Computing Distances on Graph Associahedra is Fixed-parameter Tractable

Luís Felipe I. Cunha
Instituto de Computação, Universidade Federal Fluminense, Brasil

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, France

Uéverton S. Souza
Instituto de Computação, Universidade Federal Fluminense, Brasil
IMPA - Instituto de Matemática Pura e Aplicada, Brasil

Mario Valencia-Pabon
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

An elimination tree of a connected graph G is a rooted tree on the vertices of G obtained by choosing a root v and recursing on the connected components of $G - v$ to obtain the subtrees of v. The graph associahedron of G is a polytope whose vertices correspond to elimination trees of G and whose edges correspond to tree rotations, a natural operation between elimination trees. These objects generalize associahedra, which correspond to the case where G is a path. Ito et al. [ICALP 2023] recently proved that the problem of computing distances on graph associahedra is NP-hard. In this paper we prove that the problem, for a general graph G, is fixed-parameter tractable parameterized by the distance k. Prior to our work, only the case where G is a path was known to be fixed-parameter tractable. To prove our result, we use a novel approach based on a marking scheme that restricts the search to a set of vertices whose size is bounded by a (large) function of k.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Combinatorics.

Keywords and phrases graph associahedra, elimination tree, rotation distance, parameterized complexity, fixed-parameter tractable algorithm, combinatorial shortest path, reconfiguration.

Related Version A conference version of this article will appear in the proceedings of *ICALP 2025*. We would like to thank the reviewers for their helpful remarks.

Funding *Luís Felipe I. Cunha*: FAPERJ-JCNE (E-26/201.372/2022) and CNPq-Universal (406173/2021-4).
Ignasi Sau: French project ELiT (ANR-20-CE48-0008-01).
Uéverton S. Souza: CNPq (312344/2023-6), and FAPERJ (E-26/201.344/2021).
Mario Valencia-Pabon: French project Abysm (ANR-23-CE48-0017).

1 Introduction

Given a connected and undirected graph G, an *elimination tree* T of G is any rooted tree that can be defined recursively as follows. If $V(G) = \{v\}$, then T consists of a single root vertex v. Otherwise, a vertex $v \in V(G)$ is chosen as the root of T, and an elimination tree is created for each connected component of $G - v$. Each root of these elimination trees of $G - v$ is a child of v in T. For a disconnected graph G, an *elimination forest* of G is the disjoint union of elimination trees of the connected components of G. Equivalently, an elimination forest of a graph G is a rooted forest F (that is, a forest with a root in every connected component) on vertex set $V(G)$ such that for each edge $uv \in E(G)$, vertex u is an ancestor of vertex v in F, or vice versa.

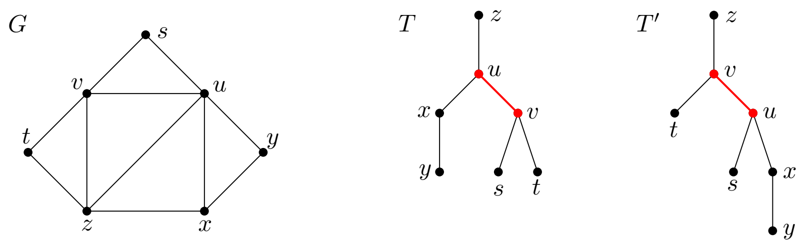


Figure 1 A graph G and two of its elimination trees T and T', where the second one is obtained from the first one by the rotation of edge uv (in red).

Figure 1 illustrates an example of two elimination trees T and T' of a graph G. With slight (and standard) abuse of notation, we use the same labels for the vertices of a graph G and any of its elimination trees. Note that an elimination tree is unordered, i.e., there is no ordering associated with the children of a vertex in the tree. Similarly, there is no ordering among the elimination trees in an elimination forest.

Elimination trees have been studied extensively in various contexts, including graph theory, combinatorial optimization, polyhedral combinatorics, data structures, or VLSI design; see the recent paper by Cardinal, Merino, and Mütze [7] and the references therein. In particular, elimination trees play a prominent role in structural and algorithmic graph theory, as they appear naturally in several contexts. As a relevant example, the *treedepth* of a graph G is defined as the minimum height of an elimination forest of G [32].

Given a class of combinatorial objects and a "local change" operation between them, the corresponding *flip graph* has as vertices the combinatorial objects, and its edges connect pairs of objects that differ by the prescribed change operation. In this article, we focus on the case where this class of combinatorial objects is the set of elimination forests of a graph G. For these objects, the commonly considered "local change" operation is that of *edge rotation* defined as follows, where we suppose for simplicity that G is connected. Given an elimination tree T of a graph G, the *rotation* of an edge $uv \in E(T)$, with u being the parent of v, creates another elimination of G, denoted by $\mathsf{rot}(T, uv)$, obtained, informally, by just swapping the choice of u and v in the recursive definition of T (that is, in the so-called *elimination ordering*), and updating the parent of the subtrees rooted at v accordingly; see Figure 2 for an illustration. The formal definition can be found in Section 3 (cf. Definition 2).

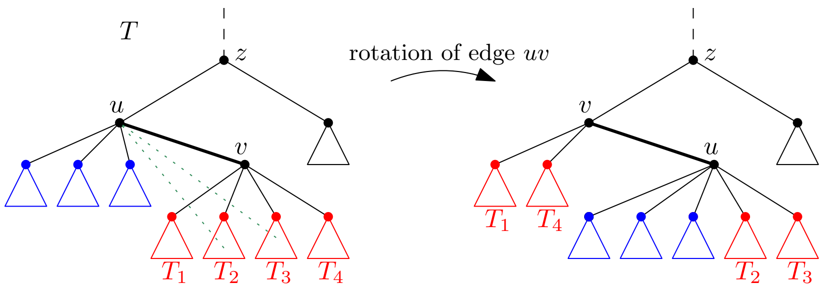


Figure 2 On the left: An elimination tree T of a graph G with adjacent vertices u and v. Vertex v has four subtrees, and two of them, namely T_2 and T_3, contain vertices adjacent to vertex u in G. On the right: Elimination tree resulting from T by applying the rotation of uv. Since both $G[V(T_2) \cup \{u\}]$ and $G[V(T_3) \cup \{u\}]$ are connected, T_2 and T_3 become subtrees of u in $\mathsf{rot}(T, uv)$.

For example, in Figure 1, $T' = \mathsf{rot}(T, uv)$. The definition of the rotation operation clearly implies that it is self-inverse with respect to any edge, that is, for any elimination tree T

of a graph G and any edge $uv \in E(T)$, it holds that $T = \mathsf{rot}(\mathsf{rot}(T, uv), vu)$. The *rotation distance* between two elimination trees T, T' of a graph G, denoted by $\mathsf{dist}(T, T')$, is the minimum number of rotations it takes to transform T into T'. The self-invertibility property of rotations discussed above implies that $\mathsf{dist}(T, T') = \mathsf{dist}(T', T)$.

