Generating all degree constrained and degree preserving spanning trees of a weighted graph in order of increasing cost

K. Sangavai*, a, †, R. Anithaa, ‡

aDepartment of Mathematics and Computer Applications, PSG College of Technology, Coimbatore-641004, Tamil Nadu, India.

ABSTRACT

The degree constrained minimum spanning tree problem is to determine a spanning tree of the minimum total edge cost and degree not more than a given value d (d-MST). A number of algorithms have been proposed for this problem.

This paper presents two algorithms to generate all degree constrained spanning trees and all vertex subset degree preserving spanning trees of a weighted graph in order of increasing cost. By generating spanning trees in order of increasing cost, it is possible to determine the second smallest or in general the \( k \)-th smallest spanning tree of a graph. Time complexity of the algorithms were analyzed to measure the efficiency.

Keywords: Weighted graphs; degree constrained spanning trees; vertex subset degree preserving spanning trees; enumeration; computational complexity.

© 2012 Darbose. All rights reserved.

1. The degree constrained minimum spanning tree problem

A spanning tree of a graph such that each vertex in the tree has degree at most \( d \) is called a degree constrained spanning tree. The problem of finding the Degree Constrained Spanning Tree of minimum cost (DCMST) in an edge weighted graph is NP-complete for every \( d \), in the range \([2, n - 2]\), where \( n \) denotes the number of vertices.

DCMST problem entails finding a spanning tree of minimum cost such that no vertex in the tree exceeds a given degree constraint. This concept is useful in designing networks ranging from computer and telephone communications to transportation and sewage system. For instance, switches in an actual communication network will each have a limited number of connections available. Transportation systems must place a limit on the number of roads meeting in one place. Also, limiting the degree of each node limits the potential impact if a node fails. A degree constraint in a communication network also limits the liability in the case of node failure. In computer networks, the degree restrictions can be used to cater for the number of line interfaces available at a server or terminal.

Many authors have proposed solution methods for the DCMST problem, which include both exact and heuristic methods. Since this problem is NP-complete, heuristic methods have dominated. Some of the heuristics that have been investigated include: a number of basic MST algorithms of Prim (1957) and Kruskal (1956); the genetic algorithm by Zhou and Gen (1997); simulated annealing by Krishnamoorthy

*Corresponding Author
†sangavai_k@yahoo.com
‡anitha.padarajan@mail.psgtech.ac.in

As such there is no algorithm for generating all degree constrained spanning trees of a graph. In this paper, an algorithm to generate all degree constrained spanning trees of a given undirected weighted graph in order of increasing cost is presented.

2. The vertex subset degree preserving minimum spanning tree problem

The degree preserving spanning tree problem is to find in a connected graph $G(V, E)$, a spanning tree $T$ with maximum number of degree preserving vertices, was introduced by Broersma et al in [3] due to a nice practical application in network flow control. Peter Damaschke in [12] proved that the above-mentioned problem is NP-complete for bipartite planar degree - 5, 4, 3 graphs. R.Bhatia et al in [13] have given an approximation algorithm to find full degree spanning tree. We tried to generalize the problems dealt in [3] and [12] by introducing the problem of finding A-DPST, where $A$ is a non-empty subset of $V$. The spanning tree $T$ is called A-Degree Preserving Spanning Tree (A-DPST) if $deg_T(v) = deg_G(v), \forall v \in A$.

A commercial computer network can be represented by an undirected graph $G(V, E)$, with node set $V$ and edge set $E$ representing all possible connections. In virtually all cases, there is a need to expand beyond the confines of a single LAN to provide interconnections to other LANs and to wide area networks. The router is a more general purpose device capable of interconnecting such networks. In such a network if we consider any spanning tree, all the connections from the router to the networks should be retained to maintain the connections between the networks. This corresponds to $A = \{v\} - DPST$, where $v$ is the router in the network.

The minimum spanning tree problem with an added constraint that the vertices of $A$ should preserve their degrees in the spanning tree which can be termed as vertex subset degree preserving minimum spanning tree problem (A-DPMST). In [1], Anitha and Sangavai have proposed an algorithm to solve the problem. Here it is enriched to generate all A-DPST for a given $A$ in order of increasing cost, in a weighted graph.

