessary to create automated tools to find the desired information resources from the web in client side and server side which mines knowledge. (Prateek Dwivedi et al, 2012).

A topology control algorithm which uses minimum spanning tree called local minimum spanning tree for wireless multi hop networks are designed in which each and every node builds its local minimum spanning tree independently. The usage of the above algorithm is network connectivity where, the degree of any node in the resulting topology is bounded by 6 and the topology can be transformed into another with bidirectional links. (HOU and sha l,2003) (C. Monma and Suri,1991).

A new algorithm is proposed to use maximum spanning tree of a graph in recorded stripes for capturing 3D surface measurements. It utilizes the connectivity and adjacency in recorded stripes which yielded more accurate and significant results than previous attempts in capturing 3D surface measurements (Willie et al, 2008), (Chartrand and Oellermann, 1992).

A real time face identification which has three phases for registering of face images using correlation, framework for global registration of face database using minimum spanning tree algorithm and a method for selecting subset of features and to discriminate between clients and impostors. The verification performance is improved in this approach. (Jonsson et al, 1998).

Another two minimum spanning tree clustering algorithms are proposed in which the first algorithm produces a k-partition of a set of points for any given k and constructs a minimum spanning tree by removing the edges which satisfies a predefined crieteria. The process is repeated until k clusters are produced.

The second algorithm partitions a point set into a group of clusters by maximizing the overall standard deviation reduction, without a given k values. (Oleksandr et al, 2006).

An algorithm to find orderly pair for any connected planar graph G which consisted of an embedded planar graph H isomorphic to G and an orderly spanning tree of H is discussed.(Chiang et al, 2005).

An algorithm to summarize the multiple documents available from the web is achieved by ranking the sentences using cosine similarity measure and reducing the summary by Maximal Marginal Relevance (MMR) technique. Sentences in the summary are organized by constructing a graph where each sentence represents nodes of graph and edges are maintained between every pair of vertices which represents the similarity between the sentences. The first sentence to be placed in the ordered summary is found by identifying the sentence which has minimum similarity with the sentences in the extracted summary. Ordering of remaining sentences in the summary is fixed one by one using Prim's Maximum Cost Spanning tree algorithm. (Ansamma, 2011).

The three different Maximum Spanning Tree approaches Kruskals, Prim's and Borovkas are discussed along with Dijktra's algorithm which implements Backtracking concept.

4.3.1 KRUSKAL'S ALGORITHM

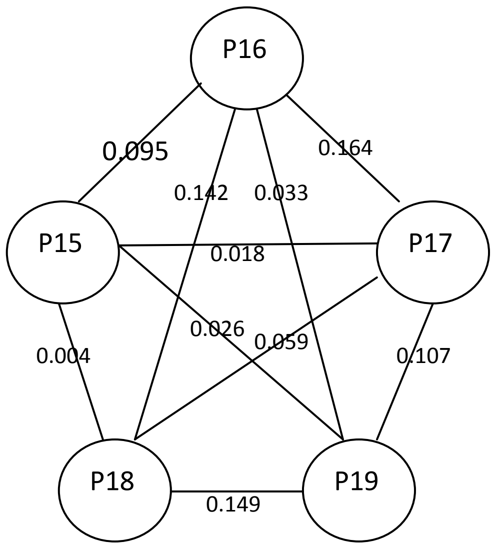
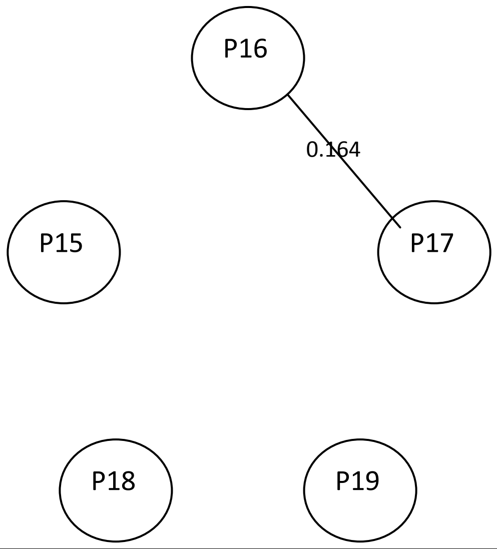
Kruskal's algorithm is a greedy algorithm which is used to design minimum/maximum spanning tree in a connected graph.

