Flip Distance Is in FPT Time $\mathcal{O}(n + k \cdot c^k)$

Iyad Kanj[1] **and Ge Xia**[2]

1 **School of Computing, DePaul University**
Chicago, USA
ikanj@cs.depaul.edu
2 **Department of Computer Science**
Lafayette College
Easton, USA, xiag@lafayette.edu

Abstract

Let $\mathcal{T}$ be a triangulation of a set $\mathcal{P}$ of n points in the plane, and let e be an edge shared by two triangles in $\mathcal{T}$ such that the quadrilateral Q formed by these two triangles is convex. A *flip* of e is the operation of replacing e by the other diagonal of Q to obtain a new triangulation of $\mathcal{P}$ from $\mathcal{T}$. The *flip distance* between two triangulations of $\mathcal{P}$ is the minimum number of flips needed to transform one triangulation into the other. The Flip Distance problem asks if the flip distance between two given triangulations of $\mathcal{P}$ is k, for some given $k \in \mathbb{N}$. It is a fundamental and a challenging problem.

In this paper we present an algorithm for the Flip Distance problem that runs in time $\mathcal{O}(n + k \cdot c^k)$, for a constant $c \leq 2 \cdot 14^{11}$, which implies that the problem is fixed-parameter tractable. The NP-hardness reduction for the Flip Distance problem given by Lubiw and Pathak can be used to show that, unless the exponential-time hypothesis (ETH) fails, the Flip Distance problem cannot be solved in time $\mathcal{O}^*(2^{o(k)})$. Therefore, one cannot expect an asymptotic improvement in the exponent of the running time of our algorithm.

1998 ACM Subject Classification F.2.2 Geometrical Problems and Computations, G.2.1 Combinatorial Algorithms

Keywords and phrases triangulations, flip distance, parameterized algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2015.500

1 Introduction

Let $\mathcal{P}$ be a set of n points in the plane. A *triangulation* of $\mathcal{P}$ is a partitioning of the convex hull of $\mathcal{P}$ into triangles such that the set of vertices of the triangles in the triangulation is $\mathcal{P}$. Note that the convex hull of $\mathcal{P}$ may contain points of $\mathcal{P}$ in its interior.

A *flip* to an (interior) edge e in a triangulation of $\mathcal{P}$ is the operation of replacing e by the other diagonal of the quadrilateral formed by the two triangles that share e, provided that this quadrilateral is convex; otherwise, flipping e is not permissible. The *flip distance* between two triangulations $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ of $\mathcal{P}$ is the length of a shortest sequence of flips that transforms $\mathcal{T}_{initial}$ into $\mathcal{T}_{final}$. This distance is always well-defined and is $O(|\mathcal{P}|^2)$ (*e.g.*, see [8]). The Flip Distance problem is: Given two triangulation $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ of $\mathcal{P}$, and $k \in \mathbb{N}$, decide if the flip distance between $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ is k.

Triangulations are a very important subject of study in computational geometry, and they have applications in computer graphics, visualization, and geometric design (see [17, 19, 20, 24], to name a few). Flips in triangulations and the Flip Distance problem have received a large share of attention (see [3] for a review). The Flip Distance problem is a very fundamental and challenging problem, and different aspects of this problem have

32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015).
Editors: Ernst W. Mayr and Nicolas Ollinger; pp. 500–512

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


been studied, including the combinatorial, geometrical, topological, and computational aspects [1, 2, 3, 4, 7, 8, 10, 18, 21, 22]. We can define the triangulations graph of $\mathcal{P}$, whose vertex-set is the set of all triangulations of $\mathcal{P}$, and in which two triangulations/vertices are adjacent if and only if their distance is 1. It is well-known that the triangulations graph has diameter $O(n^2)$ [8], and hence, we can transform any triangulation into another by a sequence of $O(n^2)$ flips. Moreover, it is known that the number of vertices in the triangulations graph is $\Omega(2.33^n)$ [1]. Therefore, solving the Flip Distance problem by finding a shortest path between the two triangulations in the triangulations graph is not feasible. A very similar problem to the Flip Distance was studied by Wagner [23] in 1936, who considered triangulated planar graphs instead.

