Design and Analysis of Algorithms Report on

MAXIMUM LEAF SPANNING TREE

Submitted by

Lingamaneni Divya B160668CS
Ritika Prathigudupu B160540CS
K.A.S Bhavana B160533EC

Department of Computer Science and Engineering
National Institute of Technology Calicut
Calicut, Kerala, India - 673 601

November 25, 2019

Contents

Chapter 1

Definitions

1.1 Spanning_Trees(G)

 # all possible spanning trees of a graph
Given a graph G = $\langle V, E \rangle$, Spanning_Trees(G) is defined by :

$$Spanning\_Trees(G) = \Bigg\{ G' = \langle V', E' \rangle \Big| G'.V' = G.V \text{ and } G'.E' \subseteq G.E$$
$$\text{and } \Big(\forall V_i \in G'.V', \exists V_j \in G'.V' \text{ such that } V_i \neq V_j \text{ and } (V_i, V_j) \in G'.E' \Big)$$
$$\text{and } \mid G'.E' \mid = \mid G.V \mid -1 \Bigg\}$$

1.2 e(G,V_i)

#Edges in a graph with one of its vertex V_i
Given a graph G(V,E) and a vertex $V_i \in V$ then e(G, V_i) is defined by:-

$$e(G, V_i) = \Bigg\{ (V_i, V_j) \Big| V_j \in G.V and (V_i, V_j) \in G.E and V_i \neq V_j \Bigg\}$$

1.3 L(T)

#Leaves of a spanning tree/ sub-tree of a graph
Given a spanning tree / sub-tree T of graph G ,L(T) is given by :

$$\mathrm{L(T)} = \left\{ V_i \in T.V \middle| |e(T, V_i)| = 1 \right\}$$

Chapter 2

Formulation of Problem

2.1 Informal Statement Of a Problem

2.1.1 Optimization problem

Given a graph G, find a tree T such that T $= \langle V', E' \rangle \subseteq G = \langle V, E \rangle$ and T is a spanning tree of G such that number of leaves of T are maximised.

2.1.2 Decision problem

Given a graph $G = \langle V, E \rangle$ and a positive integer k, does a graph G have a spanning tree T with at least k leaves?

2.2 Problem Formulation

Given : A graph encoding $G = \langle V, E \rangle$ and a positive integer k
Output :

$$\begin{cases} 1 \text{ if } \exists T \Big| T \in Spanning\_Trees(G) and |L(T)| \geq k \\ 0 \text{ otherwise} \end{cases}$$

2.3 Language Formulation

Let MLST be class of maximum leaf spanning trees

$$\text{MLST} = \left\{ \langle G, k \rangle \middle| \text{ G is a graph encoding and k is a positive integer and } \exists T \in Spanning\_Trees(G) \text{ such that } |L(T)| \geq k \right\}$$

Chapter 3

Analysis of complexity of Max Leaf Spanning Tree Problem

Claim : MLST is NP-Complete
To Prove : MLST is NP and MLST is NP-Hard
i) MLST is NP :
To show that MLST is NP, it is enough to give a polynomial time verification algorithm.
ALGORITHM:

MLST_Verification $\big(\langle G, k\rangle, T = \langle V', E'\rangle \subseteq G\big)$
{

```
if(|T.V'| != |G.V| ) #checks whether all vertices are present in T
    return 0;

if (|T.E'| != |G.V| -1 ) #checks whether T is a tree with maximum edges
    return 0;

for each V_i in T.V':
    if (|e(T,V_i)|==0) #checks if T is connected
        return 0;
if(|L(T)| >= k) #counts no.of leaves
```

```
    return 1;
else
    return 0;
}
```