Initially all the n vertices without edges are considered so that each vertex is a component. All the edges are arranged in their descending order of costs. The edge with the biggest cost is added to T if and only if the edge connects two different components (i. e) it does not form a loop. The process of adding the edges is done for n-1 times. A single tree is the output obtained from a Kruskal's algorithm. The time complexity of the algorithm is O(mlogn) where m is the number of edges and n is the nodes. This algorithm works best if the number of edges is kept to a minimum.

```
Let $G = (V, E)$ be the given graph, with $|V| = n$
    {
    $T = (V, \Phi)$ consisting of only the vertices of $G$ and no edges;
    Arrange E in the order of decreasing costs;
    **for** ($i = 1, i \le n - 1, i++$)
    { Select the next biggest cost edge;
    **if** (the edge connects two different connected components)
    add the edge to $T$;
    }}
  }
```

Figure 4.2 Kruskal's algorithm

Figure 4.2 Shows the Kruskal's algorithm based on which the stage by stage of spanning tree is obtained as below.

Original Graph	**Stage 1**
 P16 0.095 0.164 0.142 0.033 P15 P17 0.018 0.026 0.059 0.004 0.107 P18 P19 0.149	

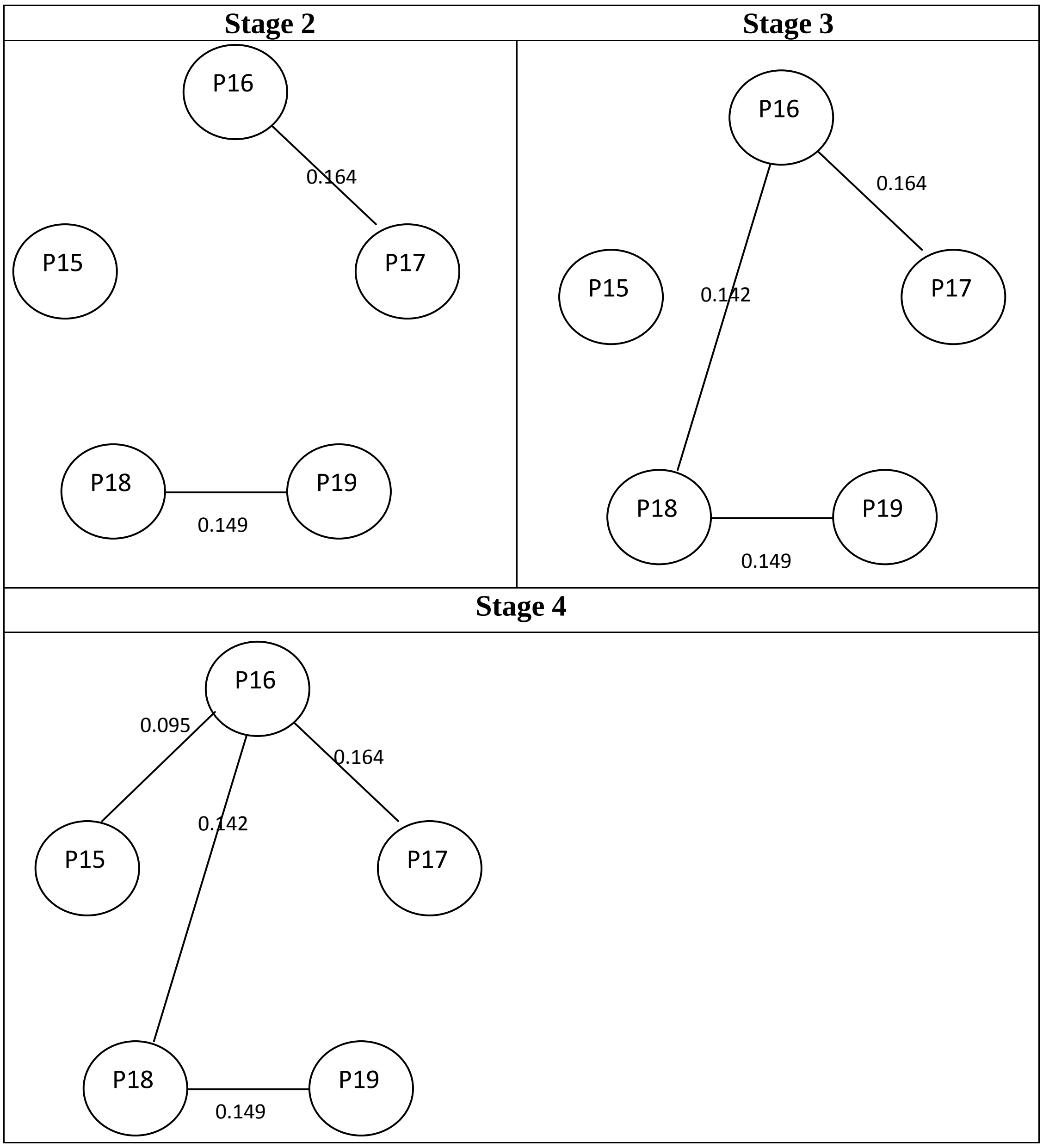


Figure 4.3 Stage by Stage Tree obtained in Kruskal's algorithm

Figure 4.3 shows the Stage by Stage maximum spanning tree obtained using Kruskal's algorithm in which initially the edge with cost 0.164 is selected followed by 0.149, 0.142 and 0.095 with a total cost of 0.550.

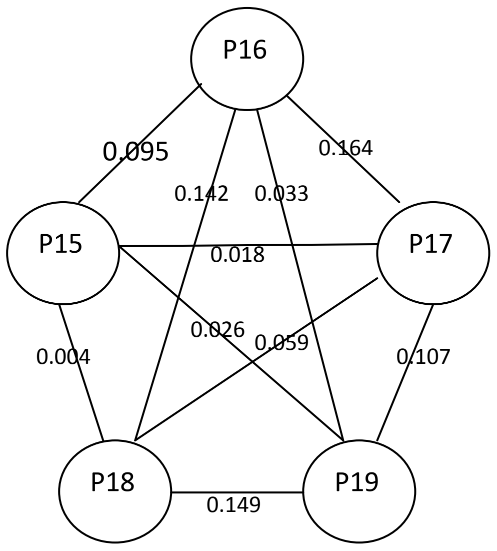
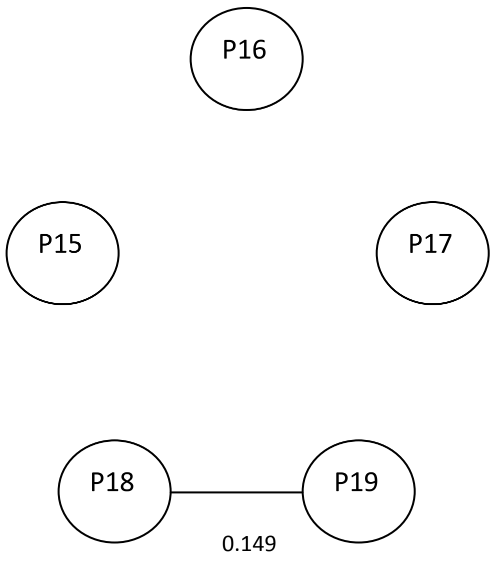
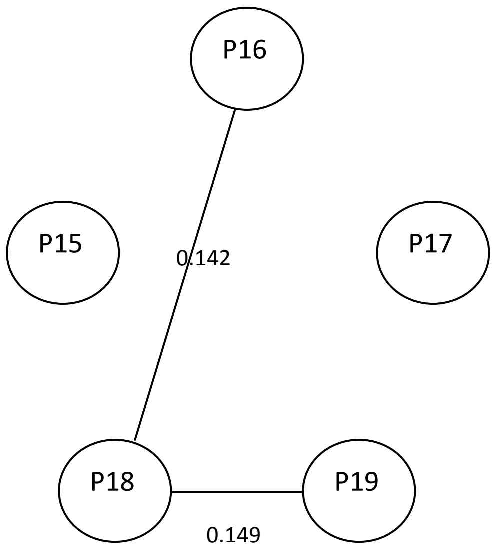
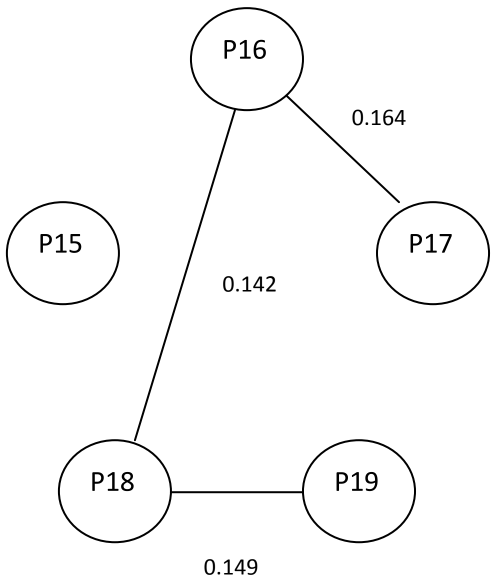
4.3.2 PRIMS ALGORITHM