It is well known that for any graph G, the flip graph of elimination forests of G under tree rotations is the skeleton of a polytope, referred to as the *graph associahedron* $\mathcal{A}(G)$ and that was introduced by Carr, Devadoss, and Postnikov [11, 17, 36]. For the particular cases of G being a complete graph, a cycle, a path, a star, or a disjoint union of edges, $\mathcal{A}(G)$ is the permutahedron, the cyclohedron, the (standard) associahedron, the stellohedron, or the hypercube, respectively; see the introduction of [7] for nice figures to illustrate these objects.

Graph associahedra naturally generalize associahedra, which correspond to the particular case where G is a path. As mentioned in [7], the associahedron has a rich history and literature, connecting computer science, combinatorics, algebra, and topology [23, 27, 37, 39]. See the introduction of the paper by Ceballos, Santos, and Ziegler [12] for a historical account. In an associahedron, each vertex corresponds to a binary tree over a set of n elements, and each edge corresponds to a rotation operation between two binary trees, an operation used in standard online binary search tree algorithms [1, 22, 40]. Binary trees are in bijection with many other Catalan objects such as triangulations of a convex polygon, well-formed parenthesis expressions, Dyck paths, etc. [41]. For instance, in triangulations of a convex polygon, the rotation operation maps to another simple operation, known as a flip, which removes the diagonal of a convex quadrilateral formed by two triangles and replaces it by the other diagonal.

Related work. Distances on graph associahedra have been object of intensive study. Probably, the most studied parameter is the diameter, that is, the maximum distance between two vertices of $\mathcal{A}(G)$. A number of influential articles either determine the diameter exactly, or provide lower and upper bounds, or asymptotic estimates, for the cases where the underlying graph G is a path [37, 39], a star [31], a cycle [38], a tree [6, 31], a complete bipartite graph [8], a caterpillar [3], a trivially perfect graph [8], a graph in which some width parameter (such as treedepth or treewidth) is bounded [8], or a general graph [31].

Our focus is on the algorithmic problem of determining the distance between two vertices of $\mathcal{A}(G)$, or equivalently, determining the rotation distance between two given elimination trees of a graph G. There are very few cases where this problem is known to be solvable in polynomial time, namely when G is a complete graph (folklore), a star [9], or a complete split graph [9]. The complexity of the case where G is a path is a notorious long-standing open problem. On the positive side, for G being a path, there exist a polynomial-time 2-approximation algorithm [15] and several fixed-parameter tractable (FPT) algorithms when the distance is the parameter [14, 25, 26, 28, 30]. It is worth mentioning that there are some hardness results on generalized settings [2, 29, 34] and polynomial-time algorithms for some type of restricted rotations [13].

Cardinal et al. [5] asked whether computing distances on general graph associahedra is NP-hard. Very recently, this question was answered positively by Ito et al. [24].

Our result. The NP-hardness result of Ito et al. [24] (see also [10]) paves the way for studying the parameterized complexity of the problem of computing distances on graph associahedra. Thus, in this article we are interested in the following parameterized problem, where we consider the natural parameter, that is, the desired distance.

ROTATION DISTANCE

Instance: A graph G, two elimination trees T and T' of G, and a positive integer k.
Parameter: k.
Question: Is the rotation distance between T and T' at most k?

As mentioned above, ROTATION DISTANCE was known to be polynomial-time solvable on complete graphs, stars, and complete split graphs [9], and FPT algorithms were only known on paths [14,25,26,28,30]. In this article we vastly generalize the known results by providing an FPT algorithm to solve the ROTATION DISTANCE problem for a general input graph G. More precisely, we prove the following theorem.

▶ **Theorem 1.** *The* ROTATION DISTANCE *problem can be solved in time* $f(k) \cdot |V(G)|$*, with* $f(k) = k^{k \cdot 2^{2^{\cdot^{\cdot^{2^{\mathcal{O}(k^2)}}}}}}$*, where the tower of exponentials has height at most* $(3k+1)4k = \mathcal{O}(k^2)$*.*

In particular, Theorem 1 yields a linear-time algorithm to solve ROTATION DISTANCE for every fixed value of the distance k. To the best of our knowledge, this is the first positive algorithmic result for the general ROTATION DISTANCE problem (i.e., with no restriction on the input graph G), and we hope that it will find algorithmic applications in the many contexts where graph associahedra arise naturally [7,11,17,24,31,36]. Our result can also been seen through the lens of the very active area of the parameterized complexity of graph reconfiguration problems; see [4] for a recent survey.

Organization. In Section 2 we present an overview of the main ideas of the algorithm of Theorem 1, which may serve as a road map to read the rest of the article. In Section 3 we provide standard preliminaries about graphs and parameterized complexity and fix our notation, in Section 4 we formally present our FPT algorithm (split into several subsections), and in Section 5 we discuss several directions for further research.

2 Overview of the main ideas of the algorithm

Our approach to obtain an FPT algorithm to solve ROTATION DISTANCE is novel, and does not build on previous work. Given two elimination trees T and T' of a connected graph G and a positive integer k, our goal is to decide whether there exists what we call an *ℓ-rotation sequence* σ from T to T', for some $\ell \leq k$, that is, an ordered list of ℓ edges to be rotated in order to obtain T' from T, going through the intermediate elimination trees $T_1, \ldots, T_{\ell-1}$ (all of the same graph G); see Section 3 for the formal definition. At a high level, our approach is based on identifying a subset of *marked vertices* $M \subseteq V(T)$, of size bounded by a function of k, so that we can assume that the desired rotation sequence σ uses only vertices in M. Once this is proved, an FPT algorithm follows directly by applying brute force and guessing all possible rotations using vertices in M.

A crucial observation (cf. Observation 3) is that a rotation may change the set of children of at most three vertices (but the parent of arbitrarily many vertices, such as the roots of T_2 and T_3 in Figure 2). Motivated by this, we say that a vertex $v \in V(T)$ is *(T,T')-children-bad* if its set of children in T is different from its set of children in T'. By Observation 3, we may assume (cf. Observation 5) that we are dealing with an instance in which the number of (T,T')-children-bad vertices is at most $3k$.

In a first step, we prove (cf. Lemma 7) that we can assume that the desired sequence σ of at most k rotations to transform T into T' uses only vertices lying in the union of the balls of radius $2k$ around (T, T')-children-bad vertices of T, which we denote by B_{cb}. The proof of Lemma 7 exploits, in particular, the fact that a rotation may increase or decrease vertex distances (in the corresponding trees) by at most one (cf. **??**). This is then used to show that if a rotation uses some vertex outside of B_{cb}, then it can be "simplified" into another one that does not (cf. **??**).

By Lemma 7, we restrict henceforth to rotations using only vertices in B_{cb}. We can consider each connected component Z of $T[B_{\mathsf{cb}}]$, since it can be easily seen that we can assume that there are at most k of them. By definition of B_{cb}, the diameter of such a component Z is $\mathcal{O}(k^2)$ (cf. Equation 1). Thus, the "only" obstacle to obtain the desired FPT algorithm is that the vertices in B_{cb} can have an arbitrarily large degree. Note that in the particular case where the underlying graph G has bounded degree, the maximum degree of any elimination tree of G is bounded, and therefore in that case $|B_{\mathsf{cb}}|$ is bounded by a function of k, and an FPT algorithm follows immediately. To the best of our knowledge, this result was not known for graphs other than paths (albeit, with a better running time than the one that results from just brute-forcing on the set B_{cb}, which is of the form $2^{2^{\mathcal{O}(k)}} \cdot |V(G)|$).

