Maximum Leaf Spanning Tree & Parallelization

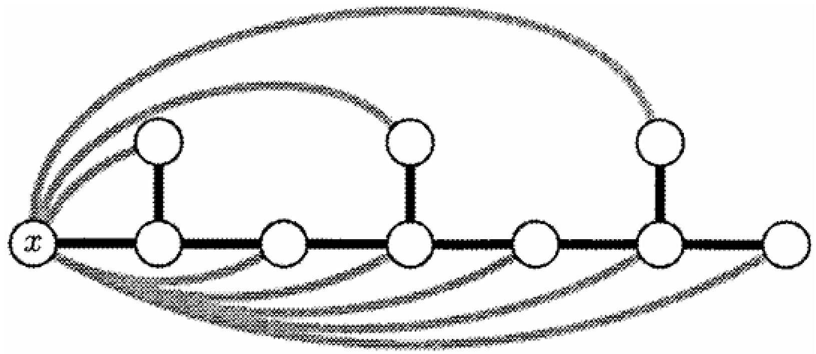
GALe Project

NGUYEN Léa xnguyel01

Table of contents

The MLST Problem

⟶ A Spanning Tree with the maximum number of leaves



MAX SNP problem:

- cycle-free.
- maximizes {v ∈ V and $deg(v)_T = 1$} ; G={V,E} and T the spanning tree
- No determinist algorithm in polynomial time

⟶ Approximation algorithms

+ Parallelizing ⟶ Reduce Time Complexity

2-Approximation MLST Algorithm, Robert Solis-Oba

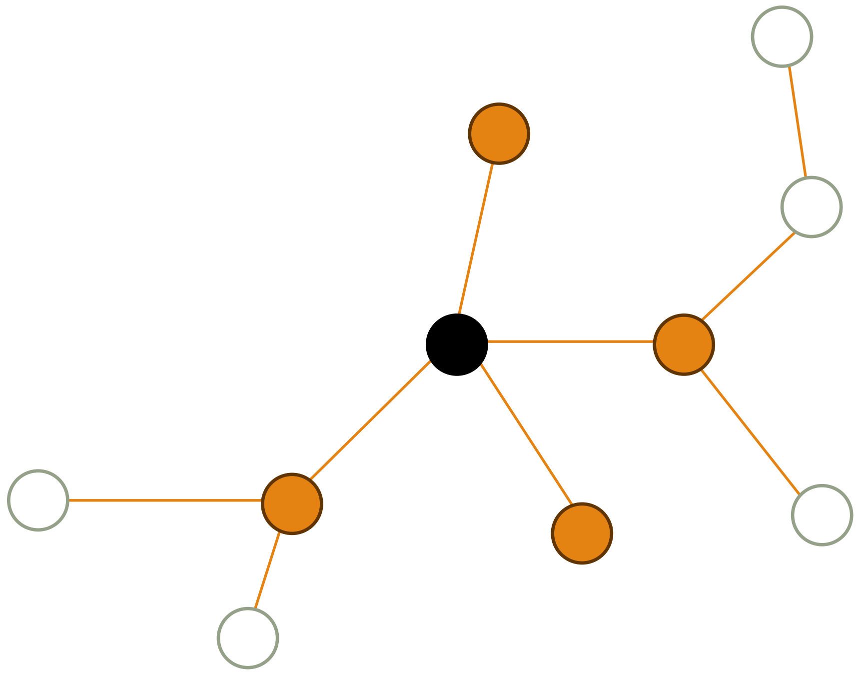
Ratio L_opt/L_2-app = 2

Linear time

2-Approximation MLST Algorithm, Robert Solis-Oba

1. MULTI THREADING AND EXPAND

Algorithm *tree*(G)
$F \leftarrow \emptyset$
while there is a vertex v of degree at least 3 **do**
    Build a tree T_i with root v and leaves the neighbors of v.
    while at least one leaf of T_i can be expanded **do**
        Find a leaf of T_i that can be expanded with
        a rule of largest priority, and expand it.
    end while
    $F \leftarrow F \cup T_i$
    Remove from G all vertices in T_i and all edges incident to them.
end while
Connect the trees in F and all vertices not in F to form a spanning tree T.

