Kernelization for Finding Lineal Topologies (Depth-First Spanning Trees) with Many or Few Leaves

Emmanuel Sam [1] Benjamin Bergougnoux[2] Petr A. Golovach[1]
Nello Blaser[1]

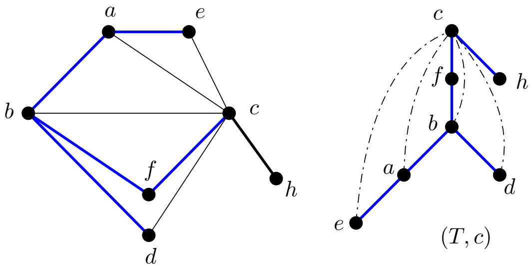
[1]Department of Informatics, University of Bergen, Norway

[2]Institute of Informatics, University of Warsaw, Poland

24th International Symposium on Fundamentals of Computation Theory

Introduction

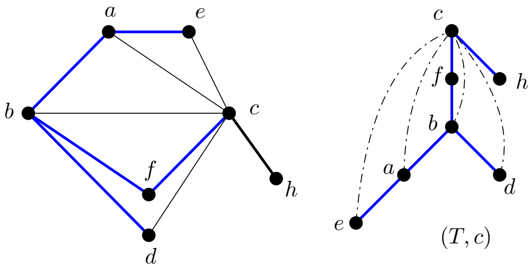
- Given a connected undirected graph G, a *depth-first spanning* (DFS) tree is a rooted spanning tree T of G with the property that
  - for every edge $xy \in E(G) \setminus E(T)$, either x is a descendant of y with respect to T, or x is an ancestor of y.



A graph G and a DFS tree (T, c) of G rooted at $c \in V(G)$

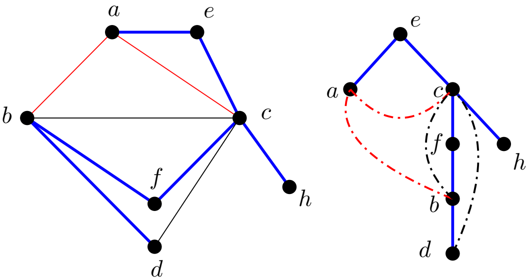
Introduction

- Given a connected undirected graph G, a *depth-first spanning* (DFS) tree is a rooted spanning tree T of G with the property that
  - for every edge $xy \in E(G) \setminus E(T)$, either x is a descendant of y with respect to T, or x is an ancestor of y.
- The edges $E(G) \setminus E(T)$ are called *back edges*.



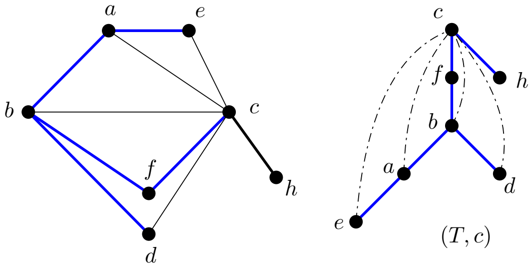
A graph G and a DFS tree (T, c) of G rooted at $c \in V(G)$

Introduction



This spanning tree (T, e) is not a DFS tree of G.

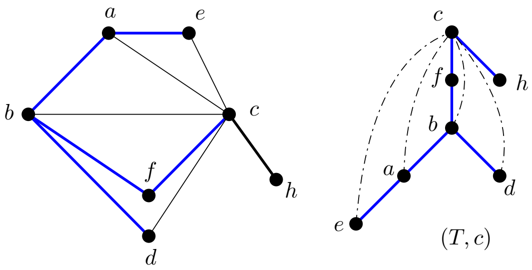
Introduction



A graph G and a DFS tree (T, c) of G rooted at $c \in V(G)$

- A DFS tree is also called *lineal topology* (LT) [Sam et al., 2023]

Introduction



A graph G and a DFS tree (T, c) of G rooted at $c \in V(G)$

- A DFS tree is also called *lineal topology* (LT) [Sam et al., 2023]
- *Leafy lineal topology* (LLT): an LT with many or few leaves.