Our strategy to deal with high-degree vertices in B_{cb} is as follows. Fix one connected component Z of $T[B_{\mathsf{cb}}]$. Our goal is to identify a subset $M_Z \subseteq V(Z)$ of size bounded by a function of k, such that we can restrict our search to rotations using only vertices in M_Z. To find such a "small" set $M_Z \subseteq V(Z)$, we define the notion of *type* of a vertex $v \in V(Z)$, in such a way that the number of different types is bounded by a function of k. Then, we will prove via our marking algorithm that it is enough to keep in M_Z, for each type, a number of vertices bounded again by a function of k.

Before defining the types, we need to define the *trace* of a vertex v in Z. To get some intuition, look at the rotation depicted in Figure 2. Note that, for each of the subtrees $T_1, \ldots, T_4$ that are children of v in T, what determines whether they are children of u or v in the resulting subtree is whether some vertex in T_i is adjacent to u or not. Iterating this idea, if we are about to perform at most k rotations starting from T, then the behavior of such a subtree T_i, assuming that no vertex of it is used by a rotation, is determined by its neighborhood in a set of ancestors of size at most the diameter of Z, and this is what the trace is intended to represent. That is, the trace of a vertex v in Z, denoted by $\mathsf{trace}(T, Z, v)$, captures "abstractly" the neighborhood of the whole subtree rooted at v among (the ordered set of) its ancestors within the designated vertex set $Z \subseteq V(T)$; see Definition 8 for the formal definition of trace and Figure 3 for an example. We stress that, when considering the neighborhood in the set of ancestors, we look at the whole subtree $T(v)$ rooted at v, and not only at its restriction to the set Z.

Equipped with the definition of trace, we can define the notion of type, which is somehow involved (cf. Definition 9) and whose intuition behind is the following. For our marking algorithm to make sense, we want that if two vertices v, v' with the same parent (called T-siblings) have the same type (within Z), denoted by $\tau(T, Z, v) = \tau(T, Z, v')$, and an ℓ-rotation sequence σ from T to T' uses some vertex from $T(v)$ but uses no vertex in $T(v')$, then there exists another ℓ-rotation sequence σ' from T to T' that uses vertices in $T(v')$ instead of those in $T(v)$. To guarantee this replacement property, we need a stronger condition than just v and v' having the same trace. Informally, we need them to have the same "variety of traces among their children within Z". More formally, this leads to a recursive definition where, in the leaves of Z (that are not necessarily leaves of T), the type corresponds to the trace, and for non-leaves, the type is defined by the trace and by the number of children of each possible lower type. Note that, a priori, the number of children of a given type may be unbounded, which would rule out the objective of bounding the number of types as a function of k. To overcome this obstacle, the crucial and simple observation is that at most k subtrees rooted at a vertex of T contain vertices used by the desired rotation sequence σ (cf. Lemma 13). This implies that if there are at least $k+1$ T-siblings of the same type, necessarily the whole subtree of at least one of them, say u, will not be used by σ, implying that u (and its whole subtree) achieves the desired parent in T' without being used by σ, and the same occurs to any other T-sibling of the same type. Thus, keeping track of the existence of at least $k+1$ such children (regardless of their actual number) is enough to capture this "static" behavior, and allows us to shrink the possible distinct numbers to keep track of to a function of k (cf. Equation 2, where the "min" is justified by the previous discussion). Finally, for technical reasons we also incorporate into the type of a vertex its desired parent in T', in case it defers from its parent in T' (cf. function $\mathsf{want\text{-}parent}(T, T', \cdot)$). See Definition 9 for the formal definition of type and Figure 4 for an example with $k = 2$.

We prove (cf. Lemma 10) that the number of types is indeed bounded by a (large) function $g(k)$ depending only on k, and this function is what yields the upper bound on the asymptotic running time of the FPT algorithm of Theorem 1. Moreover, we show (cf. Observation 11) that the type of a vertex can be computed in time $g(k) \cdot |V(G)|$. We then use the notion of type and the bound given by Lemma 10 to define the desired set $M_Z \subseteq Z$ of size bounded by a function of k. In order to find M_Z, we apply a marking algorithm on Z, that first identifies a set $M_Z^{\mathsf{pre}} \subseteq V(Z)$ of *pre-marked* vertices, whose size is not necessarily bounded by a function of k, and then "prunes" this set M_Z^{pre} in a root-to-leaf fashion to find the desired set of *marked* vertices $M_Z \subseteq M_Z^{\mathsf{pre}}$ of appropriate size. See Figure 5 for an example of the marking algorithm for $k = 1$. We define $M = \cup_{Z \in \mathsf{cc}(T[B_{\mathsf{cb}}])} M_Z$ (where cc denotes the set of connected components), and we call it the set of *marked vertices* of T. We prove (cf. Lemma 12) that the size of M is roughly equal to the number of types, and that the set M can be computed in time FPT.

Once we have our set of marked vertices M at hand, it remains to prove that we can restrict the rotations to use only vertices in M. This is proved in our main technical result (cf. Lemma 14), whose proof critically exploits the recursive definition of the types. In a nutshell, we consider an ℓ-rotation sequence σ from T to T', for some $\ell \leq k$, minimizing, among all ℓ-rotation sequences from T to T', the number of used vertices in $V(T) \setminus M$. Our goal is to define another ℓ-rotation sequence σ' from T to T' using strictly less vertices in $V(T) \setminus M$ than σ, contradicting the choice of σ. To this end, let $v \in V(T) \setminus M$ be a furthest (with respect to the distance to $\mathsf{root}(T)$) non-marked vertex of T that is used by σ. We distinguish two cases.

In Case 1, we assume that v has a marked T-sibling v' with $\tau(T, Z, v) = \tau(T, Z, v')$ (cf. **??**). It is not difficult to prove that we can define σ' from σ by just replacing v with v' in all the rotations of σ involving v (cf. **??** and **??**).

In Case 2, all T-siblings v' of v with $\tau(T, Z, v) = \tau(T, Z, v')$, if any, are non-marked. In this case, in order to define another ℓ-rotation sequence σ' from T to T' that uses more marked vertices than σ, we need to modify σ in a more *global* way than in Case 1. Namely, in order to define σ', we need a more global (and involved) replacement, which we achieve via what we call a *representative function* ρ. To define ρ, we first guarantee the existence of a very helpful vertex $v^\star$ that is a non-marked ancestor of v having a marked T-sibling v' of the same type such that no vertex in $T(v')$ is used by σ; see **??** and **??**. Exploiting the recursive definition of type, we then define our representative function ρ, mapping vertices used by σ in $T(v^\star)$ to vertices in $T(v')$ of the same type (cf. **??**), and prove that we can define σ' from σ by replacing each vertex v used by σ in $T(v^\star)$ by its image via ρ in $T(v')$ in all the rotations of σ involving v (cf. **??** and **??**).