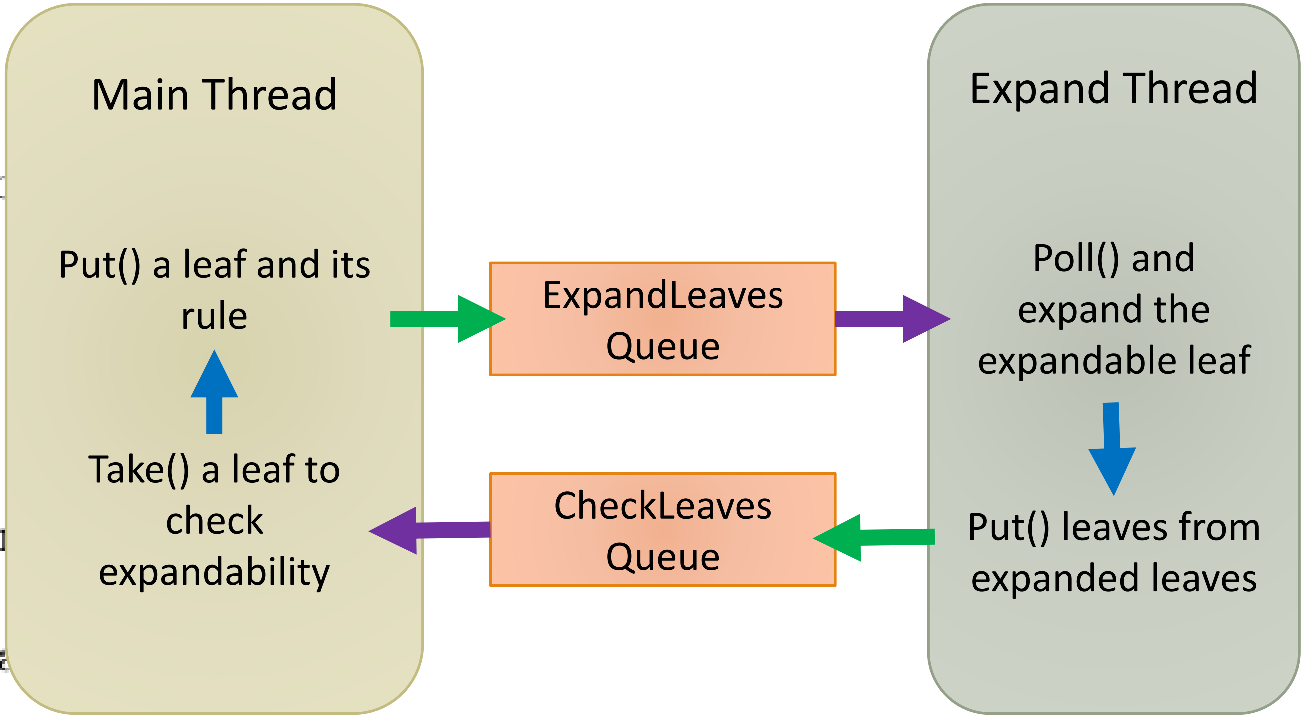


2-Approximation MLST Algorithm, Robert Solis-Oba

1. MULTI THREADING AND EXPAND

Algorithm *tree*(G)
$F \leftarrow \emptyset$
while there is a vertex v of degree at least 3 **do**
  Build a tree T_i with root v and leaves the neighbors o
  while at least one leaf of T_i can be expanded **do**
    Find a leaf of T_i that can be expanded with
    a rule of largest priority, and expand it.
  end while
  $F \leftarrow F \cup T_i$
  Remove from G all vertices in T_i and all edges inciden
end while
Connect the trees in F and all vertices not in F to form a