Computing a Leafy Lineal Topology (LLT)

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$*.*
Question: *Does* G *admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?

Computing a Leafy Lineal Topology (LLT)

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**

Computing a Leafy Lineal Topology (LLT)

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**
- When parameterized by k

Computing a Leafy Lineal Topology (LLT)

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$*.*
Question: *Does* G *admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**
- When parameterized by k
  - Min-LLT is **Para-NP-hard**.

[Sam et al., 2023]

Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.

Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**
- When parameterized by k
  - Min-LLT is **Para-NP-hard**.
  - Max-LLT is **W[1]-hard**.

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**
- When parameterized by k
  - Min-LLT is **Para-NP-hard**.
  - Max-LLT is **W[1]-hard**.

Computing a Leafy Lineal Topology (LLT)

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?
- $\leqslant n - k$ *leaves* (Dual Min-LLT)?
- $\geqslant n - k$ *leaves* (Dual Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**
- When parameterized by k
  - Min-LLT is **Para-NP-hard**.
  - Max-LLT is **W[1]-hard**.

Computing a Leafy Lineal Topology (LLT)

[Sam et al., 2023]
Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?
- $\leqslant n - k$ *leaves* (Dual Min-LLT)?
- $\geqslant n - k$ *leaves* (Dual Max-LLT)?

- Min-LLT and Max-LLT are **NP-hard**
- When parameterized by k
  - Min-LLT is **Para-NP-hard**.
  - Max-LLT is **W[1]-hard**.
- Dual Min-LLT and Dual Max-LLT are **FPT** parameterized by k.

Computing a Leafy Lineal Topology (LLT)

Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?
- $\leqslant n - k$ *leaves* (Dual Min-LLT)?
- $\geqslant n - k$ *leaves* (Dual Max-LLT)?

This work:

Theorem

Dual Min-LLT *and* Dual Max-LLT *admit kernels with* $\mathcal{O}(k^3)$ *vertices.*

Computing a Leafy Lineal Topology (LLT)

Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?
- $\leqslant n - k$ *leaves* (Dual Min-LLT)?
- $\geqslant n - k$ *leaves* (Dual Max-LLT)?

This work:

Theorem

Dual Min-LLT *and* Dual Max-LLT *admit kernels with* $\mathcal{O}(k^3)$ *vertices.*

- Dual Min-LLT and Dual Max-LLT can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.

Computing a Leafy Lineal Topology (LLT)

Input: *A graph* $G = (V, E)$ *and* $k \in \mathbb{N}$.
Question: *Does G admit an DFS Tree with:*

- $\leqslant k$ *leaves* (Min-LLT)?
- $\geqslant k$ *leaves* (Max-LLT)?
- $\leqslant n - k$ *leaves* (Dual Min-LLT)?
- $\geqslant n - k$ *leaves* (Dual Max-LLT)?

This work:

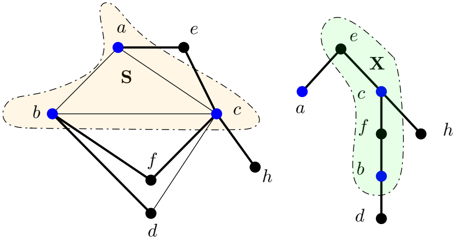
Theorem

Dual Min-LLT *and* Dual Max-LLT *admit kernels with* $\mathcal{O}(k^3)$ *vertices.*

- Dual Min-LLT and Dual Max-LLT can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT parameterized by the vertex cover τ of G admit kernels with $\mathcal{O}(\tau^3)$ vertices.

Lemma (1)

Let S be a vertex cover of G of size s. Then, every rooted spanning tree T of G has at most $2s$ internal vertices, and at most s of these internal vertices are not in S.