3 Preliminaries

Graphs. We use standard graph-theoretic notation, and we refer the reader to [18] for any undefined terms. An edge between two vertices u, v of a graph G is denoted by uv. For a graph G and a vertex set $S \subseteq V(G)$, the graph $G[S]$ has vertex set S and edge set $\{uv \mid u, v \in S \text{ and } uv \in E(G)\}$. A *connected component* Z of a graph G is a connected subgraph that is maximal (with respect to the addition of vertices or edges) with this property. We let $\mathsf{cc}(G)$ denote the set of connected components of a graph G. The *distance* between two vertices x, y in G, denoted by $\mathsf{dist}_G(x, y)$, is the length of a shortest path between x and y in G. The *diameter* of G, denoted by $\mathsf{diam}(G)$, is the maximum length of a shortest path between any two vertices of G. We will often consider distances and the diameter of some rooted tree T that is (a subtree of) an elimination tree of a graph G. We stress that $\mathsf{dist}_T(x, y)$ refers to the distance between x and y in T, not in G, and the same applies to $\mathsf{diam}(T)$.

For a graph G, a vertex $v \in V(G)$, and an integer $r \geq 1$, we denote by $N^r_G[v]$ the set of vertices within distance at most r from v in G, including v itself. For a set $S \subseteq V(G)$, we let $N^r_G[S] = \bigcup_{v \in S} N^r_G[v]$. For a subgraph H of G, we use $N^r_G(H)$ as a shortcut for $N^r_G(V(H))$. In all these notations, we omit the superscript r in the case where $r = 1$, that is, to refer to the usual neighborhood.

For a positive integer p, we let $[p]$ denote the set $\{1, 2, \ldots, p\}$. If $f : A \to B$ is a function between two sets A and B and $A' \subseteq A$, we denote by $f|_{A'}$ the restriction of f to A'.

Rooted trees. For a rooted tree T, we use $\mathsf{root}(T)$ to denote its root. For a vertex $v \in V(T)$, we denote by $\mathsf{parent}(T, v)$ the unique parent of v in T (or the empty set if v is the root), by $\mathsf{children}(T, v)$ the set of children of v in T, by $\mathsf{ancestors}(T, v)$ the set of ancestors of v in T (including v itself), and by $\mathsf{desc}(T, v)$ the set of descendants of v in T (including v itself). The *strict* ancestors (resp. descendants) of v are the vertices in the set $\mathsf{ancestors}(T, v) \setminus \{v\}$ (resp. $\mathsf{desc}(T, v) \setminus \{v\}$). We denote by $T(v)$ the subtree of T rooted at v. Two vertices $v, v' \in V(T)$ are T-*siblings* if $\mathsf{parent}(T, v) = \mathsf{parent}(T, v')$.

Rotation of an edge in an elimination tree. We provide the formal definition of the rotation operation, which has been already informally defined in the introduction (cf. Figure 2).

▶ **Definition 2** (rotation operation). *Let T be an elimination tree of a graph G and let $uv \in E(T)$ with $\mathsf{parent}(T, v) = u$. The* rotation of uv *in T creates another elimination tree $\mathsf{rot}(T, uv)$ of G defined as follows, where for better readability we use $T' = \mathsf{rot}(T, uv)$:*
1. $\mathsf{parent}(T', u) = v$.
2. $u \in \mathsf{children}(T', v)$.
3. *If $u \neq \mathsf{root}(T)$, let $z = \mathsf{parent}(T, u)$. Then $\mathsf{children}(T', z) = (\mathsf{children}(T, z) \setminus \{u\}) \cup \{v\}$.*
4. $\mathsf{children}(T, u) \subseteq \mathsf{children}(T', u)$.
5. *Let $w \in \mathsf{children}(T, v)$. If u is adjacent in G to some vertex in $T(w)$, then $w \in \mathsf{children}(T', u)$; otherwise $w \in \mathsf{children}(T', v)$.*
6. *For every vertex $s \in V(G) \setminus \{u, v, z\}$, $\mathsf{children}(T', s) = \mathsf{children}(T, s)$.*

A k-*rotation sequence* from an elimination tree T to another elimination tree T' (of the same graph G) is an ordered set $(e_1, \ldots, e_k)$ of edges such that, letting inductively $T_0 := T$ and, for $i \in [k]$, $T_i := \mathsf{rot}(T_{i-1}, e_i)$ with $e_i \in E(T_{i-1})$, we have that $T_k = T'$. In other words, a k-rotation sequence consists of the ordered list of the k edges to be rotated in order to obtain T' from T, going through the intermediate elimination trees $T_1, \ldots, T_{k-1}$ (of the same graph G). Clearly, $\mathsf{dist}(T, T') \leq k$ if and only if there exists an ℓ-rotation sequence from T to T' for some $\ell \leq k$. We say that a vertex $v \in V(T)$ is *used* by a rotation sequence σ if it is an endpoint of some of the edges that are rotated by σ.

Parameterized complexity. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, for some finite alphabet Σ. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, the value k is called the *parameter*. Such a problem is *fixed-parameter tractable* (FPT for short) if there is an algorithm that decides membership of an instance (x, k) in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f. Consult [16, 19–21, 33] for background on parameterized complexity.

4 Formal description of the FPT algorithm

In this section we present our FPT algorithm to solve the Rotation Distance problem. We start in Subsection 4.1 by providing some definitions and useful observations about the so-called *good* and *bad* vertices. In Subsection 4.2 we show that we can assume that all the rotations involve vertices within balls of small radius around bad vertices. In Subsection 4.3 we describe our marking algorithm, using the definition of type, and show that the set of marked vertices can be computed in FPT time. In Subsection 4.4 we prove our main technical result (Lemma 14), stating that we can restrict the desired rotations to involve only marked vertices. Finally, in Subsection 4.5 we wrap up the previous results to prove Theorem 1.

4.1 Good and bad vertices

Throughout the paper, we assume that all the considered elimination trees are of a same fixed graph G. For simplicity, we may assume henceforth that the considered input graph G is connected.

Our algorithm exploits how a rotation in an elimination tree T may affect the parents and the children of its vertices. Note that a single rotation of an edge $uv \in E(T)$, yielding an elimination tree T', may change the parent of arbitrarily many vertices. Indeed, these vertices are the roots of the red subtrees in Figure 2, and the considered vertex v may be adjacent to the root of arbitrarily many subtrees containing at least one vertex adjacent to u: for each such root r, $\mathsf{parent}(T, r) = v$ but $\mathsf{parent}(T', r) = u$. As a concrete example, in Figure 1, $\mathsf{parent}(T, s) = v$ but $\mathsf{parent}(T', s) = u$. On the other hand, item 6 of Definition 2 implies that there are at most three vertices whose children set changes from T to T', namely u, v, z as depicted in Figure 2. (Note that the sets of children of u and v always change, and that of z changes provided that this vertex exists.) We state this observation formally, since it will be extensively used afterwards.

