On the Parameterized Complexity of the Structure of Lineal Topologies (Depth-First Spanning Trees) of Finite Graphs: The Number of Leaves

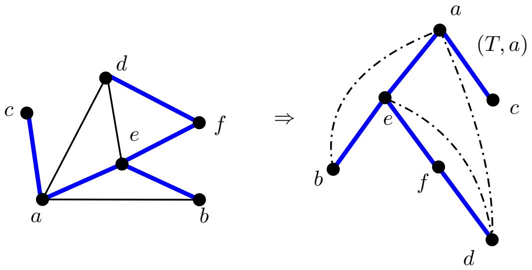
Emmanuel Sam [1] Michael Fellows[1,2] Frances Rosamond[1,2] Petr A. Golovach[1]

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

[2]Western Sydney University, Locked Bag 1797, Penrith NSW 2751, Australia

13th International Symposium on Algorithms and Complexity

- 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 $xy \in E(G) \setminus E(T)$ that is not an edge of T, either x is a descendant of y with respect to T, or x is an ancestor of y.

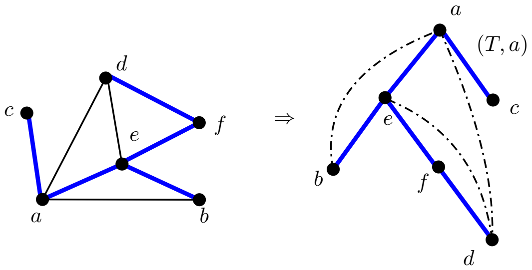


A graph G and a DFS tree (T, a) of G with back edges

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 $xy \in E(G) \setminus E(T)$ that is not an edge of T, either x is a descendant of y with respect to T, or x is an ancestor of y.

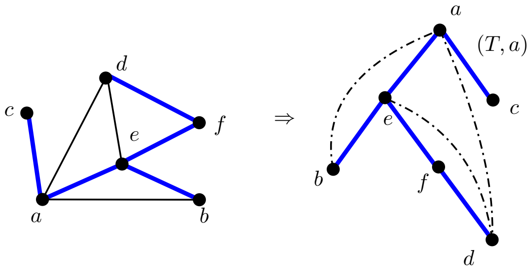
- The edges $E(G) \setminus E(T)$ are called *back edges*.



A graph G and a DFS tree (T, a) of G with back edges

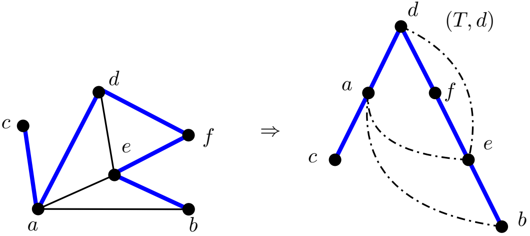
Introduction

- 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 $xy \in E(G) \backslash E(T)$ that is not an edge of T, either x is a descendant of y with respect to T, or x is an ancestor of y.
- The edges $E(G) \backslash E(T)$ are called *back edges*.
- DFS trees have also been called *lineal spanning trees*



A graph G and a DFS tree (T, a) of G with back edges

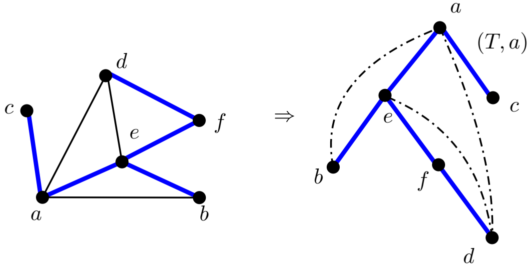
Introduction



The spanning tree (T, d) is not a DFS tree of 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 $xy \in E(G) \backslash E(T)$ that is not an edge of T, either x is a descendant of y with respect to T, or x is an ancestor of y.
- The edges $E(G) \backslash E(T)$ are called *back edges*.
- DFS trees have also been called *lineal spanning trees*



A graph G and a DFS tree (T, a) of G with back edges

Introduction

- A *lineal topology* $\mathcal{T}$, or LT for short is a graph G together with a root vertex r and an r-rooted DFS tree of G, i.e., the triple (G, r, T).