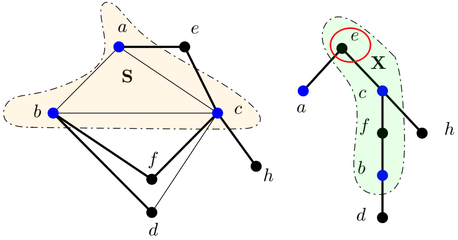
An example

Lemma (1)

Let S be a vertex cover of G of size s. Then, every rooted spanning tree T of G has at most $2s$ internal vertices, and at most s of these internal vertices are not in S.

Proof

- $\forall v \in T$, if $v \notin S$, then $\text{child}(v) \subseteq S$

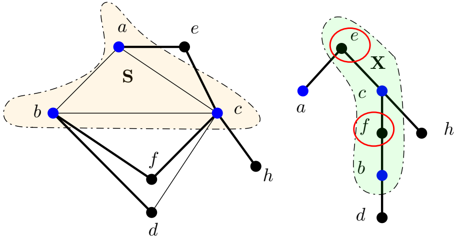


Lemma (1)

Let S be a vertex cover of G of size s. Then, every rooted spanning tree T of G has at most $2s$ internal vertices, and at most s of these internal vertices are not in S.

Proof

- $\forall v \in T$, if $v \notin S$, then $\text{child}(v) \subseteq S$
- For any distinct internal vertices u and v of T, $\text{child}(u) \cap \text{child}(v) = \emptyset$.

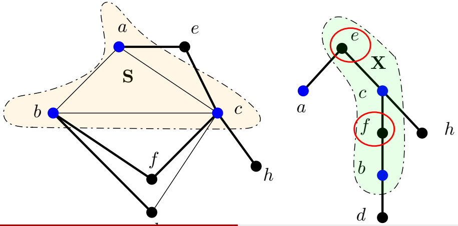


Lemma (1)

Let S be a vertex cover of G of size s. Then, every rooted spanning tree T of G has at most $2s$ internal vertices, and at most s of these internal vertices are not in S.

Proof

- $\forall v \in T$, if $v \notin S$, then $\text{child}(v) \subseteq S$
- For any distinct internal vertices u and v of T, $\text{child}(u) \cap \text{child}(v) = \emptyset$.
- Given $X \backslash S = \{x_1, \ldots, x_t\}$, $\text{child}(x_1), \ldots, \text{child}(x_t)$ are pairwise disjoint and non-empty subsets of S.



Lemma (1)

Let S be a vertex cover of G of size s. Then, every rooted spanning tree T of G has at most $2s$ internal vertices, and at most s of these internal vertices are not in S.

Proof

- $\forall v \in T$, if $v \notin S$, then $\text{child}(v) \subseteq S$
- For any distinct internal vertices u and v of T, $\text{child}(u) \cap \text{child}(v) = \emptyset$.
- Given $X \backslash S = \{x_1, \ldots, x_t\}$, $\text{child}(x_1), \ldots, \text{child}(x_t)$ are pairwise disjoint and non-empty subsets of S.
- $|X \backslash S| \leqslant s$ and $|X| \leqslant 2s$.

Lemma (2)

There is a polynomial-time algorithm that, given a vertex cover S of G of size s, outputs a graph G' with at most $s^2(s-1)+3s$ vertices such that for every integer $t \geqslant 0$:

- *G has a DFS tree with exactly t internal vertices if and only if G' has a DFS tree with exactly t internal vertices.*

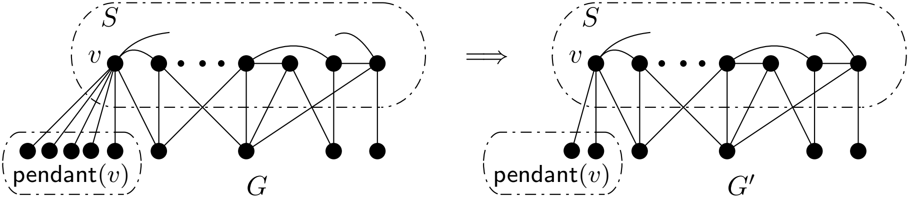
Lemma (2)

There is a polynomial-time algorithm that, given a vertex cover S of G of size s, outputs a graph G' with at most $s^2(s-1)+3s$ vertices such that for every integer $t \geqslant 0$:

- *G has a DFS tree with exactly t internal vertices if and only if G' has a DFS tree with exactly t internal vertices.*

We apply two reduction rules.