Time complexity of Algorithm: $O(1) + O(1) + O(n * n^2) = O(n^3)$
Hence it is in polynomial time.
Therefore, MLST is NP

ii) MLST is NP-Hard :
To prove that MLST is NP-Hard, it is enough to prove that any known NP-Hard problem is polynomially reducible to MLST.
Idea of Proof :- **Vertex Cover(VC)** $\leq_m^P$ **Minimum Connected Dominating Set (MCDS)** $\leq_m^P$ **MLST.**[3] [4]
Assumption: Vertex cover is the known NP-Hard problem.
Part 1: Vertex cover(VC) $\leq_m^P$ Minimum Connected Dominating Set (MCDS) for connected graphs.
Part 2: Minimum connected Dominating set (MCDS)$\leq_m^P$ MLST.
Proof for Part 1 :-

$$\textbf{Vertex cover(VC)} = \left\{ \langle G, k\rangle \,\middle|\, \text{G is a graph encoding and k is a +ve integer and } \exists V_c \,\middle|\, V_c \subseteq G.V \text{ and } |G.V| = k \forall (u,v) \in G.E \text{ (i.e) } u \in V_c \text{ or } v \in V_c \right\}$$

$$\textbf{MCDS} = \left\{ \langle G, k\rangle \,\middle|\, \exists V_c : V_c \subseteq G.V \text{ and } |V_c| = k \text{ and all } V_c \text{ are connected and } \forall V_i \in (G.V \backslash V_c), \exists V_j \in V_c \text{ such that } (V_i, V_j) \in G.E) \right\}$$

Note: We prove Part-1 and Part-2 in the context of connected graphs because, Maximum leaf spanning tree is meaningful only for connected graphs.

Now to prove Part 1 it is enough to prove that

$$\langle G, k\rangle \in VC \Leftrightarrow \langle G', k\rangle \in MCDS \tag{3.1}$$

where G' = F(G) where F is a function computable in polynomial time.

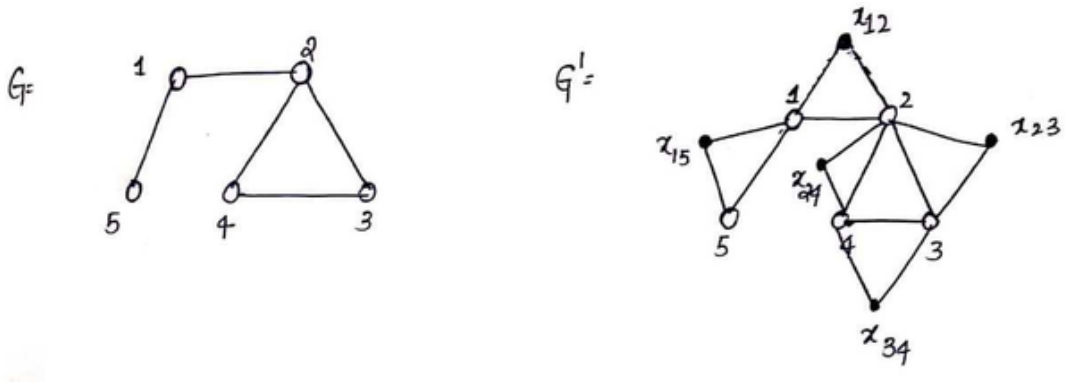


Figure 3.1: figure for demonstration of construction

Construction :

Construct G' = $\langle V', E' \rangle$ from G = $\langle V, E \rangle$ such that

$V' = G.V \cup \left\{ x_{uv} : \forall (u,v) \in G.E \right\}$

$E' = G.E \cup \left\{ (x_{uv}, u), (x_{uv}, v) : \forall (u,v) \in G.E \right\}$

Example : See Figure 3.1 G' is constructed from G

The above construction can be done in polynomial time. (addition of $2|E|$ edges)

Enough to prove : $\langle G, k \rangle \in VC \Leftrightarrow \langle G', k \rangle \in MCDS$

a) $\langle G, k \rangle \in VC \Rightarrow \langle G', k \rangle \in MCDS$

b) $\langle G', k \rangle \in MCDS \Rightarrow \langle G, k \rangle \in VC$

a) $\langle G, k \rangle \in VC \Rightarrow \langle G', k \rangle \in MCDS$

$\langle G, k \rangle \in VC \Rightarrow$

$$\exists V_c : V_c \subseteq G.V \wedge |V_c| = k \wedge \big(\forall (u,v) \in G.E \Rightarrow u \in V_c or v \in V_c\big)$$

i.e, V_c is the vertex cover.