Introduction

- A *lineal topology* $\mathcal{T}$, or LT for short is a graph G together with a root vertex r and an r-rooted DFS tree of G, i.e., the triple (G, r, T).
- This notion corresponds to a point-set topology on $E(G)$ defined by:

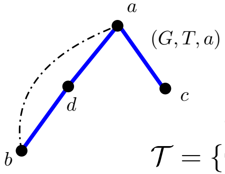
$$\mathcal{T} = \{E(G[T^{'}]) \mid T^{'} \text{ is an } r\text{-rooted subtree of the DFS tree } (T, r)\}$$

Introduction

- A *lineal topology* $\mathcal{T}$, or LT for short is a graph G together with a root vertex r and an r-rooted DFS tree of G, i.e., the triple (G, r, T).
- This notion corresponds to a point-set topology on $E(G)$ defined by:

$$\mathcal{T} = \{E(G[T^{'}]) \mid T^{'} \text{ is an } r\text{-rooted subtree of the DFS tree } (T, r)\}$$

- For example, given :



a (T, a)

$$\mathcal{T} = \{\emptyset, \{ad\}, \{ac\}, \{ad, db, ba\}, \{ad, ac\}, \{ad, db, ba, ac\}\}.$$

A topology on $E(G)$

Applications

- Planarity testing and embedding [De Fraysseix, 2008; Hopcroft and Tarjan, 1974]

Applications

- Planarity testing and embedding [De Fraysseix, 2008; Hopcroft and Tarjan, 1974]
- It can be used to define the *treedepth* of a graph [Nešetřil and de Mendez, 2012]

Applications

- Planarity testing and embedding [De Fraysseix, 2008; Hopcroft and Tarjan, 1974]
- It can be used to define the *treedepth* of a graph [Nešetřil and de Mendez, 2012]
- DFS trees have been used to structure the search space of backtracking algorithms for solving *constraint satisfaction problems* [Freuder and Quinn, 1985]

Applications

- Planarity testing and embedding [De Fraysseix, 2008; Hopcroft and Tarjan, 1974]
- It can be used to define the *treedepth* of a graph [Nešetřil and de Mendez, 2012]
- DFS trees have been used to structure the search space of backtracking algorithms for solving *constraint satisfaction problems* [Freuder and Quinn, 1985]
- The problem k-Min-HLT of asking, for a given graph G and a positive integer k, whether G has an (LT with height $h \leqslant k$) is NP-complete. [Fellows et al., 1988]

Problem Definitions

k-Minimum Leafy Lineal Topology (k-Min-LLT)

Input: A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: Does G admit an LT with $\leqslant k$ leaves?

k-Maximum Leafy Lineal Topology (k-Max-LLT)

Input: A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: Does G admit an LT with $\geqslant k$ leaves?

Problem Definitions

k-Minimum Leafy Lineal Topology (k-Min-LLT)

Input:	A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Question:	Does G admit an LT with $\leqslant k$ leaves?

k-Maximum Leafy Lineal Topology (k-Max-LLT)

Input:	A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Question:	Does G admit an LT with $\geqslant k$ leaves?

- Related to the NP-complete Min Leaf Spanning Tree and Max Leaf Spanning Tree problems [Garey and Johnson, 1990; Lu and Ravi, 1996].

Problem Definitions

k-Minimum Leafy Lineal Topology (k-Min-LLT)

Input: A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: Does G admit an LT with $\leqslant k$ leaves?

k-Maximum Leafy Lineal Topology (k-Max-LLT)

Input: A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: Does G admit an LT with $\geqslant k$ leaves?

- Related to the NP-complete Min Leaf Spanning Tree and Max Leaf Spanning Tree problems [Garey and Johnson, 1990; Lu and Ravi, 1996].
- k-Min-LLT is NP-complete.