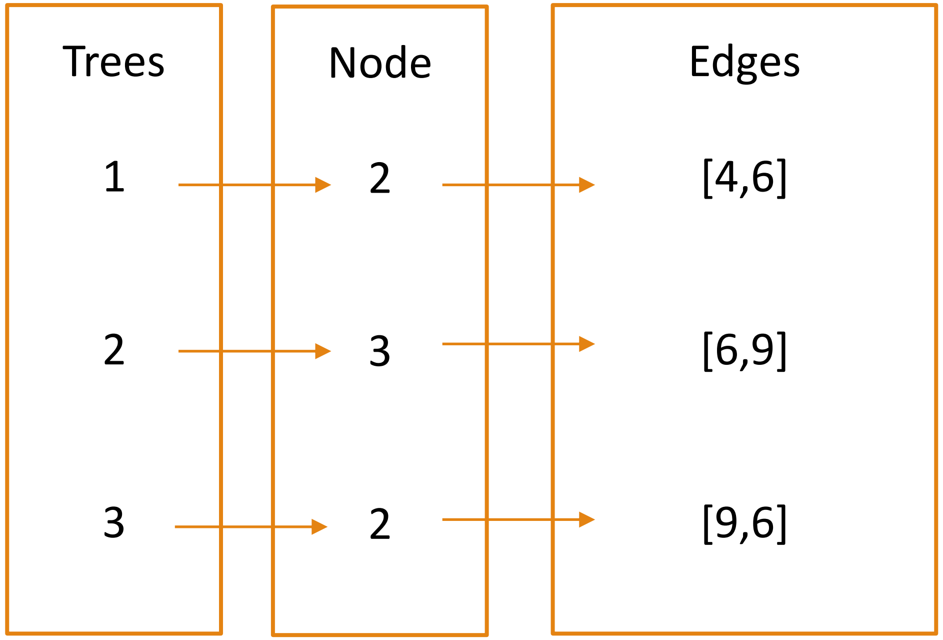
Producer – Consumer



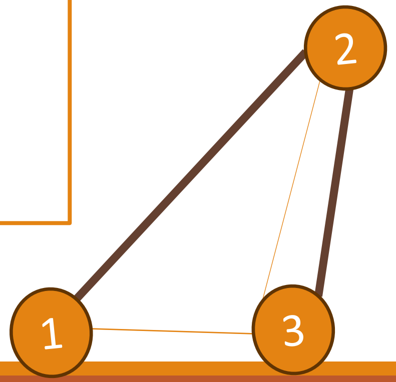
2-Approximation MLST Algorithm, Robert Solis-Oba

Algorithm *tree*(G)
$F \leftarrow \emptyset$
while there is a vertex v of degree at least 3 **do**
  Build a tree T_i with root v and leaves the neighbors of v.
  while at least one leaf of T_i can be expanded **do**
    Find a leaf of T_i that can be expanded with
    a rule of largest priority, and expand it.
  end while
  $F \leftarrow F \cup T_i$
  Remove from G all vertices in T_i and all edges incident to them.
end while
Connect the trees in F and all vertices not in F to form a spanning tree T.

2. CONNECT THE TREES & DFS



HashMap<Integer,int[]> []