This algorithm starts with one initial node. It initially selects any vertex and the maximum edge connected to that vertex.

Running Time = O(m + n log n) (m = edges, n = nodes)

If a heap is not used, the run time will be O(n^2) instead of O(m + n log n). However, using a heap complicates the code. The running time of the above algorithm is $O(n^2)$ algorithm.

```
Prim's algorithm

{
        T = Φ;
        U = { 1 };
        while ($U \neq V$)
        {
           let ($u$, $v$) be the lowest cost edge
           such that $u \in U$ and $v \in V - U$;
$T = T \cup \{(u, v)\}$
$U = U \cup \{v\}$
        }

    }
```

Original Graph	Stage -1
	
Stage 2	**Stage 3**
	

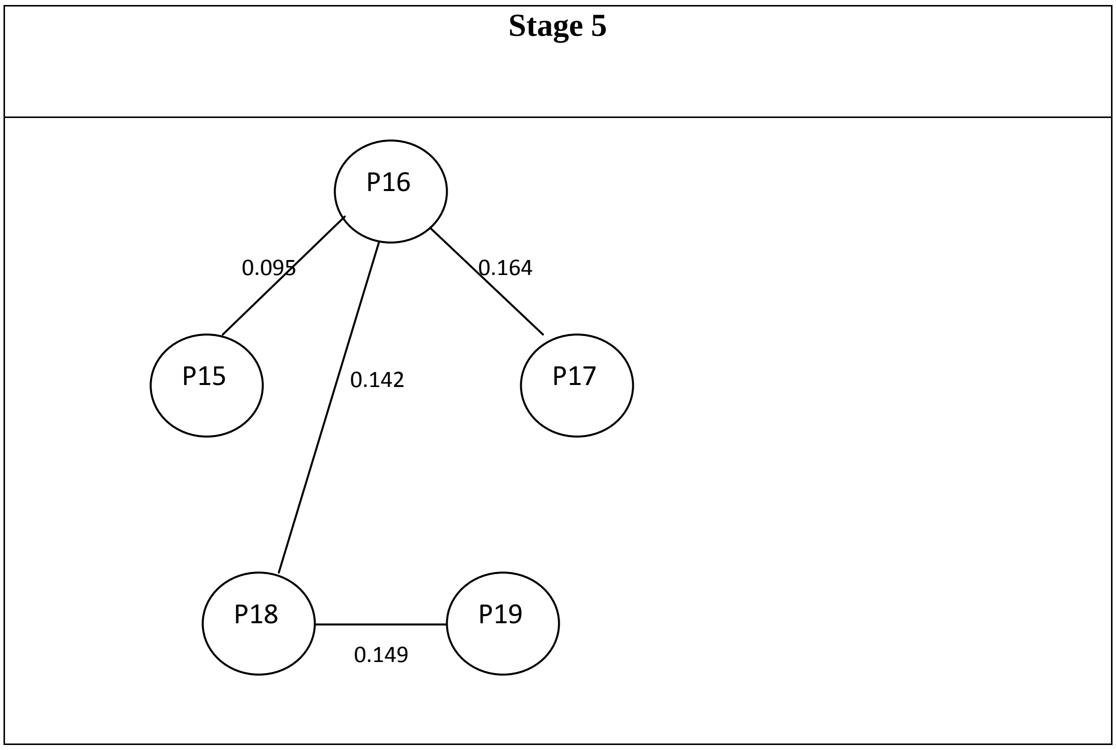


Figure 4.5 Stage by Stage tree obtained in Prim's algorithm

Figure 4.5 shows the Stage by Stage maximum spanning tree obtained in Prim's algorithm in which the edge with 0.149 is selected initially followed by 0.142 , 0.164 and 0.095 with a total cost of 0.550.