Problem Definitions

We consider:

- k-Min-LLT and k-Max-LLT, parameterized by k, and
- their *dual parameterization*:

Problem Definitions

We consider:

- k-Min-LLT and k-Max-LLT, parameterized by k, and
- their *dual parameterization*:

Dual Min LLT

Input:	A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Parameter:	k
Question:	Does G admit an LT with $\leqslant n - k$ leaves?

Dual Max LLT

Input:	A connected undirected graph $G = (V, E)$ and $k \in \mathbb{N}$
Parameter:	k
Question:	Does G admit an LT with $\geqslant n - k$ leaves?

Our Results

- The k-Min-LLT problem is para-NP-hard parameterized by k.

Our Results

- The k-Min-LLT problem is para-NP-hard parameterized by k.
- We show the following theorem by a *parameterized reduction* from the Multicolored Independence Set (MIS) problem.

Theorem

The k-Max-LLT *problem is W[1]-hard parameterized by* k

Our Results

- The k-Min-LLT problem is para-NP-hard parameterized by k.
- We show the following theorem by a *parameterized reduction* from the Multicolored Independence Set (MIS) problem.

Theorem

The k-Max-LLT problem is W[1]-hard parameterized by k and NP-complete, when considered classically

- Dual Min-LLT and Dual Max-LLT are NP-complete when considered classically.

Our Results

- The k-Min-LLT problem is para-NP-hard parameterized by k.
- We show the following theorem by a *parameterized reduction* from the Multicolored Independence Set (MIS) problem.

Theorem

The k-Max-LLT *problem is W[1]-hard parameterized by k and NP-complete, when considered classically*

- Dual Min-LLT and Dual Max-LLT are NP-complete when considered classically.

Theorem

Dual Min-LLT *and* Dual Max-LLT *are FPT parameterized by* k*.*

Proof:

We present a parameterized reduction from:

Multicolored Independent Set (MIS)

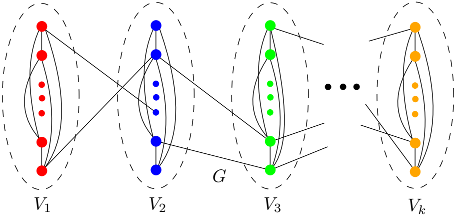
Input:	A graph $G = (V, E)$, and $f : V \to [1, k]$ with $k \in \mathbb{N}$.
Parameter:	k
Question:	Does G contain a k-colored independent set?

We assume that each color class V_i, for $i \in [1, k]$, induces a clique.

W[1]-Hardness of k-Max-LLT Parameterized by k

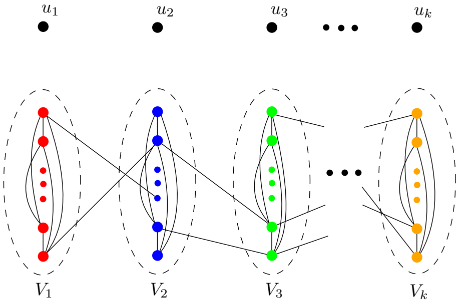
Proof:

- Given an instance (G, k) of MIS.



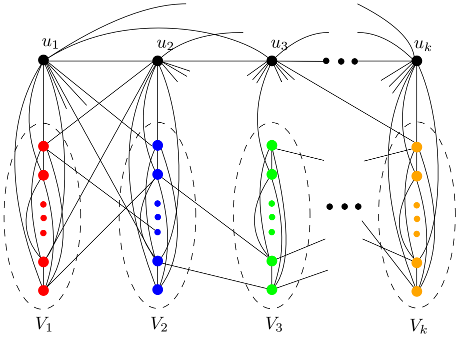
Proof:

- Given an instance (G, k) of MIS.
- We construct an instance (G', k') of k-Max-LLT with $k' = k$.