Since G' is connected, V_c is connected in G' ————(1)

$\forall V_i \in (G'.V' \backslash V_c)$

if $V_i \in (G.V)$

$\exists V_j \in V_c$ such that $(V_i, V_j) \in (G.E) \in (G'.E')$ ——(2) # by definition of vertex cover.

else

V_i is an edge vertex of the form x_{uv} where u,v $\in (G.E)$
we know that $\forall(u,v) \in G.E \Rightarrow u \in V_c$ or $v \in V_c$
and we know that $\forall x_{uv} \in G'.V', \exists(u,v) \in G.E$
and we know that $(u, x_{uv}) \in G'.E' and (v, x_{uv}) \in G'.E'$ #(by construction)
$\Rightarrow \exists V_j \in V_c$ such that $(V_j, x_{uv}) \in G'.V'$ ————————-(3)

(2) and (3) $\Rightarrow$

$$\forall V_i \in (G'.V' \backslash V_c), \exists V_j \in V_c such that (V_i, V_j) \in G'.E' - -(4)$$

(1) and (4) $\Rightarrow V_c$ is a dominant set of G' with dimension k.

$$\Rightarrow \langle G', k\rangle \in MCDS$$

b) $\langle G', k\rangle \in MCDS \Rightarrow \langle G, k\rangle \in VC$
$\langle G', k\rangle \in MCDS \Rightarrow$

$\exists V_D : V_D \subseteq G'.V' \wedge |V_D| = k \wedge V_D$ is connected $\wedge\big(\forall V_i \in \big(G'.V' \backslash V_D\big), \exists V_j \in V_D$ such that $(V_i, V_j) \in G'.E'\big)$

We know that for each edge vertex $x_{uv} \in G'.V'$ there are 2 possibilities $x_{uv} \in V_D$ or $x_{uv} \in (G'.V' \backslash V_D)$.

Case-(i) :- $x_{uv} \in (G'.V' \backslash V_D)$

$\Rightarrow u \in V_D$ or $v \in V_D$ # since $(u, x_{uv}), (v, x_{uv})$ are only edges with vertex x_{uv} and V_D is dominant set.

$\Rightarrow V_D$ covers the edge (u,v) if $x_{uv} \notin V_D$ ————(1)

Case-(ii) :- $x_{uv} \notin (G'.V' \backslash V_D) \Rightarrow x_{uv} \in V_D$

Since V_D is a minimum dominant set $u \notin V_D$ and $v \notin V_D$ so that x_{uv} dominates u,v in G'.

Therefore $\forall x_{uv} \in V_D$ **replace** x_{uv} **with u in** V_D

Now u dominates x_{uv}, v

Therefore, replacement does not change the cardinality as well as property of V_D

After replacement x_{uv} with u in V_D, Case-(i) $\equiv$ Case-(ii)———(2)

(1) and (2) $\Rightarrow$

$\forall edges(u,v) \in G.E, V_D$ covers (u,v).

Therefore, V_D is a vertex cover of G with dimension k.

$\Rightarrow \langle G, k\rangle \in VC$

Hence proved Part-1

Proof for Part-2 :-(MCDS)$\leq_m^P$ MLST.

Enough to prove that :- $\langle G, k\rangle \in MCDS \Leftrightarrow \langle G, n-k\rangle \in MLST$

To prove :-

a) $\langle G, k\rangle \in MCDS \Rightarrow \langle G, n-k\rangle \in MLST$

b) $\langle G, n-k\rangle \in MLST \Rightarrow \langle G, k\rangle \in MCDS$

a) $\langle G, k\rangle \in MCDS \Rightarrow \langle G, n-k\rangle \in MLST$

It is enough to show if a graph G has a dominant set V_D such that $|V_D| = k$, then G has a spanning tree T with atleast n-k leaves.

Let V_D be the connected dominant set, we can construct a spanning tree T such that all $G.V \backslash V_D$ are leaves.

$\langle G, k\rangle \in MCDS \Rightarrow$

$$\exists V_D : V_D \subseteq G.V \wedge |V_D| = k \wedge V_D \text{ is connected } \wedge \Big(\forall V_i \in (G.V \backslash V_D), \exists V_j \in V_D \Big| (V_j, V_i) \in G.E \Big)$$

Construction:

Therefore, we can construct a tree such that T= $\langle V', E'\rangle$ where

V'= G.V

$$\text{E'} = \left\{ (V_i, V_j) \in G.E \middle| V_i \in V_D and V_j \in V_D \right\} \cup E\text{"}$$

where E" can be obtained by the following algorithm :

Algorithm-1:

E" = ϕ;

for each V_i in {G.V $\backslash V_D$}

  for each V_j in V_D

    if$(V_i, V_j \in G.E)$

    {

      E"=E" $\cup (V_i, V_j)$;

      break; # only one edge is added into E" for each $V_i \in G.V \backslash V_D$

      # and that edge is such that each $V_i \in G.V \backslash V_D$ is neighbour of some vertex $V_j \in V_D$

    }

end for

end for

As we can see in T = $\langle V', E' \rangle$ there may be cycles in the sub-graph of T induced by V_D, we can remove the cycles by the following algorithm

Algorithm-2:

for each V_i in V_D

for each V_j in V_D

for each V_k in V_D

if $\big((V_i \neq V_j \neq V_k) \wedge (V_i, V_j) \in T.E' \wedge (V_j, V_k) \in T.E'$
$\wedge (V_k, V_i) \in T.E' \big)$

$\{$

$T.E' = T.E' \backslash (V_i, V_k)$;

$\}$

end for

end for

end for

Now the resultant sub-graph T = $\langle V', E' \rangle$ can be said as a spanning tree of G with n-k leaves.

Since, we can see that

1)T is connected. # since, V_D is connected dominant set.

2) T has no cycles # since, cycles are removed , T is a tree.

3) T has n-k leaves # since, $\forall V_i \in (T.V' \backslash V_D), |e(T', V_i)| = 1$

4) T'.V'=G.V

Therefore, $\exists T' \in$ Spanning_Trees(G) with atleast (n-k) leaves.

Therefore, $\langle G, n-k \rangle \in MLST$

Therefore, $\langle G, k \rangle \in MCDS \Rightarrow \langle G, n-k \rangle \in MLST$

b) $\langle G, n-k \rangle \in MLST \Rightarrow \langle G, k \rangle \in MCDS$

$\langle G, n-k \rangle \in MLST \Rightarrow$

$\exists T : T \in Spanning\_Trees(G)$ and $|L(T)| = n-k$ (say, T=$\langle V', E' \rangle$)

To prove :- $\exists V_D : V_D$ is a connected dominant set of G with dimension k.

Claim :- $V_D = T.V' \backslash L(T)$ acts as a connected dominant set.

Proof :-

As the tree is connected, V_D is connected.

$\forall V_i \in (T.V' \backslash V_D) = L(T), \exists V_j \in V_D$ such that $(V_i, V_j) \in T.E' \in G.E$

\# since, $|e(V_i, T)| = 1$ and T is connected and T.E $\subseteq G.E$

Since, T is a spanning tree, T.V' = G.V

Therefore, above can be written as

V_D is connected in $G \wedge \big(\forall V_i \in (T.V' \backslash V_D) = L(T), \exists V_j \in V_D$ such that $(V_i, V_j) \in T.E' \in G.E\big)$

Therefore, V_D is a connected dominant set of G with dimension

$|T.V' \backslash L(T)| = n - (n - k) = k$

Therefore G has MCDS with dimension k

$\Rightarrow \langle G, k \rangle \in MCDS$

a) and b) $\Rightarrow$ MCDS$\leq_m^P$MLST

1) and 2) $\Rightarrow$ VC $\leq_m^P$ MCDS $\leq_m^P$MLST

Therefore Decision version of Maximum leaf Spanning tree Problem is NP-Complete

$\Rightarrow$ Optimization version of Maximum leaf Spanning tree Problem is NP-Hard

Chapter 4

Analysis of Complexity of Special Cases

Characterization of Maximum-leaf Spanning tree problem does not lead to resolution of its complexity.

i) MLST problem for bipartite graph is NP-Complete.[2]

ii) MLST problem for planar bipartite graphs with maximum degree 4 also remains NP-Complete.[2]

Proof for (i):-

Idea of proof :- **Set-covering problem** $\leq_m^P$ **MLSTB** (Maximum leaf Spanning Tree for Bipartite graphs)[2]

Assumption :- Set-covering problem is known to be NP-Complete.