Several algorithms exist for generating all spanning trees of a graph (e.g. Gabow & Myers, 1978; Kapoor & Ramesh, 1995; Matsui, 1993; Minty, 1965; Shioura & Tamura, 1995; Read & Tarjan, 1975; Kapoor & Ramesh, 2000; Matsui, 1997). Good space and time complexities are the most important concerns of these algorithms. Most algorithms generate spanning trees using some fundamental cut or circuit, but none of them takes the cost of the tree into account while generating spanning trees. The algorithms, which generate all spanning trees without weights (Minty, 1965; Read & Tarjan, 1975), can be applied to our problem by sorting the trees according to an increasing weight after they have been generated.

In this paper we are presenting two algorithms to generate all Degree Constrained Spanning Trees (DCST) and all A-Degree Preserving Spanning Trees (A-DPST). For this purpose we have adapted the partition technique used in the algorithm for generating all spanning trees of a weighted graph proposed by Kenneth Sorensen and Gerrit K. Janssens [16].

3. Generating All Degree Constrained Spanning Trees in Order of Increasing Cost

All graphs considered in this paper are connected, undirected, simple and weighted. Suppose $G(V, E)$ be a graph, whose vertex set is $V$, edge set is $E$ and $|V| = n$ and $|E| = m$. Let $w(e)$ be the weight function which associates a real number to each edge $e$. We will assume that $C(S_i)$ is the cost assigned to spanning tree $S_i$ and $i$ is the rank of $S_i$, when all spanning trees are ranked in order of increasing cost. We thus mean that $C(S_i) \leq C(S_j)$ if $i < j$. The sequence $S_1, S_2, S_3, \ldots$ is a ranking of spanning trees in order of increasing cost.
3.1 Terminology

3.1.1 Partition of the set of all spanning trees

A partition $P$ is defined to be a non-empty subset of the set of all spanning trees $A$ of a graph $G$ that has the following form

$$P = \{(i_1,j_1),(i_2,j_2),...,(i_r,j_r);(m_1,p_1),..., (m_l,p_l)\}$$

In other words, $P$ is the set of spanning trees containing all of the edges $(i_1,j_1),...,(i_r,j_r)$ (called included edges), and not containing any of the edges $(m_1,p_1),...,(m_l,p_l)$ (called excluded edges). Edges of $G$ that are neither included nor excluded in the partition are called open. The bar above edges $(m_1,p_1),...,(m_l,p_l)$ indicates that they are excluded edges. Because of the excluded edges, some partitions may not contain any spanning trees. This is the case when the graph $G$ from which the excluded edges of the partition are removed, is disconnected. Partitions that do not contain any spanning trees are called empty partitions. It should be remarked that $A$, the set of all spanning trees, is also a partition, but a special one that has no included or excluded edges (i.e. all edges are open).

3.1.2 A degree constrained minimum spanning tree in partition $P$

A degree constrained minimum spanning tree (DCMST) of the partition $P$ is defined as a degree constrained spanning tree of minimal cost that contains all included edges and none of the excluded edges of $P$. Since every spanning tree of partition $P$ contains the edges $(i_1,j_1),...,(i_r,j_r)$ a DCMST can be found by searching $m - r - l$ open edges of the partition. To ensure that all required edges are included into a DCMST of the partition, they can be added before all the remaining edges. To ensure that excluded edges are not in a DCMST, they can be temporarily assigned infinite cost. The way in which partitions are formed ensures that the set of included edges does not contain any circuits.

3.1.3 A degree preserving minimum spanning tree in partition $P$

A degree preserving minimum spanning tree (DPST) of the partition $P$ is defined as a degree preserving spanning tree of minimal cost that contains all included edges and none of the excluded edges of $P$. Unlike in the degree constrained minimum spanning tree generation, here the partition $P$ itself is constructed such that in all partition, the edges whose at least one end vertex is in $A$ are retained and the partitioning starts from an edge in $A$-DPMST which is different from those. DPMST for the partition can be found by searching the open edges of the partition. Starting with the included edges, open edges can be included to get the A-DPMST corresponding to the partition.

3.2 Procedure for ranking DCST in order of increasing cost

Given graph $G$ containing $n$-vertices, the following algorithm proceeds in stages. At stage $k$, the $k$-th DCST is generated.

3.2.1 Stage 1

Find a DCMST. Let it be $S_1 = \{(i_1,j_1),(i_2,j_2),...,(i_{n-1},j_{n-1})\}$. Now create the partitions $P_1, P_2, \ldots, P_{n-1}$, defined as

$$P_1 = \{(i_1,j_1)\}$$
\[P_2 = \{(i_1, j_1), (i_2, j_2)\}\]
\[P_3 = \{(i_1, j_1), (i_2, j_2), (i_3, j_3)\}\]
\[
\vdots
\]
\[P_{n-1} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_{n-2}, j_{n-2}), (i_{n-1}, j_{n-1})\}\]

Then the set \(\{P_1, P_2, \ldots, P_{n-1}\}\) forms a list for stage 1. The partitions that do not contain any degree constrained spanning tree may be removed from the list.