3-Approximation MLST Algorithm, Lu & Ravi

Ratio L_opt/L_3-app = 3

Linear time

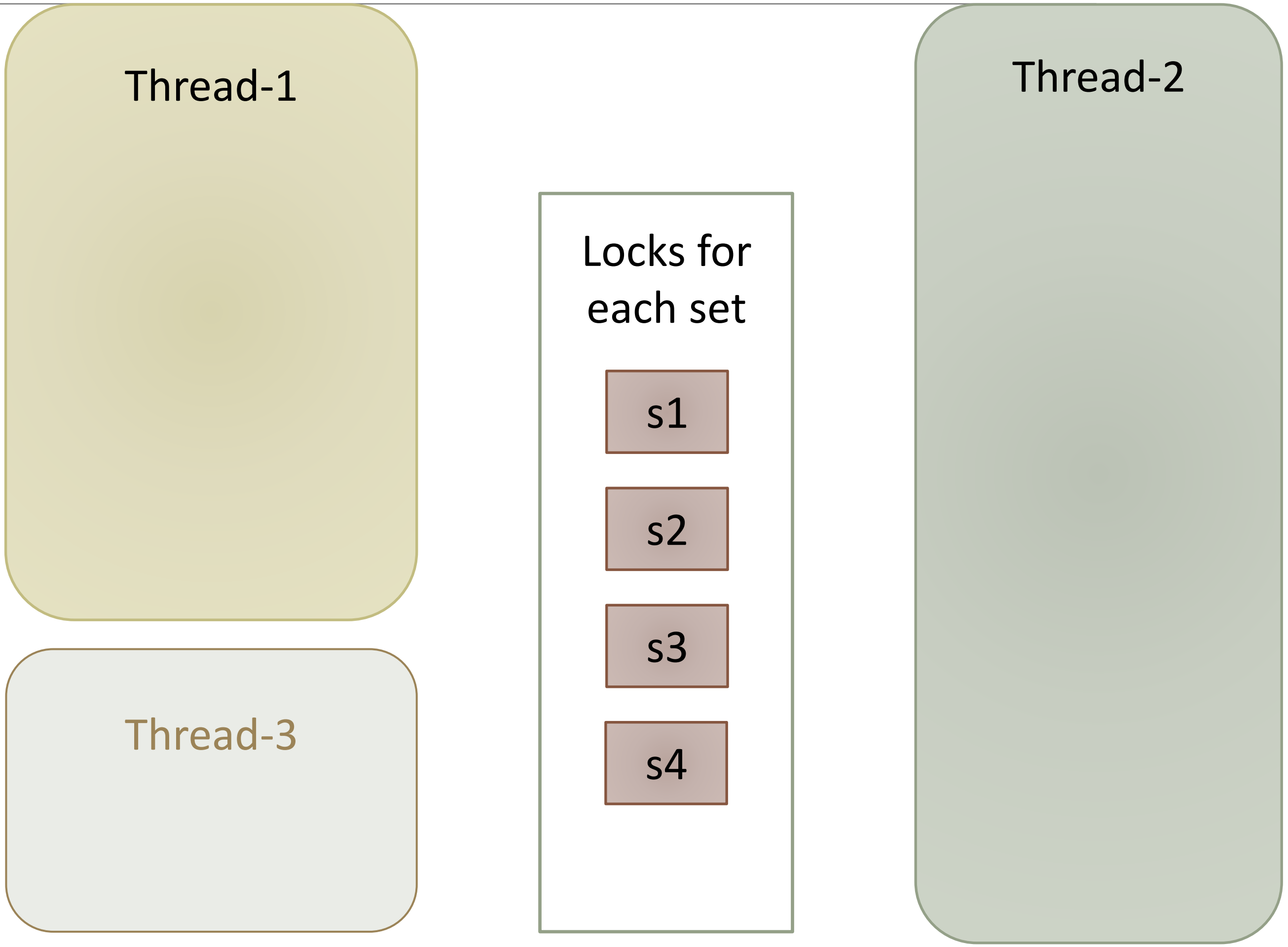
3-Approximation MLST Algorithm, Lu & Ravi

```
MaximallyLeafyForest(G)
Let F be an empty set.
For every node v in G do
    S(v) := {v}.
    d(v) := 0.
For every node v in G do
    S' := ∅.
    d' := 0.
    For every node u that is adjacent to v in G do
        If u ∉ S(v) and S(u) ∉ S' then
            d' := d' + 1.
            Insert S(u) into S'.
    If d(v) + d' ≥ 3 then
        For every S(u) in S' do
            Add edge uv to F.
            Union S(v) and S(u).
            Update d(u) := d(u) + 1 and d(v) := d(v) + 1.
Output F as a maximally leafy forest of G.
```