To prove :- SC $\leq_m^P$ MLSTB

SC $= \left\{ \langle A, C, k \rangle \middle|$ A is a finite set, C is a collection of subsets of A, $\bigcup_{c \in C} c = A$, k is a positive integer, $\exists S \subseteq C \middle| |S| = k$ and $\bigcup_{s \in S} s = A \right\}$

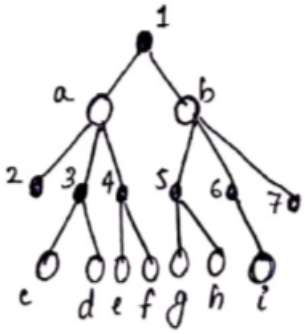
MLSTB $= \left\{ \langle G, k \rangle \middle|$ G is a bipartite graph with partite set X,Y , $G = \langle X \cup Y, E \rangle$, k is a positive integer and $\exists T \in$ Spanning_Trees(G) $\middle| |L_X(T)| = k$ where $L_X(T) = \left\{ l \in L(T) \middle| l \in X \right\}$

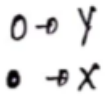
Property adopted(P_1) :-

Let G = $\langle V, E\rangle$ be a connected bipartite graph with partite sets X and Y. Let $k \leq |X|$ be a positive integer. (i.e) $G = \langle X \cup Y, E\rangle \Big| |L_X(T)| \geq k$ where $L_X(T) = \left\{ l \in L(T) \Big| l \in X \right\} \Leftrightarrow$ a)$|X/S| \geq k$

b) induced subgraph $S \cup Y$ of G is connected.

Example :-





Leaves that belong to X= $2, 7 \Rightarrow |X| = 2$

$S = 1, 3, 4, 5, 6$ such that $S \cup Y$ is connected and we see that $|X/S| \geq 2$

Therefore, the above property holds. It is intuitive/trivial and no need to prove it.

Proof:- **SC $\leq_m^P$ MLSTB**

Let $\langle A, C, K\rangle \in SC$ i.e, Let A be a finite set, C be a collection of subsets of A such that $\bigcup_{c\in C} c = A$, k is a positive integer.

$\Rightarrow \exists S \subseteq C \Big| |S| = k$ and $\bigcup_{s\in S} s = A$

Construction:

Step I :- Construct a bipartite graph G = ($X \cup Y$,E) from the incidence relation between C and A.

Example:- Let A = $\left\{ 1, 2, 3, 4 \right\} \equiv$ Y

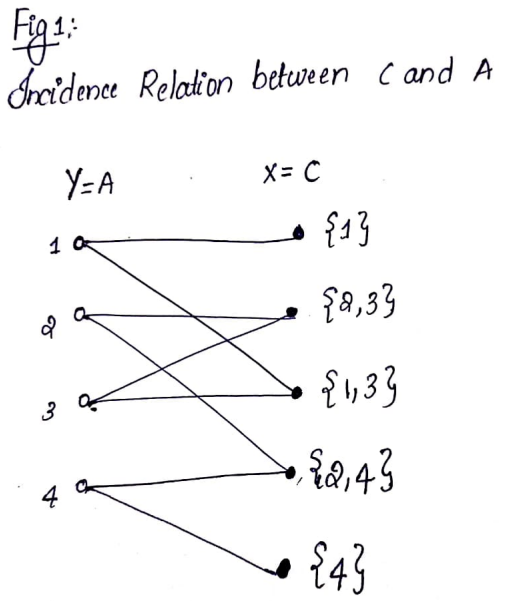


Figure 4.1: Figure corresponding to Step I

$$\mathrm{C} = \Big\{(1),(2,3),(1,3),(2,4),(4)\Big\} \equiv \mathrm{X}$$

Step II :- Add ∞ to Y and to each element of X to make sure that G is connected as in Fig 2.

Enough to prove :-(by property P_1

C contains a cover A of size k $\Leftrightarrow \exists S \subseteq X$ such that i) $|X/S| \geq |X| - k$

ii) $S \cup Y$ of G is connected.