▶ **Observation 3.** *One rotation may change the set of children of at most three vertices.*

The above discussion motivates the following definition.

▶ **Definition 4** (bad vertices). *Given two elimination trees T and T', a vertex $v \in V(T)$ is* (T, T')-children-bad *(resp. (T, T')-*parent-bad*) if* $\mathsf{children}(T, v) \neq \mathsf{children}(T', v)$ *(resp.* $\mathsf{parent}(T, v) \neq \mathsf{parent}(T', v)$*). A vertex $v \in V(T)$ is* (T, T')-bad *if it is (T, T')-children-bad, or (T, T')-parent-bad, or both. A vertex $v \in V(T)$ is* (T, T')-good *if it is not (T, T')-bad.*

Note that T contains no (T, T')-children-bad (or (T, T')-parent-bad, or just (T, T')-bad) vertices if and only if $T = T'$, that is, if and only if $\mathsf{dist}(T, T') = 0$. Also, note that a vertex $v \in V(T)$ is (T, T')-children-bad, with $\mathsf{children}(T, v) \neq \emptyset$, if and only if at least one of its children is (T, T')-parent-bad. Observation 3 directly implies the following necessary condition for the existence of a solution.

▶ **Observation 5.** *Given two elimination trees T and T', if $\mathsf{dist}(T, T') \leq k$ then the number of (T, T')-children-bad vertices is at most $3k$.*

Observation 5 is equivalent to saying that we can safely conclude that any instance (G, T, T', k) of Rotation Distance with at least $3k + 1$ (T, T')-children-bad vertices is a no-instance. Thus, we can assume henceforth that we are dealing with an instance of Rotation Distance containing at most $3k$ (T, T')-children-bad vertices.

4.2 Restricting the rotations to small balls around bad vertices

Our next goal is to prove (Lemma 7) that we can assume that the desired sequence of at most k rotations to transform T into T' uses only edges whose both endvertices lie in the union of all the balls of appropriate radius (depending only on k) around (T, T')-children-bad vertices of T, whose number is bounded by a function of k by Observation 5.

In the next definition, for the sake of notational simplicity we omit T, T', and k from the notation B_{cb}, as we assume that they are already given, and fixed, as the input of our problem. We include $\mathsf{root}(T)$ in the considered set for technical reasons, namely in the proof of **??**.

▶ **Definition 6** (union of balls of children-bad vertices)**.** *Let $C \subseteq V(T)$ be the set of (T, T')-children-bad vertices. We define $B_{\mathsf{cb}} = N_T^{2k}[C \cup \mathsf{root}(T)]$.*

▶ **Lemma 7.** *If $\mathsf{dist}(T, T') \leq k$, then there exists an ℓ-rotation sequence from T to T', with $\ell \leq k$, using only vertices in B_{cb}.*

Proof. TOPROVE 0 ◀

By Lemma 7, we focus henceforth on trying to find an ℓ-rotation sequence from T to T', with $\ell \leq k$, consisting only of edges with both endvertices in B_{cb}. First, we will consider each of the at most $3k + 1$ connected components of $T[B_{\mathsf{cb}}]$ separately. In fact, we can get a better bound, as if $T[B_{\mathsf{cb}}]$ has at least $k + 1$ connected components, we can immediately conclude that we are dealing with a no-instance, since at least one rotation is needed in each component. Thus, we may assume that $T[B_{\mathsf{cb}}]$ has at most k connected components. On the other hand, since $T[B_{\mathsf{cb}}]$ is defined as the union of at most $3k + 1$ balls of radius $2k$, it follows that every $Z \in \mathsf{cc}(T[B_{\mathsf{cb}}])$ satisfies

$$\mathsf{diam}(Z) \leq (3k + 1)4k. \tag{1}$$

Thus, by Equation 1, the "only" obstacle to obtain the desired FPT algorithm is that the vertices in B_{cb} can have an arbitrarily large degree. Note that in the particular case where the underlying graph G has bounded degree, for instance if G is a path [14, 25, 26, 28], the maximum degree of any elimination tree of G is bounded, and therefore in that case $|B_{\mathsf{cb}}|$ is bounded by a function of k, and an FPT algorithm follows immediately. To the best of our knowledge, this result was not known for graphs other than paths.

4.3 Description of the marking algorithm

As discussed in Section 2, our strategy to deal with high-degree vertices in B_{cb} is as follows. For each connected component $Z \in \mathsf{cc}(T[B_{\mathsf{cb}}])$, our goal is to identify a subset $M_Z \subseteq V(Z)$

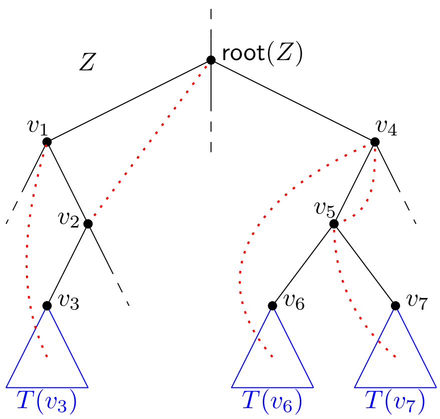
of size bounded by a function of k, such that we can restrict our search to rotations involving only pairs of vertices in M_Z. Clearly, this would yield the desired FPT algorithm. To find such a "small" set $M_Z \subseteq V(Z)$, we define the notion of *type* of a vertex $v \in V(Z)$, in such a way that the number of different types is bounded by a function of k. Then, we will prove that it is enough to keep in M_Z, for each type, a number of vertices bounded again by a function of k.

Let henceforth Z be a connected component of $T[B_{\mathsf{cb}}]$, which we consider as a rooted tree with its own set of leaves, which are not necessarily leaves in T. We define $\mathsf{root}(Z)$ to be the vertex in $V(Z)$ closest to $\mathsf{root}(T)$ in T.

Before defining the types, we need to define the *trace* of a vertex v in a designated vertex set $Z \subseteq V(T)$ that will correspond to a connected component of B_{cb}. Roughly speaking, the trace of a vertex v captures "abstractly" the neighborhood of a (whole) subtree rooted at v among (the ordered set of) its ancestors within the designated vertex set $Z \subseteq V(T)$. We stress that, when considering the neighborhood in the set of ancestors, we look at the whole subtree $T(v)$ rooted at v, and not only at its restriction to the set Z.