3-Approximation MLST Algorithm, Lu & Ravi

1. MULTI THREADING AND EXPAND

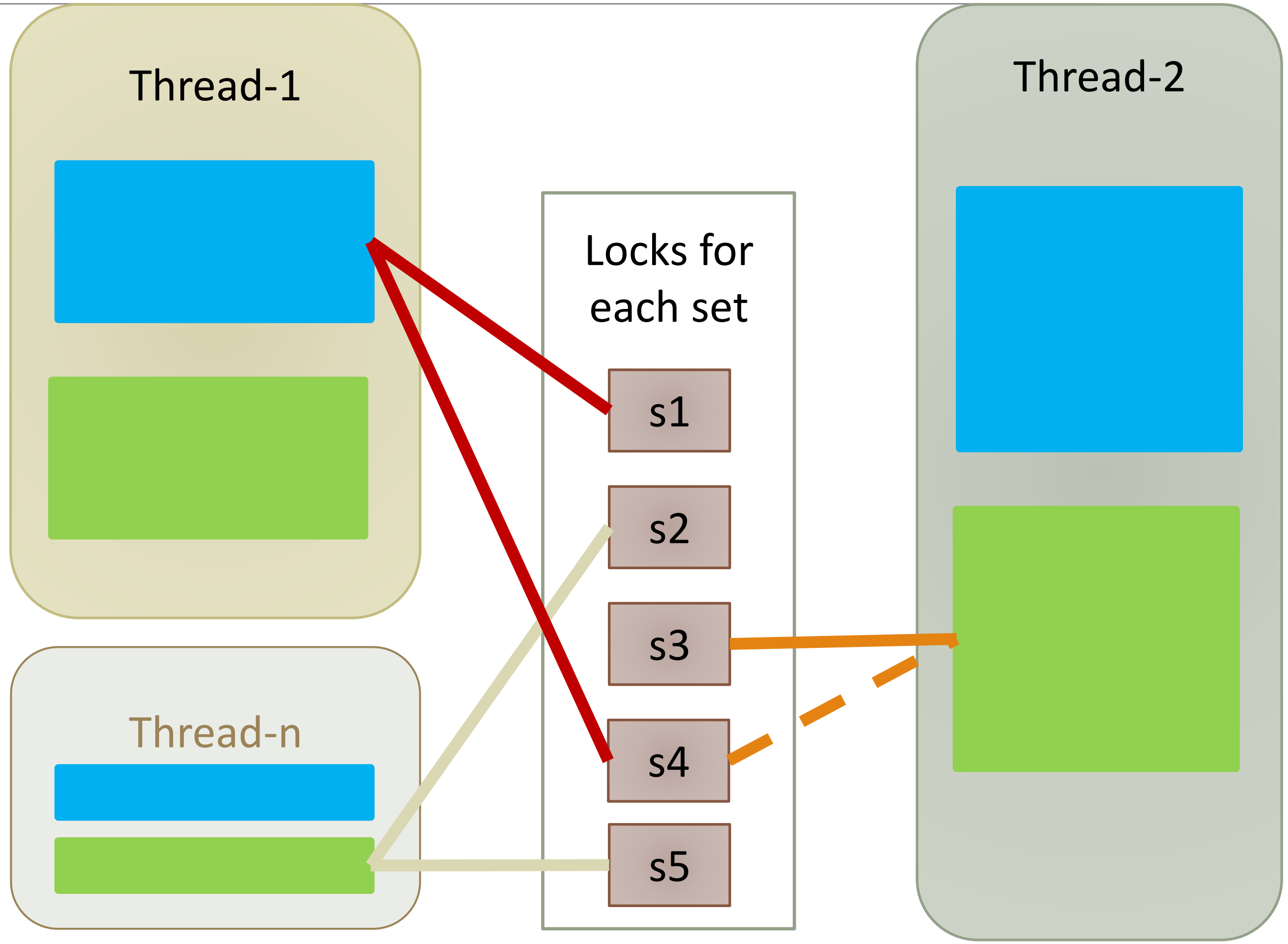
```
MaximallyLeafyForest(G)
Let F be an empty set.
For every node v in G do
        S(v) := {v}.
        d(v) := 0.
For every node v in G do
        S' := ∅.
        d' := 0.
        For every node u that is adjacent to v in G do
                If u ∉ S(v) and S(u) ∉ S' then
                        d' := d' + 1.
                        Insert S(u) into S'.
        If d(v) + d' ≥ 3 then
                For every S(u) in S' do
                        Add edge uv to F.
                        Union S(v) and S(u).
                        Update d(u) := d(u) + 1 and d(v) := d(v) + 1.
Output F as a maximally leafy forest of G.
```