### 3.2.2 Stage \(k\)

Given a list for stage \(k - 1\), consisting of \(t\) (say) partitions \(L_1, L_2, \ldots, L_t\), we construct the degree constrained minimum spanning trees \(S(L_1), S(L_2), \ldots, S(L_t)\) (if possible) for each partition in the list and calculate the costs \(C[S(L_1)], C[S(L_2)], \ldots, C[S(L_t)]\), of each of these spanning trees.

Then the \(k\)-th smallest DCST is the degree constrained spanning tree with lowest cost.

\[S_k = \left\{ S(L_k)/C[S(L_k)] = \min_{j=1}^{t} C[S(L_j)] \right\} \]

\(L_k\) is the partition that contains the smallest degree constrained spanning tree of all spanning trees not yet generated. A list for stage \(k\) is formed by deleting \(L_k\) and all other partitions those do not possess any DCST from the list for stage \(k - 1\) and including the partitions formed by partitioning \(L_k\) by \(S(L_k)\). Ties are solved by picking one partition at random and by leaving the others in the list.

### 3.3 Example

![Graph G](image)

The first step in ranking all degree constrained spanning trees in order of increasing cost is to determine the degree constrained minimum spanning tree of \(G\). Let us fix \(d = 2\). Now we have to generate all spanning trees of \(G\), in which the degrees of all the vertices are less than or equal to 2.

We first get degree constrained minimum spanning tree of \(G\) by using Kruskal’s algorithm with modification mentioned in section 3.2.1. It equals \(S_1 = \{ab, ac, cd, de\}\). The weight of the spanning tree \(S_1\) is 10.

Now we partition \(G\) by \(S_1\), obtaining four partitions \(P_1, P_2, P_3\) and \(P_4\), are forming a list for stage 1.

\[P_1 = \{ab\}\]
\[P_2 = \{ab, ac\}\]
The next step is to calculate a degree constrained minimum spanning tree in each partition in the list. First let us try to construct the tree for the partition $P_1$. By deleting the excluded edge from $S_1$ we get the graph in Figure 2.

Now to form a DCMST, we can connect $b$ either by including the edge $bc$ or $bd$. Addition of $bc$ makes the degree of $c$ as 3 and addition of $bd$ makes the degree of $d$ as 3. Both violate the degree constraint. Therefore it is not possible to get a DCMST from the partition $P_1$ and therefore it is removed from the list.

Consider $P_2 = \{ab, ac\}$. Now to form a DCMST, we can connect $b$ either by including the edge $bc$ or $bd$. Addition of $bd$ makes the degree of $d$ as 3 which violates the degree constraint. So we include the edge $bc$ and get the spanning tree corresponding to $P_2$ as $\{ab, bc, cd, de\}$ whose cost is 12.

Similarly proceeding with other partitions the degree constrained minimum spanning tree of $P_3$ is $\{ab, ac, ce, de\}$, whose cost is 11 and for $P_4$ doesn’t exist. Now the list of partitions for the stage $2 = \{P_2, P_3\}$. Since $P_3$ has DCMST with lowest cost, let $S_2 = S(P_3) = \{ab, ac, de, ce\}$. By partitioning $P_3$ by its spanning tree we obtain the partitions, $P_{31} = \{ab, ac, cd, de\}$ and $P_{32} = \{ab, ac, cd, de, ce\}$. The degree constrained spanning trees are $\{ab, ac, ce, bd\}$ and $\{ab, ac, bd, de\}$ with costs 16 and 14 respectively. The list for stage 2 becomes $\{P_2, P_{31}, P_{32}\}$. Among those $P_2$ has the minimum DCST with minimum cost and therefore it is used for further partitioning. Continuing in the same way, ten degree constrained spanning trees with costs ranging from 10 to 17 are obtained.

3.4 Algorithm

1. Input the graph $G$ and the weight function $w(e)$

2. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the vertex set and $E = \{e_1, e_2, \ldots, e_m\}$ be the edge set of the graph $G$ arranged in non decreasing order of their edge weights. Let $d$ be the degree constraint.