Proof:- As X=C and Y=A, it is easy to see that :

1) C contains cover of A

$\Rightarrow \exists S \subseteq C \Big| |S| = k$ and $\bigcup_{s \in S} s = A$

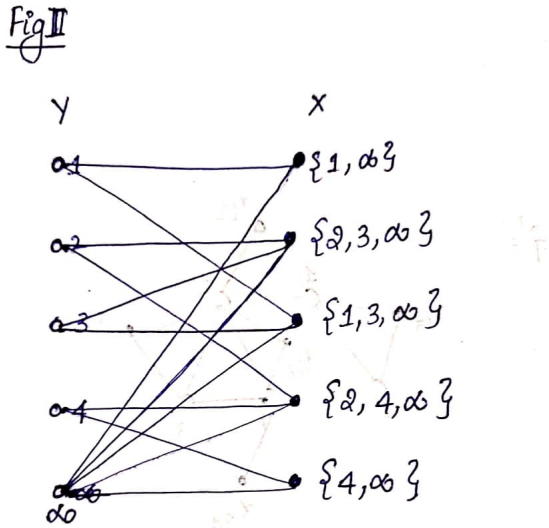


Figure 4.2: Figure corresponding to Step II

$\Rightarrow \exists S \subseteq X \Big| |S| = k$ and subgraph induced by $S \cup Y$ of G is connected (since, $\bigcup_{s \in S} s = Y$)

$\Rightarrow \exists S \subseteq X$ such that i) $|X/S| \geq |X| - k$

ii) $S \cup Y$ of G is connected.

2) If $\exists S \subseteq X \Big| |X/S| \geq |X| - k$ and $S \cup Y$ of G is connected $\Rightarrow$ C contains a cover of A (Easy to see)

Therefore, from (1) and (2),

$\langle A, C, k \rangle \in SC \Leftrightarrow \langle G, |X| - k \rangle \in MLSTB$

$\Rightarrow SC \leq_m^P MLSTB$

$\Rightarrow$ **MLSTB is NP-Complete.**

Therefore, no time complexity improvement for special cases.

Chapter 5

Approximate Algorithm for Max Leaf Spanning Tree Problem

As we have seen that Maximum-leaf Spanning tree problem is NP-Hard/NP-Complete it is likely that it does not have a polynomial time exact solution.Therefore, we concentrate on approximation algorithms.

5.1 Ravi and Lu's Greedy 3-Approximate Algorithm for Max Leaf Spanning Tree Problem[1]

Given a graph G, our algorithm computes a spanning tree of G by the following 2 steps:-

1) Obtain a Maximally Leafy forest F for G.

2) Obtain a spanning tree T of G which contains F by adding edges to F.

5.2 Proof for 3-approximability of the Algorithm[1]

Theorem 1. ***Let*** T^* ***be a maximum leaf spanning tree which is optimum. Let T' be a spanning tree obtained from Ravi and Lu's algorithm, which contains Maximally Leafy Forest F.***

Claim :- $|V_1(T^*)| \leq 3|V_1(T')$ ***i.e, solution is 3-approximate. where,*** $V_i(G) \rightarrow$

Vertices of G with degree = i, $\overline{V_i(G)} \rightarrow$ ***Vertices of G with degree*** $\geq i$

Proof. Assume that maximally leafy forest of G is F and F $= T_1 \cup T_2 \cup ..... \cup T_k$ where $T_1, T_2, ...T_k$ are leafy subtrees of G.
Leafy Subtree :- T is called leafy subtree of G if it has the following 2 properties :-
1) $\overline{V_3(T)} \neq \emptyset$
2) Neighbours of nodes with degree=2 are nodes with degree ≥ 3
$\Rightarrow |V_2(T)| \leq \overline{V_3(T)} - 1$.....(1)

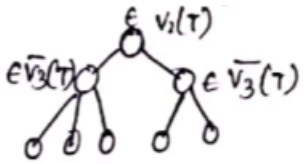


Figure 5.1: (1)

And we know that for any tree T the following property holds :
$|V_3(T)| \leq |V_1(T)| - 2$.....(2)

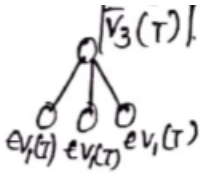


Figure 5.2: (2)

Therefore from (1) and (2),
For each $T_i \in F$
$|V(T_i)| = |V_1(T_i)| + \overline{|V_2(T_i)|} + \overline{|V_3(T_i)|}$
$\leq |V_1(T_i)| + \overline{|V_3(T_i)|} - 1 + \overline{|V_3(T_i)|}$
$\leq |V_1(T_i)| + 2\overline{|V_3(T_i)|} - 1$
$\leq |V_1(T_i)| + 2(|V_1(T_i)| - 2) - 1$
$\Rightarrow |V(T_i)| \leq 3|V_1(T_i)| - 5$ .....(3)
Because, F $= T_1 \cup T_2 \cup ..... \cup T_k$
(3) $\Rightarrow |V(F)| \leq 3|V_1(F)| - 5k$ .....(4)