▶ **Definition 8** (trace of a vertex in a component Z). *Let T be an elimination tree (of a graph G), let Z be a rooted subtree of T corresponding to a connected component of B_{cb}, and let $v \in V(Z)$. The* trace *of v in Z, denoted by $\mathsf{trace}(T, Z, v)$, is a binary vector of dimension $\mathsf{dist}_T(v, \mathsf{root}(Z))$ defined as follows (note that if $v = \mathsf{root}(Z)$, then its trace is empty). For $i \in [\mathsf{dist}_T(v, \mathsf{root}(Z))]$, let $u_i \in \mathsf{ancestors}(T, v)$ be the ancestor of v in T such that $\mathsf{dist}_T(v, u_i) = i$. Then the i-th coordinate of $\mathsf{trace}(T, Z, v)$ is 1 if $wu_i \in E(G)$ for some vertex $w \in V(T(v))$, and 0 otherwise.*

See Figure 3 for an example of the trace of some vertices in a component Z.



$$\begin{aligned}
\mathsf{trace}(T, Z, v_1) &= (1)\\
\mathsf{trace}(T, Z, v_2) &= (1, 1)\\
\mathsf{trace}(T, Z, v_3) &= (0, 1, 0)\\
\mathsf{trace}(T, Z, v_4) &= (0)\\
\mathsf{trace}(T, Z, v_5) &= (1, 0)\\
\mathsf{trace}(T, Z, v_6) &= (0, 1, 0)\\
\mathsf{trace}(T, Z, v_7) &= (1, 0, 0)
\end{aligned}$$

Figure 3 A component Z of $T[B_{\mathsf{cb}}]$ and the trace of some of its vertices $v_1, \ldots, v_7$. Red dotted edges represent adjacencies in G. Note the $\mathsf{trace}(T, Z, v_3) = \mathsf{trace}(T, Z, v_6)$, even if v_3 and v_6 are not siblings in Z.

For a vertex $v \in V(T)$, let $\mathsf{want\text{-}parent}(T, T', v)$ be equal to $\emptyset$ if $\mathsf{parent}(T, v) = \mathsf{parent}(T', v)$, and to $\mathsf{parent}(T', v)$ otherwise. Note that, by Observation 5, the function $\mathsf{want\text{-}parent}(T, T', v)$ can take up to $3k + 1$ distinct values when ranging over all $v \in V(T)$.

▶ **Definition 9** (type of a vertex in a component Z). *Let T be an elimination tree (of a graph G), let Z be a rooted subtree of T corresponding to a connected component of B_{cb}, and let $v \in V(Z)$. The* type *of vertex v, denoted by $\tau(T, Z, v)$, is recursively defined as follows, where $\mathsf{type\text{-}children}(T, Z, v) := \{\tau(T, Z, u) \mid u \in \mathsf{children}(Z, v)\}$ is the set of types occurring in the children of v:*

- *If v is a leaf of Z, then $\tau(T, Z, v)$ consists of the pair* $(\mathsf{want\text{-}parent}(T, T', v), \mathsf{trace}(T, Z, v))$.
- *Otherwise, $\tau(T, Z, v)$ consists of a tuple* $(\mathsf{want\text{-}parent}(T, T', v), \mathsf{trace}(T, Z, v), f_v)$*, where* $f_v : \mathsf{type\text{-}children}(T, Z, v) \to [k + 1]$ *is a mapping defined such that, for every* $\tau \in \mathsf{type\text{-}children}(T, Z, v)$,

$$f_v(\tau) = \min\{k + 1 \ , \ |\{u \in \mathsf{children}(Z, v) \mid \tau(T, Z, u) = \tau\}|\}. \tag{2}$$

See Figure 4 for an example for $k = 2$ of how the types are computed in a component Z.

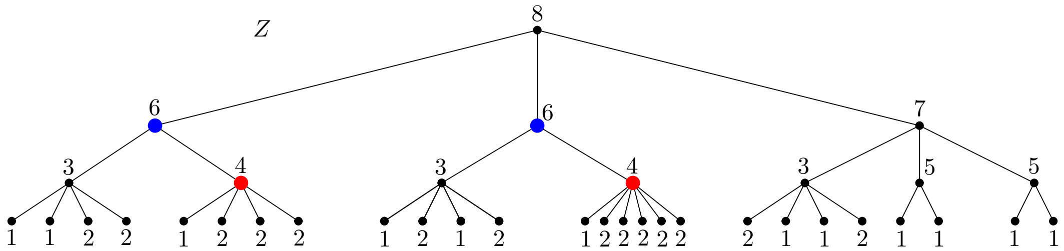


Figure 4 A component Z of $T[B_{\mathsf{cb}}]$ and the types of its vertices, for an instance with $k = 2$. For the sake of simplicity, different types are depicted with different numbers. Assume that the leaves have only two possible types, namely 1 and 2, and that all non-leaf vertices at the same distance from the root have the same trace and the same function $\mathsf{want\text{-}parent}(T, T', \cdot)$. Note that the red vertices have the same type (namely, 4) because they both have one child of type 1 and at least $k+1 = 3$ children of type 2. Note also that the blue vertices have the same type (namely, 6) because they both have one child of type 3 and one child of type 4.

▶ **Lemma 10.** *The set $\{\tau(T, Z, v) \mid v \in V(Z)\}$ has size bounded by a function $g(k)$, with*

$$g(k) = k^{2^{2^{\cdot^{\cdot^{2^{\mathcal{O}(k^2)}}}}}}, \text{ where the tower of exponentials has height } \mathsf{diam}(Z) = \mathcal{O}(k^2). \tag{3}$$

Proof. TOPROVE 1 ◀

Note that, in order to compute the type of a vertex in a component Z, the recursive definition of the types together with Lemma 10 easily imply the following observation, where the term $|V(G)|$ comes from checking the neighborhood of $T(v)$ within the set $\mathsf{ancestors}(T, v)$ in the computation of the trace (cf. Definition 8).

▶ **Observation 11.** *Let T be an elimination tree of a graph G, let Z be a rooted subtree of T corresponding to a connected component of B_{cb}, and let $v \in V(Z)$. Then $\tau(T, Z, v)$ can be computed in time $g(k) \cdot |V(G)|$, where $g(k)$ is the function from Lemma 10.*

We will now use the notion of type and the bound given by Lemma 10 to define the desired set $M_Z \subseteq Z$ of size bounded by a function of k. In order to find M_Z, we apply a marking algorithm on Z, that first identifies a set $M_Z^{\mathsf{pre}} \subseteq V(Z)$ of *pre-marked* vertices, whose size is not necessarily bounded by a function of k, and then "prunes" this set M_Z^{pre} in a root-to-leaf fashion to find the desired set of *marked* vertices $M_Z \subseteq M_Z^{\mathsf{pre}}$ of appropriate size.

Start with $M_Z^{\mathsf{pre}} = \emptyset$. For every vertex $v \in V(Z)$ and every $\tau \in \mathsf{type\text{-}children}(Z, v)$, do the following:

- If $|\{u \in \mathsf{children}(Z, v) \mid \tau(Z, u) = \tau\}| \leq k + 1$, add the whole set $\{u \in \mathsf{children}(Z, v) \mid \tau(Z, u) = \tau\}$ to M_Z^{pre}.
- Otherwise, add to M_Z^{pre} an arbitrarily chosen subset of $\{u \in \mathsf{children}(Z, v) \mid \tau(Z, u) = \tau\}$ of size $k + 1$.

Finally, add $\mathsf{root}(Z)$ to M_Z^{pre}. We define $M_{\mathsf{pre}} = \cup_{Z \in \mathsf{cc}(T[B_{\mathsf{cb}}])} M_Z^{\mathsf{pre}}$ and we call it the set of *pre-marked vertices* of T.