4.3.3 BORAVKA'S ALGORITHM

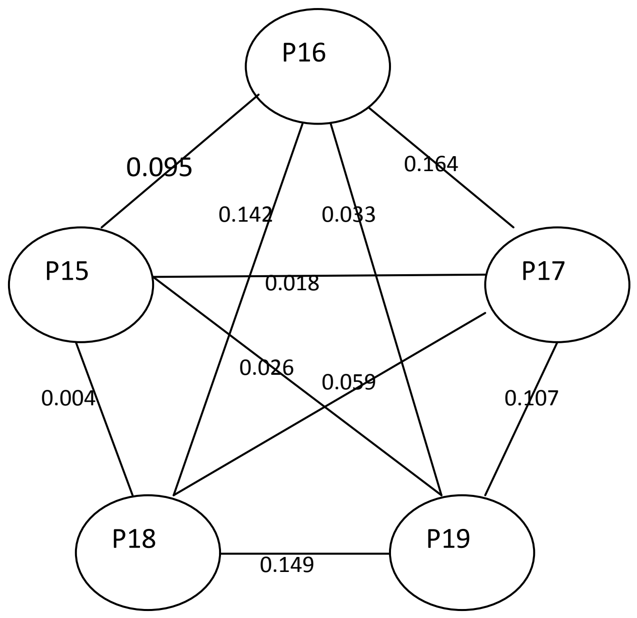
In this algorithm for any vertex select the maximum weight of this vertex and that edge/edges are marked. Search connected vertices and replace them by the new vertex. Repeat the cycles and if two vertices are connected by more than one edge delete all edges except the costliest.

The number of connected components will be reduced by at least a factor of 2 in each iteration. Therefore, there are at most log n iterations. Therefore, the total time can be O(m log n).

```
Boruvka MST
    Given G = (V,E)
    T = graph consisting of V with no edges
    while T has < n-1 edges do
      for each connected component C of T do
        e = min cost edge (v,u) s.t. v in C and u not in C
        T := T union {e}
```

Figure 4.6 Boravka's algorithm

Figure 4.6 shows the Boravka's algorithm

Original Graph
 P16 0.095 0.164 0.142 0.033 P15 P17 0.018 0.026 0.059 0.004 0.107 P18 P19 0.149