3-Approximation MLST Algorithm, Lu & Ravi

1. MULTI THREADING AND EXPAND

```
MaximallyLeafyForest(G)
    Let F be an empty set.
    For every node v in G do
        S(v) := {v}.
        d(v) := 0.
    For every node v in G do
        S' := ∅.
        d' := 0.
        For every node u that is adjacent to v in G do
            If u ∉ S(v) and S(u) ∉ S' then
                d' := d' + 1.
                Insert S(u) into S'.
        If d(v) + d' ≥ 3 then
            For every S(u) in S' do
                Add edge uv to F.
                Union S(v) and S(u).
                Update d(u) := d(u) + 1 and d(v) := d(v) + 1.
    Output F as a maximally leafy forest of G.
```

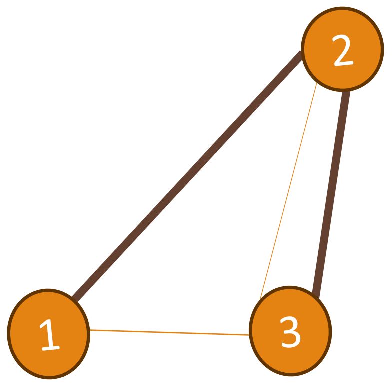


3-Approximation MLST Algorithm, Lu & Ravi

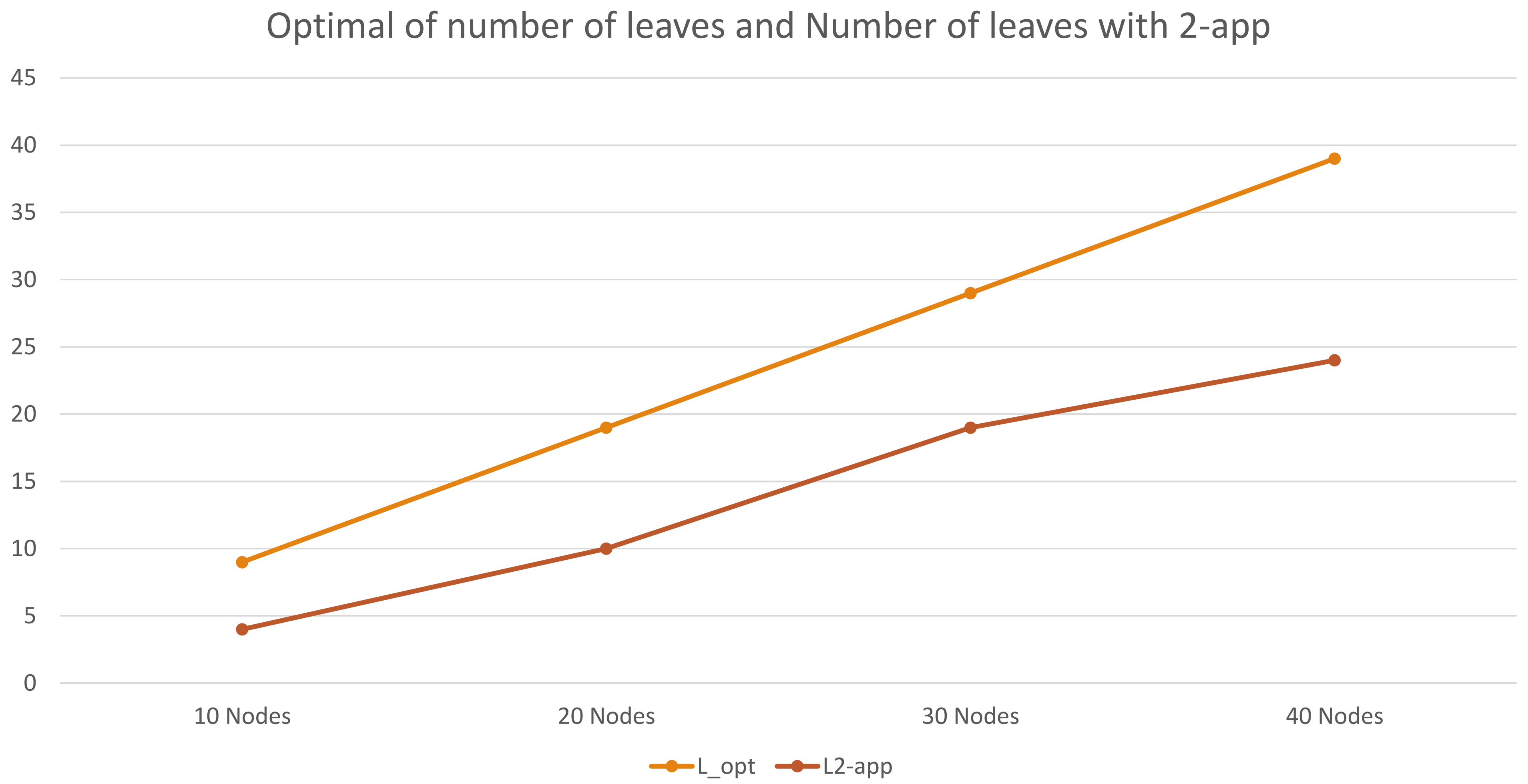
MAXIMALLYLEAFYFOREST(G)

- Let F be an empty set.
- For every node v in G do
  - $S(v) := \{v\}$.
  - $d(v) := 0$.
- For every node v in G do
  - $S' := \emptyset$.
  - $d' := 0$.
  - For every node u that is adjacent to v in G do
    - If $u \notin S(v)$ and $S(u) \notin S'$ then
      - $d' := d' + 1$.
      - Insert $S(u)$ into S'.
  - If $d(v) + d' \geq 3$ then
    - For every $S(u)$ in S' do
      - Add edge uv to F.
      - Union $S(v)$ and $S(u)$.
      - Update $d(u) := d(u) + 1$ and $d(v) := d(v) + 1$.
- ~~Output F as a maximally leafy forest of G.~~