We are now ready to define our bounded-size set $M_Z \subseteq M_Z^{\mathsf{pre}}$. Start with $M_Z = \{\mathsf{root}(Z)\}$ and for $i = 0, \ldots, \mathsf{diam}(Z) - 1$, proceed inductively as follows: if $v \in V(Z)$ is a vertex with $\mathsf{dist}_Z(v, \mathsf{root}(Z)) = i$ that already belongs to M_Z, add to M_Z the set $\mathsf{children}(Z, v) \cap M_Z^{\mathsf{pre}}$. Finally, for every (T, T')-children-bad vertex v of T that belongs to Z, we add to M_Z the set $\mathsf{ancestors}(Z, v)$. This concludes the construction of M_Z. Note that if a vertex $v \in V(Z)$ belongs to M_Z, then the whole set $\mathsf{ancestors}(Z, v)$ belongs to M_Z as well. We define $M = \cup_{Z \in \mathsf{cc}(T[B_{\mathsf{cb}}])} M_Z$, and we call it the set of *marked vertices* of T. See Figure 5 for an example of the marking algorithm.

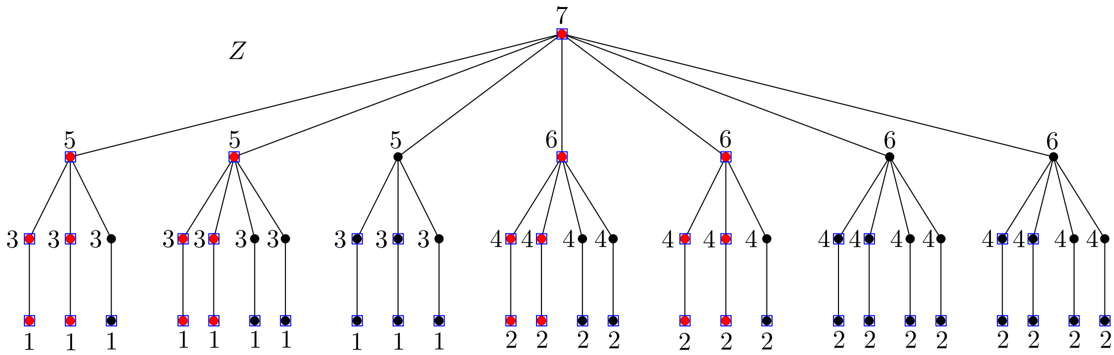


Figure 5 Example of the marking algorithm applied to a component Z of $T[B_{\mathsf{cb}}]$, for an instance with $k = 1$. As in Figure 4, different types are depicted with different numbers. Vertices inside blue squares belong to M_Z^{pre}, and red vertices belong to M_Z.

▶ **Lemma 12.** *The set $M \subseteq V(T)$ of marked vertices has size bounded by a function $h(k)$, where $h(k)$ has the same asymptotic growth as the function $g(k)$ given by Lemma 10. Moreover, M can be computed in time $h(k) \cdot |V(G)|$.*

Proof. TOPROVE 2 ◀

4.4 Restricting the rotations to marked vertices

In this subsection we prove our main technical result (Lemma 14), which immediately yields the desired FPT algorithm combined with Lemma 12 (whose proof uses Lemma 7), as discussed in Subsection 4.5. We first need an easy lemma that will be extensively used in the proof of Lemma 14.

▶ **Lemma 13.** *Let σ be an ℓ-rotation sequence from T to T', for some $\ell \leq k$. For every vertex $v \in V(T)$, there are at most k vertices $u_1, \ldots, u_k \in \mathsf{children}(T, v)$ such that σ uses a vertex in each of the rooted subtrees $T(u_1), \ldots, T(u_k)$.*

Proof. TOPROVE 3 ◀

Note that, if in the statement of Lemma 13 we replaced "at most k vertices" with "at most $2k$ vertices", then its proof would be trivial, as any of the at most k rotations of σ involves two vertices, so at most $2k$ distinct vertices overall. In that case, for the proof of Lemma 14 to go through, we would have to replace, in Equation 2 in the definition of type, "$k + 1$" with "$2k + 1$" when taking the minimum. In the sequel we will often use a weaker version of Lemma 13, namely that for every vertex $v \in V(T)$, at most k vertices in $\mathsf{children}(T, v)$ are used by an ℓ-rotation sequence from T to T'.

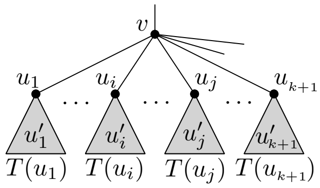


Figure 6 Illustration of the proof of Lemma 13.

We are now ready to prove our main lemma.

▶ **Lemma 14.** *If* $\mathsf{dist}(T, T') \leq k$*, then there exists an* ℓ*-rotation sequence from* T *to* T'*, with* $\ell \leq k$*, using only vertices in* M*.*

Proof. TOPROVE 4 ◀

4.5 Wrapping up the algorithm

We finally have all the ingredients to prove our main result, which we restate for convenience.

▶ **Theorem 1.** *The* Rotation Distance *problem can be solved in time* $f(k) \cdot |V(G)|$*, with* $f(k) = k^{k \cdot 2^{2^{\cdot^{\cdot^{\cdot^{2^{\mathcal{O}(k^2)}}}}}}}$*, where the tower of exponentials has height at most* $(3k+1)4k = \mathcal{O}(k^2)$*.*

Proof. TOPROVE 5 ◀

5 Further research

We proved that the Rotation Distance problem, for a general graph G, can be solved in time $f(k) \cdot |V(G)|$, where $f(k)$ is the function given by Theorem 1. This function is quite large, and it is worth trying to improve it. The growth of $f(k)$ is mainly driven by the number of different types of vertices (cf. Definition 9) that we consider in our marking algorithm. We need this recursive definition of type to guarantee that, when two vertices v, v' have the same type, then for each possible type τ and every integer d at most the bound given in Equation 1, vertices v and v' have the same number (up to $k+1$) of descendants of type τ within distance d. This is exploited, for instance, in Case 2 of the proof of Lemma 14 to apply a recursive argument. It may possible to find a simpler argument in the replacement operation performed in the proof of Lemma 14 (using the representative function ρ), and in that case, one may allow for a less refined notion of type, leading to a better bound.

Another natural direction is to investigate whether Rotation Distance admits a polynomial kernel parameterized by k. So far, this is only known when the considered graph G is a path, where even linear kernels are known [14, 30]; see Table 1. As an intermediate step, one may consider graphs of bounded degree, for which it seems plausible that Lemma 7 (restriction to few balls of bounded diameter) provides a helpful opening step.

Finally, Ito et al. [24] also proved the NP-hardness of a related problem called Combinatorial Shortest Path on Polymatroids, relying on the fact that graph associahedra can be realized as the base polytopes of some polymatroids [35]. To the best of our knowledge, the parameterized complexity of this problem has not been investigated.

References

1 Georgy M. Adel'son-Vel'skii and Evgenii M. Landis. An algorithm for the organization of information. *Soviet Mathematics Doklady*, 3:1259–1263, 1962. URL: https://zhjwpku.com/assets/pdf/AED2-10-avl-paper.pdf.