3. Define $dc_1, dc_2, \ldots, dc_n$ as the degree counters of the vertices $v_1, v_2, \ldots, v_n$ respectively and initialize them as zero. Increment them by 1 whenever an edge incident to the corresponding vertex is added to the tree. Also initialize the tree $T$ as empty.

4. For $i=1$ to $m$, if $e_i (= (v_i, v_j))$ be the minimum cost edge that does not form a circuit when added to $T$ and $(dc_i + 1) \leq d$ and $(dc_j + 1) \leq d$, $T=T \cup \{e_i\}$ otherwise look for the next minimum weight edge.

5. The tree generated is a d-DCMST let it be $S_1$. 

\[ \text{Darbose} \]
6. Partition the edges of the MST obtained in the previous step as follows:

   (a) Let $e_1, e_2, \ldots, e_{n-1}$ be the edges in DCMST which are arranged in non-decreasing order of their weights

   (b) Let $L$ be the list of partition and initially it is empty. Form the partitions as follows:

   \[ P_1 = \{v_1\}; P_2 = \{e_1, v_2\}; \ldots; P_{n-1} = \{e_1, e_2, \ldots, e_{n-2}, v_{n-1}\} \]

7. After listing the partitions in the current stage, generate $d$-DCMST corresponding to each partition using step 4 and calculate their costs. Find the one whose cost is minimum. This is $S_2$.

8. Now proceed with step 6 and partition the edges of $S_2$.

9. Repeat until no spanning tree is left out.

10. Return the list of degree constrained spanning trees $S_1, S_2, \ldots$ in order of increasing cost.

Implementation of the algorithm can be done by using the following structure of the program.

---

**Algorithm 1: GENERATE DCST IN ORDER OF INCREASING COST**

**Input:** Graph $G(V, E)$; weight function and the degree constraint $d$  
**Output:** $S_1 :=$ d-DCMST $(G, w, d)$;  
List: $\triangleright$ Partition $(S_1)$;

**foreach** $P_i$ in list do  
\hspace{1em} if $e_i$ is not included in $P_i$ then  
\hspace{2em} Exclude $e_i$ from $S_i$;  
\hspace{2em} Add an edge $e_j := (v_i, v_k)$ with minimum cost, that does not form a circuit and $(dc_l + 1 \leq d)$ & $(dc_k + 1 \leq d)$;  
\hspace{2em} $dc_l = dc_l + 1; dc_k = dc_k + 1; S_2 := S_1$;  
\hspace{2em} Write $S_2 :=$ DCMST of $P_i$ to Output_File Remove $P_i$ from List Partition $(P_i)$
end

**Procedure** d-DCMST($G, w, d$);  
**Input:** Adjacency matrix of a graph $G$ (V, E), weight function and the degree constraint $d$.  
**Output:** The degree constrained minimum spanning tree (d-DCMST)  
List all the edges of the graph $G$ in order of non decreasing weight. $|V| = n; |E| = m$;  
Define the degree counters $dc_1, dc_2, \ldots, dc_n$ for the vertices $v_1, v_2, \ldots, v_n$ and  
$dc_1 = dc_2=\ldots = dc_n = 0$;  
$T := \phi$;  
$|E_T| = 0$;

**for** $i \leftarrow 0 \text{ to } m$ do  
\hspace{1em} $e_i(= (v_l, v_k)) :=$ the minimum cost edge that does not form a circuit when added to $T$ and $(dc_l + 1 \leq d)$ & $(dc_k + 1 \leq d)$;  
\hspace{1em} $T := T \cup \{e_i\}; dc_i = dc_i + 1; dc_k = dc_k + 1; |E_T| = |E_T| + 1$;  
\hspace{1em} if ($|E_T| = n-1$) then  
\hspace{2em} exit();
end

Return($T$)

---

Darbose
How does the algorithm work? Suppose $G$ is the given connected weighted graph and $L$ be the list of all d-DCST in order of increasing cost obtained as the output of the algorithm. Also let the number of elements in the list be $k$. Now we must prove that $k$ is equal the number of d-DCST and also they are in order of increasing cost.

The algorithm terminates only when there exists no open edge to the partitions corresponding to the spanning trees in the list at the current stage or all other partitions become empty. Suppose to the contrary that $k$ is not equal to the number of d-DCST. That is, let there be a spanning tree $T$ which is not in the list $L$. Then there must exists at least one open edge to the partition corresponding to any spanning tree in the list at the present stage. Then the algorithm could continue, which is a contradiction.