Let T' be a spanning tree which contains F.
$|V_1(T')| \geq |V_1(F)| - 2(k-1)$ .....(5) # since while joining 2 trees to construct a spanning tree maximum of 2 leaves are lost.
Let T^* be the optimum spanning tree,
$|V_1(T^*)| \leq |V(F)| - (k-1)$ .....(6) # since minimum of (k-1) leaves are lost while forming a spanning tree.

(4) and (6) $\Rightarrow |V_1(T^*)^*| \leq |V(F)| - (k-1)$
$\leq 3|V_1(F)| - 5k - k + 1$
$\Rightarrow |V_1(T^*)^*| \leq 3|V_1(F)| - 6k + 1$ .....(7)

(5) and (7) $\Rightarrow |V_1(T^*)^*| \leq 3|V_1(F)| - 6k + 1$
$\leq 3(|V_1(T')| + 2(k-1)) - 6k + 1$
$\leq 3(|V_1(T')|) + 6k - 6 - 6k + 1$
$\leq 3(|V_1(T')|) - 5$
$\Rightarrow |V_1(T^*)| \leq 3|V_1(T')|$

$\Rightarrow \frac{|V_1(T^*)|}{V_1(T')|} \leq 3$

Therefore, Ravi and Lu's algorithm which finds a spanning tree which contains maximal leafy forest , is a 3-approximate algorithm.
Hence proved.

■

5.3 Pseudo code for Ravi and Lu's 3-Aprroximate Algorithm[1]

MaximumLeafSpanningTree(G)
1) Let F be an empty set # F = Maximal leafy forest
2) For every node v in G **do**
S(v) = v ; # set of vertices in tree with vertex v initiated to singleton set v
d(v) = 0 ; # degree(v) is initiated as 0
3) For every node v in G **do**
$S' = \emptyset$; # initializing trees containing vertices u such that (u,v) $\in G.E$

d'=0 ; # no.of such trees is initialized to 0

4) For every node $u \in adj(v)$ in G **do**

If $u \notin S(v)$ and $S(u) \notin S'$ then

d' = d'+1 ;

$S' = S' \cup$ S(u) ;

5) If d(u)+d' ≥ 3 then

For every S(u) in S' **do**

F = F $\cup$ (u,v);

S(v) = S(v) $\cup$ S(u);

d(u) = d(u)+1 ;

d(v) = d(v)+1 ;

6) # We obtain F from the above 5 steps which is maximally leafy forest.

7) $V' = \emptyset$; # vertices in F

for each edge (u,v) $\in$ F

if $(u \notin V')$

$V' = V' \cup u$;

if $(v \notin V')$

$V' = V' \cup v$;

8) $E' = \emptyset$;

for each edge (u,v) $\in$ G.E

if ($u \notin V'$ and $v \notin V'$)

$E' = E' \cup (u, v)$;

else if ($u \in V'$ and $v \in V$')

if ($u \notin S(u)$ *and* $v \notin S(u)$)

$E' = E' \cup (u, v)$;

9) $E' = E' \cup F$;

10) $V' = V' \cup \left\{ G.V/V' \right\}$

11) $G' = \langle V', E' \rangle$;

12) T=SpanningTree(G') ; # obtain spanning tree using Prim's algorithm which is easy.

13) return T' ;

Chapter 6

Conclusions & Reflections

The time complexity of the above suggested approximate algorithm is linear O((m+n)α(m,n)).

6.1 *Suggestions for Improvement:*

1. *The Approximation Ratio of our algorithm might be improved by growing the maximally leafy forest by the* ***descending order of the degree of nodes in G****.*
   Sorting the nodes in the descending order of their degree takes time O(m+n), since the degree of every node is no more than n.When we try to process the nodes of the Graph from high-degree nodes to low-degree nodes we could see an improvement of approximation ratio and the time complexity remains linear.