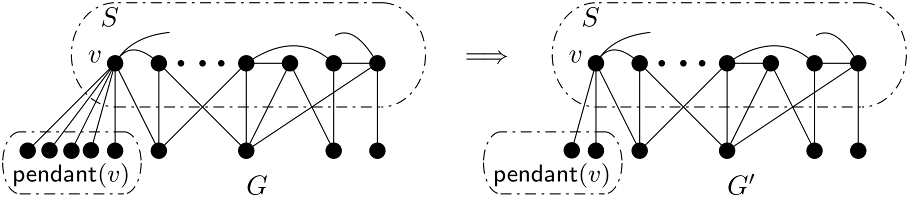
Rule 1



foreach $v \in S$ **do**
- **if** $|\text{pendant}(v)| > 2$ **then**
  - delete all but two vertices in pendant(v) from G
- **end**

end

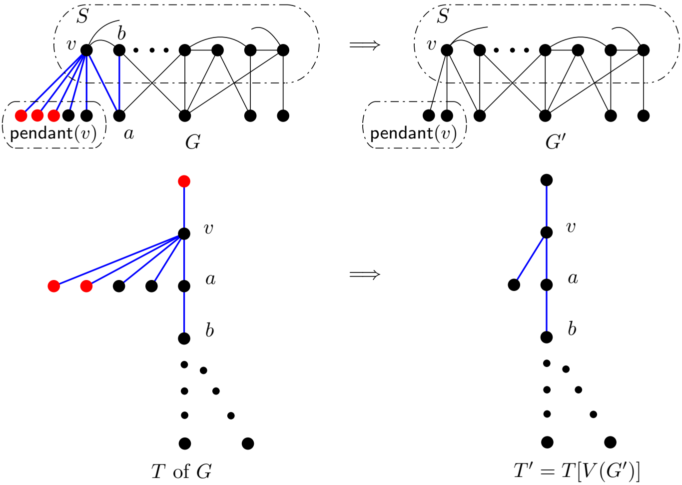
Safeness of Rule 1



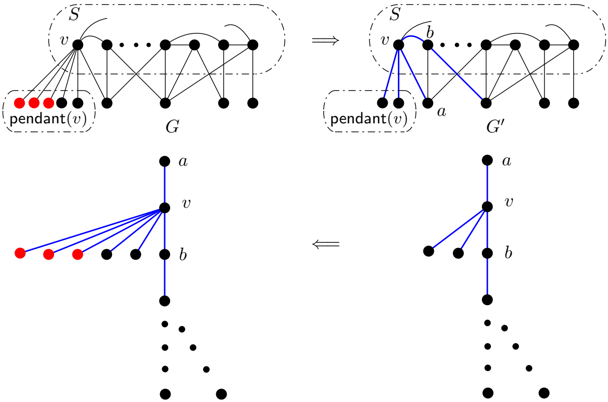
foreach $v \in S$ **do**
  if $|\text{pendant}(v)| > 2$ **then**
    delete all but two vertices in pendant(v) from G
  end
end

Observation: For each $v \in S$, if $|\text{pendant}(v)| > 2$, then at most one vertex of pendant(v) is the root and the rest are leaves of any DFS tree of G.