Since at each stage, the partition of minimum weight spanning tree is chosen and new partitions are formed. Open edges are included using Kruskal’s algorithm, which selects the minimum possible edge at that incident and therefore spanning trees generated corresponding to the partitions must be with minimum possible cost. And after the completion of each stage, the spanning trees are arranged in non-decreasing order of their costs and the one with minimum cost is considered for the next stage. Hence, the spanning trees in the output file will be in order of increasing cost.

**Time complexity analysis:** Let $|V| = n$ be the number of vertices, $|E| = m$, the number of edges and $N$ be the number of spanning trees of a given graph $G(V,E)$. The generation of spanning tree using Kruskal’s algorithm is $O(m \log m)$, the complexity of generating the spanning tree from a partition instead of a graph using this algorithm is also the same. The main loop in the algorithm is executed exactly $N$ times and therefore the procedure of partitioning is also executed $N$ times. Listing the partition and retrieving an item from the list can be done in $O(N)$ operations, since the maximum number of partitions is $N$-the number of spanning trees. Other steps including the degree constraint checking can be executed in constant time. Therefore the algorithm has time complexity $O(Nm \log m + N)$.

### 3.5 Procedure for ranking DPST in order of increasing cost

Find a A-DPMST.
Let it be $S_1 = \{(i_1, j_1), (i_2, j_2), ..., (i_l, j_l), (i_{l+1}, j_{l+1}), ..., (i_{n-1}, j_{n-1})\}$, where $(i_1, j_1), (i_2, j_2), ..., (i_l, j_l)$ are the edges whose at least one end vertex is in $A$. Now create the partitions $P_1, P_2, ..., P_{n-1-l}$, defined as

$$P_1 = \{(i_1, j_1), (i_2, j_2), ..., (i_l, j_l), (i_{l+1}, j_{l+1})\}$$

$$P_2 = \{(i_1, j_1), (i_2, j_2), ..., (i_l, j_l), (i_{l+1}, j_{l+1}), (i_{l+2}, j_{l+2})\}$$

$$...$$

$$P_{n-1-l} = \{(i_1, j_1), (i_2, j_2), ..., (i_l, j_l), (i_{l+1}, j_{l+1}), ..., (i_{n-1-l}, j_{n-1-l})\}$$

Then the set $\{P_1, P_2, ..., P_{n-1-l}\}$ forms a list for stage 1. As in the case of degree constrained spanning trees, here the degree preserving minimum spanning trees for partition in the list are formed. The partitions that do not contain any spanning trees may be removed from the list. The smallest A-degree preserving spanning tree is found. $L_k$ be the corresponding partition. A list for stage $k$ is formed by deleting $L_k$ and all other partitions those do not posses any A-DPST from the list for stage $k - 1$ and including the partitions formed by partitioning $L_k$ by the spanning tree $S(L_k)$ corresponding to it.

### 3.6 Example:

The first step in ranking all A-degree preserving spanning trees in order of increasing cost is to determine the A-degree preserving minimum spanning tree of $G$. Let us take $A = \{a, f\}$. Now we have to generate all spanning trees of $G$, in which the degrees of the vertices $a$ and $f$ are same as in $G$ (ie) 2, 2 respectively.

Darbose
We first get A-Degree Preserving Minimum Spanning Tree of $G$ by using the algorithm in [1]. It equals $S_1 = \{ab, ac, fd, fe, cd\}$. The weight of the spanning tree $S_1$ is 18.

Now we partition by $S_1$, obtaining $n - 1 - l = 1 (= 6 - 1 - 4)$ partition $P_1$, are forming a list for stage1 where $P_1 = \{ab, ac, fd, fe, cd\}$. The spanning tree corresponding to $P_1$ is $\{(ab, ac, fd, fe, ce)\}$ and the cost $C[S(P_1)] = 19$. Let $S_2 = S(P_1)$.

Now partition by $S_2$ as $P_{12} = \{ab, ac, fd, fe, cd, ee\}$. Then the spanning tree corresponding to $P_{12}$ is $\{ab, ac, fd, fe, bd\}$ and its weight is 22 and let it be $S_3$. There are no more partitions and all three, $A = \{a, f\}$-DPST are obtained.

3.7 Algorithm

Given weighted graph $G$ containing $n$ vertices and $m$ edges, the following algorithm generates all A-DPST in order of increasing cost. At stage $k$, the $k$-th minimum A-DPST is generated.