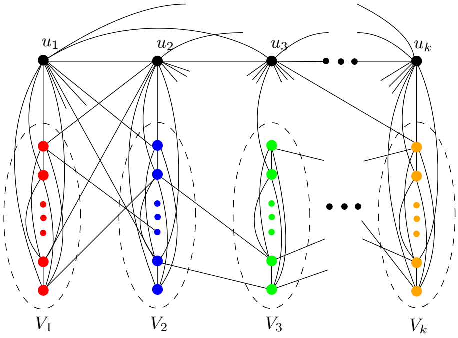
Proof:

- Given an instance (G, k) of MIS.
- We construct an instance (G', k') of k-Max-LLT with $k' = k$.



Proof:

- Given an instance (G, k) of MIS.
- We construct an instance (G', k') of k-Max-LLT with $k' = k$.
- (G', k') can be constructed in polynomial time.



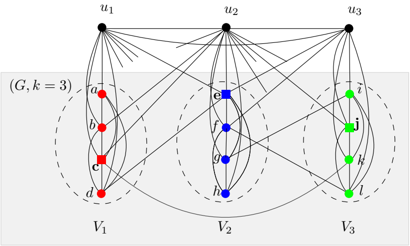
W[1]-Hardness of k-Max-LLT Parameterized by k

Lemma

G has a k-colored independent set $\Rightarrow$ G' admits an LT with at least k leaves.

Proof.

- Let $X = \{x_1, ..., x_k\}$ be a k-colored independent set in G.



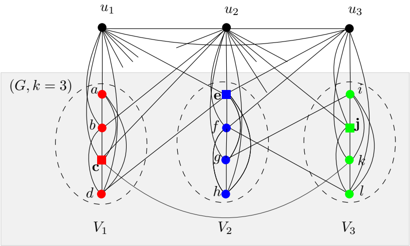
G' and a k-colored independence set $X = \{c, e, j\}$ in G

Lemma

G has a k-colored independent set $\Rightarrow$ G' admits an LT with at least k leaves.

Proof.

- Let $X = \{x_1, ..., x_k\}$ be a k-colored independent set in G.
- A DFS of G' that excludes the vertices in X until all the vertices in $V(G') \backslash X$ have been visited yields an LT with the vertices in X as its leaves.



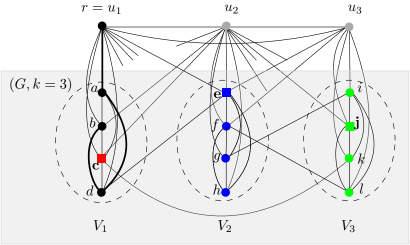
G' and a k-colored independence set $X = \{c, e, j\}$ in G

Lemma

G has a k-colored independent set $\Rightarrow$ G' admits an LT with at least k leaves.

Proof.

- Let $X = \{x_1, ..., x_k\}$ be a k-colored independent set in G.
- A DFS of G' that excludes the vertices in X until all the vertices in $V(G') \setminus X$ have been visited yields an LT with the vertices in X as its leaves.



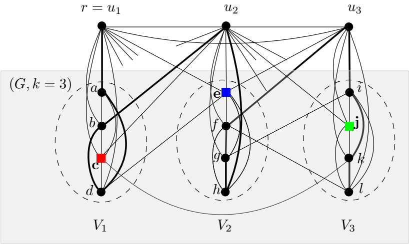
G' and a k-colored independence set $X = \{c, e, j\}$ in G

Lemma

G has a k-colored independent set $\Rightarrow$ G' admits an LT with at least k leaves.

Proof.

- Let $X = \{x_1, ..., x_k\}$ be a k-colored independent set in G.
- A DFS of G' that excludes the vertices in X until all the vertices in $V(G')\backslash X$ have been visited yields an LT with the vertices in X as its leaves.



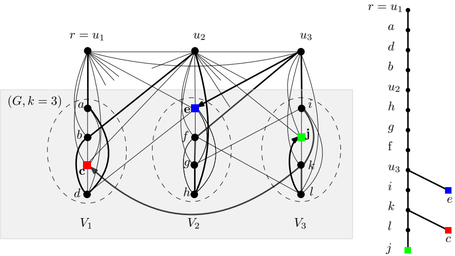
G' and a k-colored independence set $X = \{c, e, j\}$ in G

Lemma

G has a k-colored independent set $\Rightarrow G'$ admits an LT with at least k leaves.