Safeness of Rule 1



Safeness of Rule 1

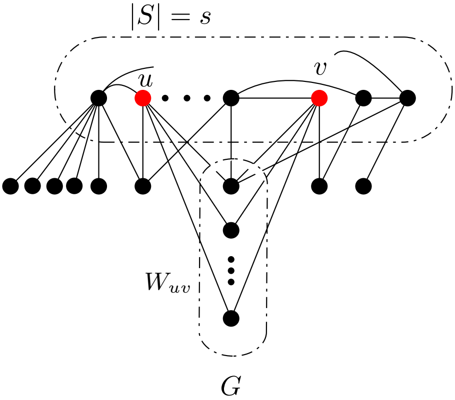


$T = T'[V(G') + V(G) \setminus V(G')]$

A DFS tree T' of G'

Rule 2

Rule 2

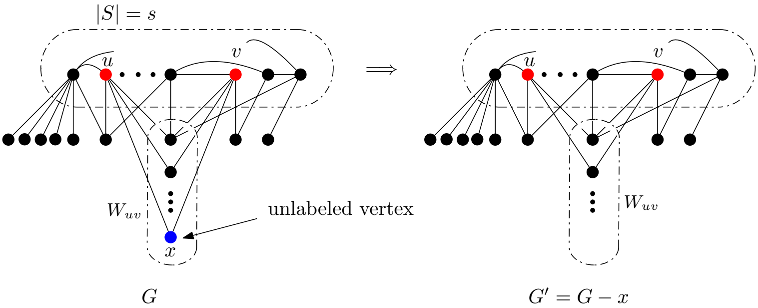


forall *pairs* $\{u, v\}$ *of distinct vertices of* S **do**

| Label at most $2s$ vertices in W_{uv}

end

Rule 2



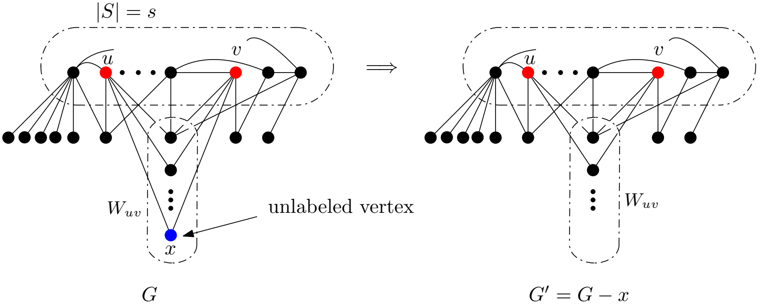
forall *pairs* $\{u, v\}$ *of distinct vertices of* S **do**

| Label at most $2s$ vertices in W_{uv}

end

Delete the unlabeled vertices of $V(G)\backslash S$ with at least two neighbors in S.

Rule 2



forall *pairs* $\{u, v\}$ *of distinct vertices of* S **do**

| Label at most $2s$ vertices in W_{uv}

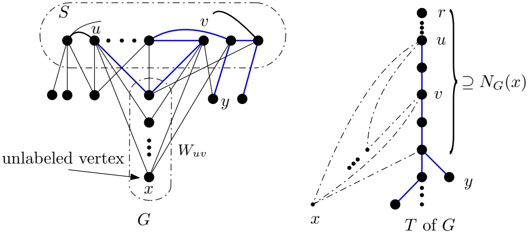
end

Delete the unlabeled vertices of $V(G) \backslash S$ with at least two neighbors in S.

Observation: *If* $|W_{uv}| \geqslant 2s + 1$ *then any* T *has at most* s *internal vertices and at least* $s + 1$ *leaves from* W_{uv}*.*

Claim 1:

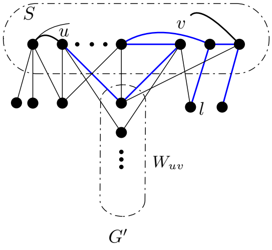
(i) For any DFS tree T of G, the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T.

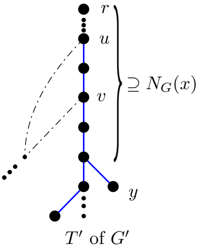


Claim 1:

(i) For any DFS tree T of G, the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T.

(ii) For any DFS tree T' of G', the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T'

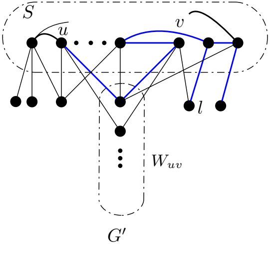


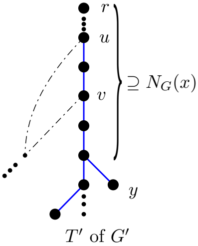


Claim 1:

(i) For any DFS tree T of G, the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T.