Rotation Distance	paths	general graphs
NP-hard	open	✓ [24]
FPT	✓ [14, 25, 26, 28, 30]	✓ [Theorem 1]
Polynomial kernel	✓ [14, 30]	open

Table 1 Known results and open problems about the (parameterized) complexity of the Rotation Distance problem, both on paths and general graphs.

2 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip Distance Between Triangulations of a Simple Polygon is NP-Complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015. doi:10.1007/S00454-015-9709-7.

3 Benjamin Aram Berendsohn. The diameter of caterpillar associahedra. In *Proc. of the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 227 of *LIPIcs*, pages 14:1–14:12, 2022. doi:10.4230/LIPICS.SWAT.2022.14.

4 Nicolas Bousquet, Amer E. Mouawad, Naomi Nishimura, and Sebastian Siebertz. A survey on the parameterized complexity of reconfiguration problems. *Computer Science Review*, 53:100663, 2024. doi:10.1016/J.COSREV.2024.100663.

5 Jean Cardinal, Linda Kleist, Boris Klemz, Anna Lubiw, Torsten Mütze, Alexander Neuhaus, and Lionel Pournin. Working grup 4.2: Rotation distance between elimination trees. *Report from Dagstuhl Seminar 22062: Computation and Reconfiguration in Low-Dimensional Topological Spaces*, page 35. URL: https://drops.dagstuhl.de/storage/04dagstuhl-reports/volume12/issue02/22062/DagRep.12.2.17/DagRep.12.2.17.pdf.

6 Jean Cardinal, Stefan Langerman, and Pablo Pérez-Lantero. On the diameter of tree associahedra. *Electronic Journal of Combinatorics*, 25(4):4, 2018. doi:10.37236/7762.

7 Jean Cardinal, Arturo Merino, and Torsten Mütze. Efficient generation of elimination trees and graph associahedra. In *Proc. of the 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2128–2140, 2022. doi:10.1137/1.9781611977073.84.

8 Jean Cardinal, Lionel Pournin, and Mario Valencia-Pabon. Diameter estimates for graph associahedra. *Annals of Combinatorics*, 26:873–902, 2022. doi:10.1007/s00026-022-00598-z.

9 Jean Cardinal, Lionel Pournin, and Mario Valencia-Pabon. The rotation distance of brooms. *European Journal of Combinatorics*, 118:103877, 2024. doi:10.1016/J.EJC.2023.103877.

10 Jean Cardinal and Raphael Steiner. Shortest paths on polymatroids and hypergraphic polytopes. *CoRR*, abs/2311.00779, 2023. arXiv:2311.00779.

11 Michael Carr and Satyan L. Devadoss. Coxeter complexes and graph-associahedra. *Topology and its Applications*, 153(12):2155–2168, 2006. doi:10.1016/j.topol.2005.08.010.

12 Cesar Ceballos, Francisco Santos, and Günter M. Ziegler. Many non-equivalent realizations of the associahedron. *Combinatorica*, 35(5):513–551, 2015. doi:10.1007/s00493-014-2959-9.

13 Sean Cleary. Restricted rotation distance between k-ary trees. *Journal of Graph Algorithms and Applications*, 27(1):19–33, 2023. doi:10.7155/JGAA.00611.

14 Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Information Processing Letters*, 109(16):918–922, 2009. doi:10.1016/J.IPL.2009.04.023.

15 Sean Cleary and Katherine St. John. A linear-time approximation algorithm for rotation distance. *Journal of Graph Algorithms and Applications*, 14(2):385–390, 2010. doi:10.7155/JGAA.00212.

16 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Satyan L. Devadoss. A realization of graph associahedra. *Discrete Mathematics*, 309(1):271–276, 2009. doi:10.1016/J.DISC.2007.12.092.

18 Reinhard Diestel. *Graph Theory*. Springer-Verlag, Heidelberg, 5th edition, 2016. URL: https://link.springer.com/book/10.1007/978-3-662-53622-3.

19 Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

20 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006. doi:10.1007/3-540-29953-X.

21 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

22 Leonidas J. Guibas and Robert Sedgewick. A dichromatic framework for balanced trees. In *Proc. of the 19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 8–21, 1978. doi:10.1109/SFCS.1978.3.

23 Christophe Hohlweg and Carsten E. M. C. Lange. Realizations of the associahedron and cyclohedron. *Discrete & Computational Geometry*, 37(4):517–543, 2007. doi:10.1007/S00454-007-1319-6.

24 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, Shun-ichi Maezawa, Yuta Nozaki, and Yoshio Okamoto. Hardness of Finding Combinatorial Shortest Paths on Graph Associahedra. In *Prof. of the 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *LIPIcs*, pages 82:1–82:17, 2023. doi:10.4230/LIPIcs.ICALP.2023.82.

25 Iyad Kanj, Eric Sedgwick, and Ge Xia. Computing the flip distance between triangulations. *Discrete & Computational Geometry*, 58(2):313–344, 2017. doi:10.1007/S00454-017-9867-X.

26 Haohong Li and Ge Xia. An $O(3.82^k)$ time FPT algorithm for convex flip distance. In *Proc. of the 40th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 254 of *LIPIcs*, pages 44:1–44:14, 2023. doi:10.4230/LIPICS.STACS.2023.44.

27 Jean-Louis Loday. Realization of the Stasheff polytope. *Archiv der Mathematik*, 83:267–278, 2004. doi:10.1007/s00013-004-1026-y.

28 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. doi:10.1016/J.COMGEO.2014.11.001.

29 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. doi:10.1016/J.COMGEO.2014.11.001.

30 Joan M. Lucas. An improved kernel size for rotation distance in binary trees. *Information Processing Letters*, 110(12-13):481–484, 2010. doi:10.1016/J.IPL.2010.04.022.

31 Thibault Manneville and Vincent Pilaud. Graph properties of graph associahedra. *CoRR*, abs/1409.8114, 2010. arXiv:1409.8114.

32 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

33 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/acprof:oso/9780198566076.001.0001.

34 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/J.COMGEO.2014.01.001.

35 Alex Postnikov, Victor Reiner, and Lauren Williams. Faces of generalized permutohedra. *Documenta Mathematica*, 13:207–273, 2008. doi:10.4171/DM/248.

36 Alexander Postnikov. Permutohedra, associahedra, and beyond. *International Mathematics Research Notices*, 2009(6):1026–1106, 2009. doi:10.1093/imrn/rnn153.

37 Lionel Pournin. The diameter of associahedra. *Advances in Mathematics*, 259:13–42, 2014. doi:10.1016/j.aim.2014.02.035.

38 Lionel Pournin. The asymptotic diameter of cyclohedra. *Israel Journal of Mathematics*, 219(2):609–635, 2017. doi:10.1007/s11856-017-1492-0.

39 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988. doi:10.1090/S0894-0347-1988-0928904-4.

40 Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, 1985. doi:10.1145/3828.3835.

41 Richard P. Stanley. *Catalan numbers*. Cambridge University Press, 2015. doi:10.1017/CBO9781139871495.