Proof.

- Let $X = \{x_1, ..., x_k\}$ be a k-colored independent set in G.
- A DFS of G' that excludes the vertices in X until all the vertices in $V(G')\backslash X$ have been visited yields an LT with the vertices in X as its leaves.



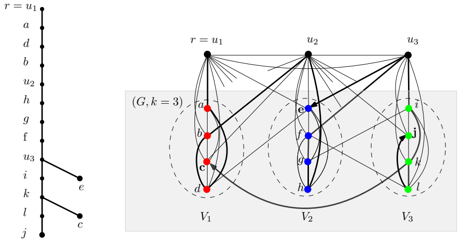
G' with $X = \{c, e, j\}$, and the resulting DFS tree T

Lemma

G has a k-colored independent set $\Leftarrow$ G' admits an LT with at least k leaves.

Proof.

- Suppose that $k \geqslant 2$, and G' admits a DFS tree T' with at least k leaves.

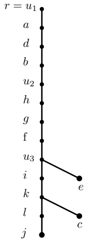


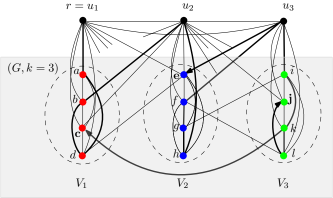
Lemma

G has a k-colored independent set $\Leftarrow$ G' admits an LT with at least k leaves.

Proof.

- Suppose that $k \geqslant 2$, and G' admits a DFS tree T' with at least k leaves. Let $X = \{x_1, \ldots, x_k\}$ be the set of leaves of T'.





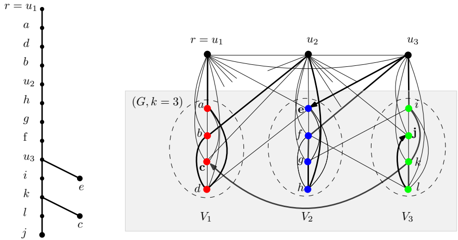
W[1]-Hardness of k-Max-LLT Parameterized by k

Lemma

G has a k-colored independent set $\Leftarrow$ G' admits an LT with at least k leaves.

Proof.

- Suppose that $k \geqslant 2$, and G' admits a DFS tree T' with at least k leaves. Let $X = \{x_1, ..., x_k\}$ be the set of leaves of T'.
- *Claim 1: Each vertex in X belongs to at most one color class V_i in G.*



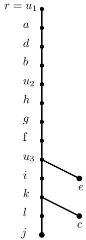
W[1]-Hardness of k-Max-LLT Parameterized by k

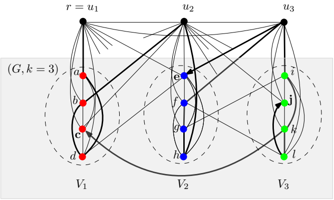
Lemma

G has a k-colored independent set $\Leftarrow$ G' admits an LT with at least k leaves.

Proof.

- Suppose that $k \geqslant 2$, and G' admits a DFS tree T' with at least k leaves. Let $X = \{x_1, ..., x_k\}$ be the set of leaves of T'.
- *Claim 1: Each vertex in X belongs to at most one color class V_i in G.*
- *Claim 2: None of the vertices in X belongs to $U = \{u_1, \ldots, u_k\}$.*





FPT Algorithms

Dual Max-LLT and Dual Min-LLT parameterized by k

Method

We employ the following theorem:

Theorem (Courcelle, 1990)

Given $k \in \mathbb{N}$ and a fixed monadic second-order logic (MSO) formula ϕ of length ℓ expressing a graph property, there is an algorithm that takes G with treewidth at most k as input and decides whether $G \models \phi$ in time $\mathcal{O}\big(f(\ell, k) \cdot n\big)$, for some computable function f.

We express in MSO_1 (a version of MSO) the property of having:

- an LT with at most $n - k$ leaves
- an LT with at least $n - k$ leaves.