The complexity of the Flip Distance problem was resolved very recently (2012) by Lubiw and Pathak [10, 11] who showed the problem to be NP-complete. Simultaneously, and independently, the problem was shown to be APX-hard by Pilz [18]. Very recently, Aichholzer et al. [2] showed the problem to be NP-complete for triangulations of a simple polygon. Resolving the complexity of the problem for the special case when $\mathcal{P}$ is in a convex position (*i.e.*, triangulations of a convex polygon) remains a longstanding open problem for at least 25 years (see [22]); this problem is equivalent to the problem of computing the rotation distance between two rooted binary trees [4, 22]. Cleary and St. John [4] showed that this special case (convex polygon) is fixed-parameter tractable (FPT): They gave a kernel of size $5k$ for the problem and presented an $\mathcal{O}^*((5k)^k)$-time FPT algorithm based on this kernel (the $\mathcal{O}^*()$ notation suppresses polynomial factors in the input size). The upper bound on the kernel size for the convex case was subsequently improved to $2k$ by Lucas [12], who also gave an $\mathcal{O}^*(k^k)$-time FPT algorithm for this case. The kernelization approaches used in [4, 12] for the convex case are not applicable to the general case. In particular, the reduction rules used in [4, 12] to obtain a kernel for the convex case, and hence the FPT algorithms based on these kernels, do not generalize to the problem under consideration in this paper.

In this paper we present an $\mathcal{O}(n + k \cdot c^k)$-time algorithm ($c \leq 2 \cdot 14^{11}$) for the Flip Distance problem for triangulations of an arbitrary point-set in the plane, which shows that the problem is FPT. Our result is a significant improvement over the $\mathcal{O}^*(k^k)$-time algorithm by Lucas [12] for the simpler convex case. The NP-hardness reduction by Lubiw and Pathak can be used to show that, unless the exponential-time hypothesis (ETH) fails [9], the Flip Distance problem cannot be solved in time $\mathcal{O}^*(2^{o(k)})$. Therefore, one should not expect an asymptotic improvement in the exponent of the running time of the presented algorithm. While it is not very difficult to show that the Flip Distance problem is FPT based on some of the structural results in this paper, obtaining an $\mathcal{O}^*(c^k)$-time algorithm, for some constant c, is quite involved, and requires a deep understanding of the structure of the problem.

Our approach is as follows. For any solution to a given instance of the problem, we can define a directed acyclic graph (DAG), whose nodes are the flips in the solution, that captures the dependency relation among the flips. We show that any topological sorting of this DAG corresponds to a valid solution of the instance. The difficult part is how, without knowing the DAG, to navigate the triangulation and perform the flips in an order that corresponds to a topological sorting of the DAG. We present a *very simple* nondeterministic algorithm that performs a sequence of "flip/move"-type local actions in a triangulation, where each local action has constant-many choices. The key is to show that there exists such a sequence of actions that corresponds to a topological sorting of the DAG associated with a solution to the instance, and that the length of this sequence is linear in the number of nodes in the DAG. This will us to simulate the nondeterministic algorithm by an $\mathcal{O}^*(c^k)$-time deterministic algorithm. To achieve the above goal, we develop structural results that reveal some of the structural intricacies of this fundamental and challenging problem.

Even though the triangulations considered in the paper are triangulations of a point-set in the plane, the presented algorithm works as well for triangulations of any polygonal region (even with points in its interior). Moreover, using a reduction in [11] from the Flip Distance of triangulations of a polygonal region with holes to the Flip Distance of triangulations of a polygonal region with points in its interior, the algorithm presented in this paper can solve the (more general) Flip Distance problem of triangulations of a polygonal region with (possible) holes within the same time upper bound.