(ii) For any DFS tree T' of G', the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T' and none of them is a leaf of T'.





Safeness of Rule 2

Claim 1:

(i) For any DFS tree T of G, the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T.

(ii) For any DFS tree T' of G', the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T' and none of them is a leaf of T'.

Claim 2: If G has a DFS tree with t internal vertices, then G has a DFS tree T with t internal vertices such that x is a leaf of T.

Safeness of Rule 2

Claim 1:

(i) For any DFS tree T of G, the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T.

(ii) For any DFS tree T' of G', the vertices of $N_G(x)$ are vertices of a root-to-leaf path of T' and none of them is a leaf of T'.

Claim 2: If G has a DFS tree with t internal vertices, then G has a DFS tree T with t internal vertices such that x is a leaf of T.

We show that G has a DFS tree with exactly t internal vertices if and only if $G' = G - x$ has a DFS tree with exactly t internal vertices.

Let G' be the graph obtained from G after applying **Rule 1** and **Rule 2**.

Let G' be the graph obtained from G after applying **Rule 1** and **Rule 2**.

- **Rule 1** guarantees that $G' - S$ has at most $2s$ pendant vertices.
- **Rule 2** guarantees that $G' - S$ has at most $2s\binom{s}{2} = s^2(s-1)$ vertices.

Let G' be the graph obtained from G after applying **Rule 1** and **Rule 2**.

- **Rule 1** guarantees that $G' - S$ has at most $2s$ pendant vertices.
- **Rule 2** guarantees that $G' - S$ has at most $2s\binom{s}{2} = s^2(s-1)$ vertices.
- Thus, G' has at most $s^2(s-1) + 2s + s = s^2(s-1) + 3s$ vertices.

Dual Min-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:
Dual Min-LLT: Does G admit a DFS tree with $\leqslant n - k$ leaves

Dual Min-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:
Dual Min-LLT: Does G admit a DFS tree with $\leqslant n - k$ leaves (or $\geqslant k$ internal vertices)

- Construct a DFS tree T of G with S internal vertices by DFS.
- If $|S| \geqslant k$ return a trivial yes-instance of Dual Min-LLT of constant size and stop.

Recall:

Dual Min-LLT: Does G admit a DFS tree with $\leqslant n-k$ leaves (or $\geqslant k$ internal vertices)

- Construct a DFS tree T of G with S internal vertices by DFS.
- If $|S| \geqslant k$ return a trivial yes-instance of Dual Min-LLT of constant size and stop.
- Otherwise, we have a vertex cover S of G of size $s \leqslant k-1$.

Dual Min-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Min-LLT: Does G admit a DFS tree with $\leqslant n - k$ leaves (or $\geqslant k$ internal vertices)

- Construct a DFS tree T of G with S internal vertices by DFS.
- If $|S| \geqslant k$ return a trivial yes-instance of Dual Min-LLT of constant size and stop.
- Otherwise, we have a vertex cover S of G of size $s \leqslant k - 1$.
- Use S to call the algorithm from Lemma 2, which outputs G' with $\mathcal{O}(k^3)$ vertices.

Dual Min-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Min-LLT: Does G admit a DFS tree with $\leqslant n - k$ leaves (or $\geqslant k$ internal vertices)

- Construct a DFS tree T of G with S internal vertices by DFS.
- If $|S| \geqslant k$ return a trivial yes-instance of Dual Min-LLT of constant size and stop.
- Otherwise, we have a vertex cover S of G of size $s \leqslant k - 1$.
- Use S to call the algorithm from Lemma 2, which outputs G' with $\mathcal{O}(k^3)$ vertices.
- Return (G', k) and stop.