Spanning Tree obtained using Borovka's

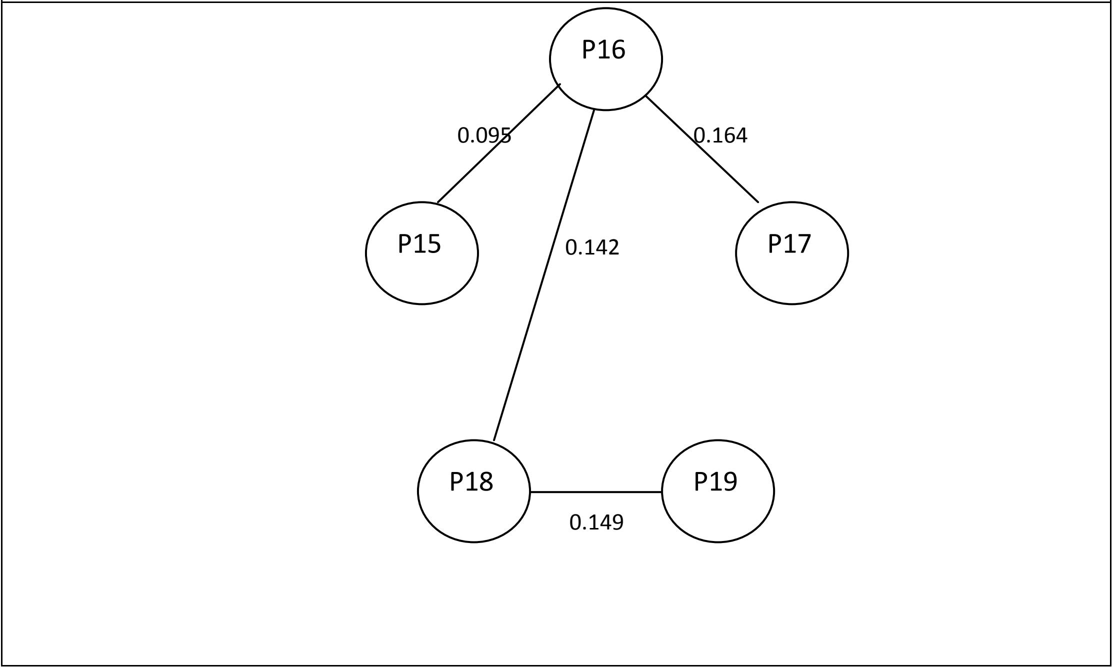


Figure 4.7 Spanning Tree in Borovka's algorithm

Figure 4.7 shows the spanning tree obtained using Boravka's algorithm and the maximum spanning tree cost obtained is 0.550.

4.4 DIJKSTRA'S ALGORITHM

Single-Source Shortest Path Problem is the problem of finding shortest paths from a source vertex v to all other vertices in the graph. Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory. It works on both directed and undirected graphs. However, all edges must have nonnegative weights.

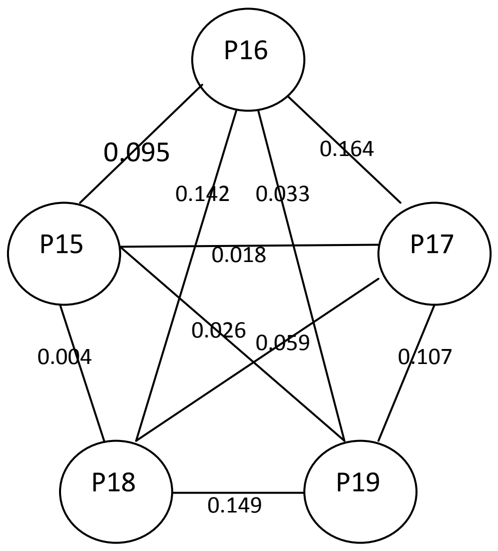
The working of Dijkstra's algorithm is as follows. Mark the distance to every node on the graph with infinity to mark that the node has not yet been visited. Now, at each iteration, select a *current* node. For the first iteration the current node is the starting point and the distance to itself will be zero. For subsequent iterations after the first the current node will be the closest unvisited node to the starting point.