2. *We could also improve our algorithm to* ***2-approximate*** *by assigning priorities to the rules to expand the maximal leafy forest [5] which is done in polynomial time.*

3. *We could also use* ***Local Optimization Technique*** *(i.e)We could choose a random spanning tree and perform a* ***k-swap*** *(swap k non-tree edges with k tree edges) in each step until a local optimum is obtained.*
   As k increases the approximation ratio is improved but the complexity of the solution raises exponentially with k: $O(n^{(3k+1)})$
   2-LOT is 3-approximable and as k increases the approximability factor improves [6]

6.2 Applications of Maximum Leaf Spanning Tree Problem

Several applications of Maximum Leaf Spanning Tree Problem can be found in the area of communication networks and circuit layouts.

For example, let us consider the case of communication networks where the vertices correspond to terminals and the aim is to design a tree-like layout in the network. Then leaf terminals may have lighter work loads than intermediate terminals of degree at least two when intermediate terminals have the work on message routing.

Hence, in this case, the solution of Maximum Leaf Spanning Tree Problem could provide a reasonable layout.

6.3 Comparison of Performance of Various Approximation Algorithms[8]

Since the problem is NP-hard, several approximation algorithms have been considered.
BFS: For $v \in V$, apply the breadth first search algorithm rooted at v in the input graph G,to make a spanning tree in G.
Lu–Ravi: 3-approximation algorithm of Lu and Ravi.
Solis-Oba: 2-approximation algorithm of Solis-Oba.

Some results of applying above 3 algorithms for various random graphs is shown in Figure 6.1. In the table, *Leaves means the average number of leaves* and *Time means the average time in seconds.*As the table shows, in our implementation, ***BFS is best for graphs with*** (probability density of a vertex pair being in edge set of the graph) $p \geq 0.3$ and ***Lu–Ravi is best for sparse graphs with*** $p \leq 0.2$. This tendency remains true for other graphs.

Table 1
Comparison of heuristics for randomly generated graphs

Graph		**BFS**		**Lu–Ravi**		**Solis-Oba**	
n	*p*	Leaves	Time	Leaves	Time	Leaves	Time
50	0.1	33.8	0.00	35.4	0.00	34.0	0.00
	0.2	40.4	0.00	40.8	0.00	40.0	0.00
	0.3	44.0	0.00	43.3	0.00	43.2	0.00
	0.4	45.1	0.00	44.4	0.00	44.3	0.00
	0.5	46.4	0.00	45.8	0.00	45.5	0.00
	0.6	47.1	0.01	46.3	0.00	46.1	0.00
	0.7	47.8	0.01	47.0	0.00	47.0	0.00
100	0.1	78.3	0.01	82.0	0.00	77.7	0.00
	0.2	88.4	0.02	88.8	0.00	87.2	0.00
	0.3	92.4	0.03	91.8	0.00	90.9	0.01
	0.4	94.3	0.04	93.2	0.00	93.1	0.01
	0.5	95.8	0.04	94.7	0.00	94.8	0.01
	0.6	96.3	0.05	95.9	0.00	95.5	0.01
	0.7	97.1	0.06	96.3	0.00	96.2	0.01

Figure 6.1: Comparsion of Approximate algorithms for Random graphs

Bibliography

[1] Lu HI, Ravi R. Approximating maximum leaf spanning trees in almost linear time. Journal of algorithms. 1998 Oct 1;29(1):132-41

[2] Toulouse M. Variations of the maximum leaf spanning tree problem for bipartite graphs. Information Processing Letters, Elsevier. 2006 Feb 28;97(4):129-32.

[3] [Online] Available: `https://en.wikipedia.org/wiki/Connected_dominating_setp`

[4] [Online] Available:`https://cs.stackexchange.com/questions/108801/is-maximum-leaves-spanning-tree-np-complete?rq=1`

[5] Solis-Oba R. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In European Symposium on Algorithms 1998 Aug 24 (pp. 441-452). Springer, Berlin, Heidelberg.

[6] Lu HI, Ravi R. The power of local optimization: Approximation algorithms for maximum-leaf spanning tree. In Proceedings of the Annual Allerton Conference on Communication Control and Computing 1992 Oct (Vol. 30, pp. 533-533). University of Illinois.

[7] [Online] Available:`http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15854-f05/www/scribe/lec05.pdf`

[8] Fujie T. An exact algorithm for the maximum leaf spanning tree problem. Computers & Operations Research. 2003 Nov 1;30(13):1931-44.