Dual Max-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Max-LLT: Does G admit a DFS tree with $\geqslant n - k$ leaves

Dual Max-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Max-LLT: Does G admit a DFS tree with $\geqslant n - k$ leaves (or $\leqslant k$ internal vertices)

- Approximate the vertex cover $\tau(G)$ of G by an inclusion-maximal matching M in G.
- If $|M| > k$, then return a trivial no-instance of Dual Max-LLT of constant size and stop.

Dual Max-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Max-LLT: Does G admit a DFS tree with $\geqslant n - k$ leaves (or $\leqslant k$ internal vertices)

- Approximate the vertex cover $\tau(G)$ of G by an inclusion-maximal matching M in G.
- If $|M| > k$, then return a trivial no-instance of Dual Max-LLT of constant size and stop.
- otherwise, S is the set of endpoints of the edges of M with $|S| \leqslant 2k$.

Dual Max-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Max-LLT: Does G admit a DFS tree with $\geqslant n - k$ leaves (or $\leqslant k$ internal vertices)

- Approximate the vertex cover $\tau(G)$ of G by an inclusion-maximal matching M in G.
- If $|M| > k$, then return a trivial no-instance of Dual Max-LLT of constant size and stop.
- otherwise, S is the set of endpoints of the edges of M with $|S| \leqslant 2k$.
- Use S to call the algorithm from Lemma 2, outputs G' with $\mathcal{O}(k^3)$ vertices.

Dual Max-LLT admits a kernel with $\mathcal{O}(k^3)$ vertices.

Recall:

Dual Max-LLT: Does G admit a DFS tree with $\geqslant n - k$ leaves (or $\leqslant k$ internal vertices)

- Approximate the vertex cover $\tau(G)$ of G by an inclusion-maximal matching M in G.
- If $|M| > k$, then return a trivial no-instance of Dual Max-LLT of constant size and stop.
- otherwise, S is the set of endpoints of the edges of M with $|S| \leqslant 2k$.
- Use S to call the algorithm from Lemma 2, outputs G' with $\mathcal{O}(k^3)$ vertices.
- Return (G', k) and stop.

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT admit polynomial kernels with $\mathcal{O}(\tau^3)$ vertices when parameterized by the vertex cover number τ of the input graph.

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT admit polynomial kernels with $\mathcal{O}(\tau^3)$ vertices when parameterized by the vertex cover number τ of the input graph.
  - **Open problem:** Do these problems have linear kernels?

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT admit polynomial kernels with $\mathcal{O}(\tau^3)$ vertices when parameterized by the vertex cover number τ of the input graph.
  - **Open problem:** Do these problems have linear kernels?
  - **Open problem:** Can they be solved by single-exponential ($2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$) FPT algorithms?

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT admit polynomial kernels with $\mathcal{O}(\tau^3)$ vertices when parameterized by the vertex cover number τ of the input graph.
  - **Open problem:** Do these problems have linear kernels?
  - **Open problem:** Can they be solved by single-exponential ($2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$) FPT algorithms?
  - **Open problem:** Are there polynomial kernels for other structural parameterization, such as the *feedback vertex* number of the input graph?

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT admit polynomial kernels with $\mathcal{O}(\tau^3)$ vertices when parameterized by the vertex cover number τ of the input graph.
  - **Open problem:** Do these problems have linear kernels?
  - **Open problem:** Can they be solved by single-exponential ($2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$) FPT algorithms?
  - **Open problem:** Are there polynomial kernels for other structural parameterization, such as the *feedback vertex* number of the input graph?

Conclusion

We showed that

- Dual Min-LLT and Dual Max-LLT parameterized by k admit kernels with $\mathcal{O}(k^3)$ vertices and can be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time.
- Min-LLT and Max-LLT admit polynomial kernels with $\mathcal{O}(\tau^3)$ vertices when parameterized by the vertex cover number τ of the input graph.
  - **Open problem:** Do these problems have linear kernels?
  - **Open problem:** Can they be solved by single-exponential ($2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$) FPT algorithms?
  - **Open problem:** Are there polynomial kernels for other structural parameterization, such as the *feedback vertex* number of the input graph?

Thank you!