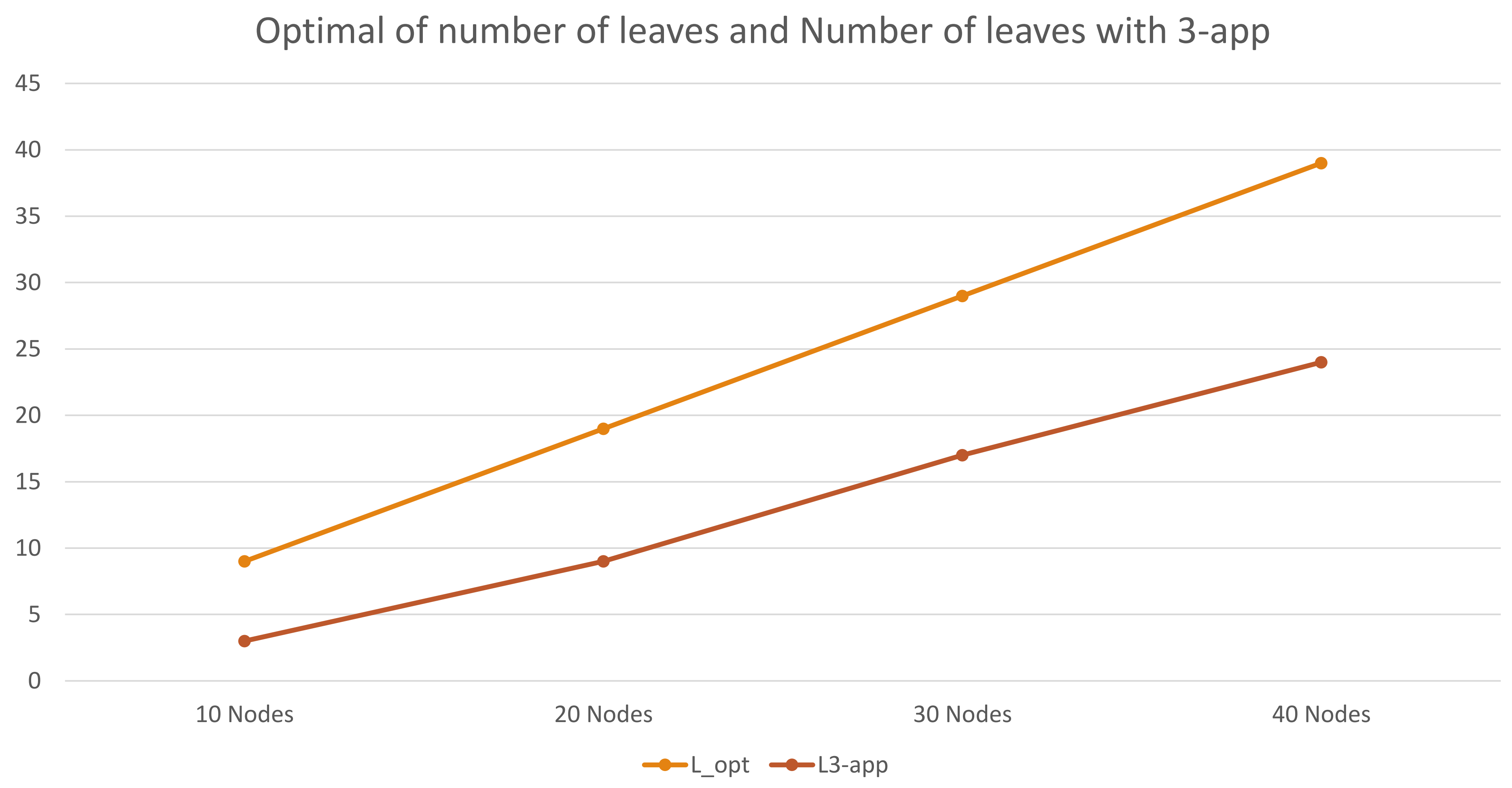
2. CONNECT THE TREES & DFS



a) Find trees
b) Connect the same way as 2-approximation



Experiments


Optimal of number of leaves and Number of leaves with 3-app
45
40
35
30
25
20
15
10
5
0
10 Nodes
20 Nodes
30 Nodes
40 Nodes
L_opt
L3-app

Conclusion

MLST Problem is MAX NSP-problem

Approximation Algorithm in polynomial time

2-approximation >=50% of optimal number of leaves in linear time

3-approximation >=33% of optimal number of leaves in linear time

More nodes, More MLST leaves

Parallelization can help reducing Time Complexity