1. Input the graph through its adjacency matrix and a non-empty subset $A$ of $V$.

2. Find whether $G_A = (V, E_A), E_A$- the collection of edges in $G$ whose at least one end point is in $A$, is a forest or not. If it is so, proceed to step3. Else return “No A-DPST exists for the given $A$”

3. Initialize A- DPMST $T_A = E_A$. Include edges of minimum weight to $T_A$, without forming circuits until there are $n-1$ edges in $T$. 

Darbose
4. Now T will be the A-DPMST. Now create partitions P1, P2, ..., Pn−l as in the procedure explained above which forms the list for stage 1.

5. The minimum spanning trees corresponding to each partition in the list are calculated. Find the one whose cost is minimum. This is S2.

6. Now proceed with step 4 and partition the edges of S2. Repeat until no spanning tree is left out.

7. Return the list of A-degree preserving spanning trees S1, S2 ... in order of increasing cost.

Implementation of the algorithm can be done by using the following structure of the program.

<table>
<thead>
<tr>
<th>Algorithm 2: Generate All A-DPSTs in order of increasing cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph G (V, E), weight function and the vertex subset A whose degree preserving spanning trees are to be generated.</td>
</tr>
<tr>
<td><strong>Output:</strong> Output File (All A-DPSTs of G, sorted in order of increasing cost)</td>
</tr>
<tr>
<td>S1 := A-DPMST (G, w, A);</td>
</tr>
<tr>
<td>List := Partition (S1);</td>
</tr>
<tr>
<td>foreach Pi in list do</td>
</tr>
<tr>
<td>if ei is not included in Pi then</td>
</tr>
<tr>
<td>Exclude ei from S1;</td>
</tr>
<tr>
<td>Add an edge ej := (vl, vk) with minimum cost, that does not form a circuit;</td>
</tr>
<tr>
<td>S2 := S1;</td>
</tr>
<tr>
<td>Write S2 := DPMST of Pi to Output File Remove Pi from List Partition (S1)</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>Procedure A-DPMST (G, w, A);</td>
</tr>
<tr>
<td><strong>Input:</strong> Adjacency matrix of a graph G (V, E), weight function and a subset A of the vertex set V.</td>
</tr>
<tr>
<td><strong>Output:</strong> The A-Degree Preserving Minimum Spanning Tree (A-DPMST) List all the edges of the graph G in order of non decreasing weight.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>T := (A, EA), where EA - the collection of edges in G whose at least one end point is in A;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>for i ← 0 to m − k do</td>
</tr>
<tr>
<td>e1 := the minimum cost edge in E-EA that does not form a circuit when added to T;</td>
</tr>
<tr>
<td>T := T∪{e1};</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>if</td>
</tr>
<tr>
<td>exit();</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>Return(T)</td>
</tr>
</tbody>
</table>

By similar argument under the Algorithm 3.4, we can justify the correct output of this algorithm also.

**Time complexity analysis** Let |V| = n be the number of vertices, |E| = m, the number of edges |A| = l− the number of edges whose at least one end vertex is in A and M be the number of A-degree preserving spanning trees of a given graph G(V, E) which can be calculated by using Theorem 2.5 in [2]. The generation of spanning tree using Kruskal’s algorithm is \(O(m \log m)\). Since all spanning trees are initialized as the edges whose at least one end vertex is in A whose number is l, the complexity of generating the spanning tree from a partition instead of a graph using this algorithm is \(O((m – l) \log(m – l))\). The main loop in the algorithm is executed exactly M times and therefore the procedure
of partitioning is also executed $M$ times. Listing the partition and retrieving an item from the list can be done in $O(M)$ operations, since the maximum number of partitions is $M$, the number of A-degree preserving spanning trees. Other steps can be executed in constant time. Therefore the algorithm has time complexity $O(M(m - l) \log(m - l) + M)$.

4. Conclusion

In this paper two algorithms have been developed for generating all degree constrained spanning trees and all vertex subset degree preserving spanning trees of a given weighted graph in order of increasing cost. The time complexities of the algorithms are discussed. As such since no result regarding the number of degree constrained spanning trees in a graph, the algorithm may need to generate up to $n^{n-2}$ trees, to generate all such trees. But in the case of vertex subset degree preserving spanning trees, the number of edges to be added is reduced to $m - l$ and the number of such trees is obviously less than the number of spanning trees of the graph, the time complexity is not as much of the previous algorithm. However if we have one more constraint on the cost of the spanning tree, the algorithm can be stopped after reaching that constraint, in both the cases.

References