Plan:

Given a DFS tree T resulting from any DFS of the graph G, T either:

- solves the problem trivially, or
- yields a bounded path decomposition.

FPT Algorithm for Dual Max-LLT

Observation

For any given graph G and $k \in \mathbb{N}$, if G is a YES-instance of Dual Max-LLT, *then G admits an LT with height at most k.*

FPT Algorithm for Dual Max-LLT

Observation

For any given graph G and $k \in \mathbb{N}$, if G is a YES-instance of Dual Max-LLT, *then G admits an LT with height at most k.*

Lemma

Given a graph G and $k \in \mathbb{N}$, if G admits an LT of height at most k, then the length of any path in G is at most $2^{k+1} - 2$.

FPT Algorithm for Dual Max-LLT

Observation

For any given graph G and $k \in \mathbb{N}$, if G is a YES-instance of Dual Max-LLT*, then G admits an LT with height at most k.*

Lemma

Given a graph G and $k \in \mathbb{N}$, if G admits an LT of height at most k, then the length of any path in G is at most $2^{k+1} - 2$.

Algorithm

Let T be a DFS tree of G given by DFS.

- If the height h of T is more than $2^{k+1} - 2$, then return NO.

FPT Algorithm for Dual Max-LLT

Observation

For any given graph G and $k \in \mathbb{N}$, if G is a YES-instance of Dual Max-LLT, *then G admits an LT with height at most k.*

Lemma

Given a graph G and $k \in \mathbb{N}$, if G admits an LT of height at most k, then the length of any path in G is at most $2^{k+1} - 2$.

Algorithm

Let T be a DFS tree of G given by DFS.

- If the height h of T is more than $2^{k+1} - 2$, then return NO.
- Otherwise, use T to construct a path decomposition of G of pathwidth at most $2^{k+1} - 1$.

FPT Algorithm for Dual Min-LLT

The algorithm is similar to that for Dual Max-LLT

FPT Algorithm for Dual Min-LLT

The algorithm is similar to that for Dual Max-LLT

Difference:

- If the number of leaves of T is at most $n - k$, then return YES.
- Otherwise, use T to construct a path decomposition of G of pathwidth at most k.

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.
  - **Open problem:** Is the problem in XP?

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.
  - **Open problem:** Is the problem in XP?
- We showed that Dual Min-LLT and Dual Max-LLT are FPT parameterized by k.

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.
  - **Open problem:** Is the problem in XP?
- We showed that Dual Min-LLT and Dual Max-LLT are FPT parameterized by k.
  - **Open problem:** Is there a single-exponential time algorithm that solves the problems explicitly via dynamic programming?

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.
  - **Open problem:** Is the problem in XP?
- We showed that Dual Min-LLT and Dual Max-LLT are FPT parameterized by k.
  - **Open problem:** Is there a single-exponential time algorithm that solves the problems explicitly via dynamic programming?
- k-Min-HLT (LT with height $h \leqslant k$) is FPT with respect to k by our Theorem for Dual Max-LLT.

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.
  - **Open problem:** Is the problem in XP?
- We showed that Dual Min-LLT and Dual Max-LLT are FPT parameterized by k.
  - **Open problem:** Is there a single-exponential time algorithm that solves the problems explicitly via dynamic programming?
- k-Min-HLT (LT with height $h \leqslant k$) is FPT with respect to k by our Theorem for Dual Max-LLT.
  - **Open problem:** Dual Min-HLT (LT with height $h \leqslant n - k$), parameterized by k.

Conclusion

- We showed that k-Max-LLT is hard for W[1] parameterized k.
  - **Open problem:** Is the problem in XP?
- We showed that Dual Min-LLT and Dual Max-LLT are FPT parameterized by k.
  - **Open problem:** Is there a single-exponential time algorithm that solves the problems explicitly via dynamic programming?
- k-Min-HLT (LT with height $h \leqslant k$) is FPT with respect to k by our Theorem for Dual Max-LLT.
  - **Open problem:** Dual Min-HLT (LT with height $h \leqslant n - k$), parameterized by k.

Thank you!