From the current node, update the distance to every unvisited node that is directly connected to it. This is done by determining the sum of the distance between an unvisited node and the value of the current node, and relabeling the unvisited node with this value if it is less than its current value. In effect, the node is relabeled if the path to it through the current node is shorter than the previously identified paths. After updating the distances to each neighboring nodes, mark the current node as *visited* and select the unvisited nodes with the lowest distance from the starting point as the current node. Nodes marked as visited are labeled with the shortest path from the starting point to it and will not be revisited or returned to. This process of updating the neighboring nodes with the shortest distances, then marking the current node as visited and moving onto the closest unvisited node until marking the destination as visited. After marking the destination as visited it has been determined the shortest path to it, from the starting point, can be traced back, following the arrows in reverse.

```
Approach: Greedy

Input: Weighted graph G={E,V} and source vertex $v \in V$, such that all edge weights are
nonnegative

Output: Lengths of shortest paths (or the shortest paths themselves) from a given
source vertex $v \in V$ to all other vertices

dist[s]$\leftarrow$0                  (distance to source vertex is zero)
for all $\in$ V–{s}
    do  dist[v] $\leftarrow\infty$            (set all other distances to infinity)
S$\leftarrow \phi$                    (S, the set of visited vertices is initially empty)
Q$\leftarrow$V                     (Q, the queue initially contains all vertices)
while Q $\neq\phi$                 (while the queue is not empty)

do   u $\leftarrow$ mindistance(Q,dist)   (select the element of Q with the min. distance)
     S$\leftarrow$S$\cup${u}              (add u to list of visited vertices)
     for all v $\in$ neighbors[u]
         do  if   dist[v] > dist[u] + w(u, v)        (if new shortest path found)
                then    d[v] $\leftarrow$d[u] + w(u, v)     (set new value of shortest path)

                                    (if desired, add traceback code)

return dist
```

Figure 4.8 Dijkstra's algorithm

Original Graph



Single Source shortest Path obtained using Dijktra's algorithm

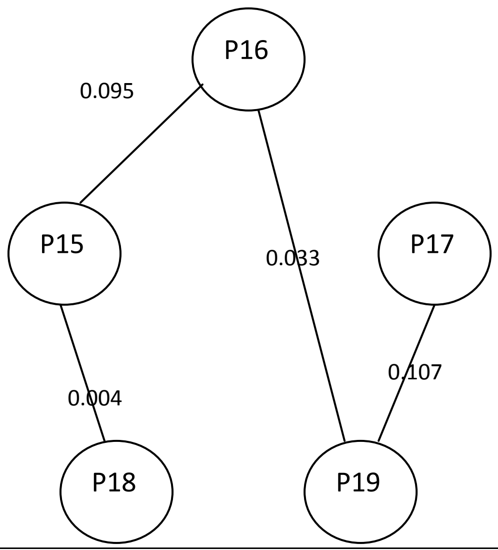


Figure 4.9 Single Source Shortest Path obtained in Dijkstra's algorithm

Figure 4.9 shows the Single Source Shortest Path obtained using Dijktra's algorithm in which the shortest path from the node P16 to P15 is 0.095 and to P18 is 0.099, to P19 is 0.033 and to P17 is 0.140.

4.5 DISCUSSIONS

The maximum spanning tree model is useful to evince maximization problems for further modeling . The same has been applied in web 3.0 for designing a domain specific models for Students, Faculty & IT Professionals. A maximum spanning tree model in web 3.0 for students, faculty and IT professionals reveals the factors which are required for design and development of a model which maximizes the weightage on the whole. In this problem the following are to be addressed in which it differs from the traditional model.

1. Instead of overall maximization of weight, the inter node weight must also be maximum at each and every point.

2. Must conclude the starting point and stopping point

3. Must know about the ordering so that can get a flow of vertices

In this chapter, the existing algorithms for maximum spanning tree is analyzed with their costs. The insight into the applicability of maximum spanning tree model in web 3.0 domain analysis was presented with the enhancements needed to be considered in the algorithm to be designed for maximum spanning tree model in Web 3.0.