2 Preliminaries

Let $\mathcal{P}$ be a set of n points in the plane, and let $\mathcal{T}$ be a triangulation of $\mathcal{P}$. Let e be an interior (non-boundary) edge in $\mathcal{T}$. The *quadrilateral associated with* e in $\mathcal{T}$ is defined to be the quadrilateral formed by the two adjacent triangles in $\mathcal{T}$ that share e as an edge. Let e be an edge in $\mathcal{T}$ such that the quadrilateral Q in $\mathcal{T}$ associated with e is convex. A *flip* f with underlying edge e is an operation performed to e in triangulation $\mathcal{T}$ that removes e and replaces it with the other diagonal of Q, thus obtaining a new triangulation of $\mathcal{P}$ from $\mathcal{T}$. We use the notation $\varepsilon(f)$ to denote the underlying edge e of a flip f in $\mathcal{T}$, and the notation $\varphi(f)$ to denote the new diagonal/edge resulting from flip f. Note that $\varphi(f)$ is not in $\mathcal{T}$. We say that a flip to an edge e is *admissible* in triangulation $\mathcal{T}$ if e is in $\mathcal{T}$ and the quadrilateral associated with e is convex. We say that two distinct edges e and e' in $\mathcal{T}$ *share a triangle* if e and e' appear in the same triangle in $\mathcal{T}$. We say that two distinct edges e and e' between points in $\mathcal{P}$ *cross* if e and e' intersect in their interior.

Let $\mathcal{T}$ be a triangulation. A sequence of flips $F = \langle f_1, \ldots, f_r \rangle$ is *valid* with respect to $\mathcal{T}$ if there exist triangulations $\mathcal{T}_0, \ldots, \mathcal{T}_r$ such that $\mathcal{T}_0 = \mathcal{T}$, f_i is admissible in $\mathcal{T}_{i-1}$, and performing flip f_i in $\mathcal{T}_{i-1}$ results in triangulation $\mathcal{T}_i$, for $i = 1, \ldots, r$. In this case we say that $\mathcal{T}_r$ is the *outcome* of applying F to $\mathcal{T}$ and that F *transforms* $\mathcal{T}$ into $\mathcal{T}_r$, and we write $\mathcal{T} \xrightarrow{F} \mathcal{T}_r$. The *length* of F, denoted $|F|$, is the number of flips in it. Many flips in a sequence F may have the same underlying edge, but all those flips are distinct flips. For two flips f_i and f_h of F such that $i < h$, a flip f_p in F is said to be *between* f_i and f_h if $i < p < h$.

For two triangulations $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ of $\mathcal{P}$, the *flip distance* between $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ is the smallest $d \in \mathbb{N}$ such that there is a sequence F of length d satisfying that $\mathcal{T}_{initial} \xrightarrow{F} \mathcal{T}_{final}$. The Flip Distance problem is defined as follows:

Flip Distance
Given: Two triangulation $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ of $\mathcal{P}$.
Parameter: k.
Question: Is the flip distance between $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$ equal to k?

Let G be a graph. $V(G)$ and $E(G)$ denote the vertex-set and the edge-set of G, respectively, and $|G|$ denotes the *size* of G, which is $|V(G)| + |E(G)|$. For a directed graph G, a *weakly connected component* of G is a (maximal) connected component of the underlying undirected graph of G; for simplicity, we will use the term *component* of a directed graph G to refer to a weakly connected component of G. Otherwise, we assume familiarity with basic graph theory, and refer to [5] for more information.

A *parameterized problem* is a set of instances of the form (x, k), where x is the input instance and $k \in \mathbb{N}$ is the *parameter*. A parameterized problem is *fixed-parameter tractable*, shortly FPT, if there is an algorithm that solves the problem in time $f(k)|x|^c$, where f is a

computable function and $c > 0$ is a constant. We refer to [6, 16] for more information about parameterized complexity.

3 Structural results

Let $\mathcal{T}$ be a triangulation and let $F = \langle f_1, \ldots, f_r \rangle$ be a valid sequence of flips with respect to $\mathcal{T}$. We denote by $\mathcal{T}_i$, for $i = 1, \ldots, r$, the triangulation that is the outcome of applying the (valid) subsequence of flips $\langle f_1, \ldots, f_i \rangle$ to $\mathcal{T}$.

▶ **Definition 1.** Let f_i and f_j be two flips in F such that $1 \leq i < j \leq r$. Flip f_j is said to be *adjacent* to flip f_i, denoted $f_i \to f_j$, if:
(1) either $\varphi(f_i) = \varepsilon(f_j)$ or $\varphi(f_i)$ and $\varepsilon(f_j)$ share a triangle in triangulation $\mathcal{T}_{j-1}$; and
(2) $\varphi(f_i)$ is not flipped between f_i and f_j, that is, there does not exist a flip f_p in F, where $i < p < j$, such that $\varepsilon(f_p) = \varphi(f_i)$.

The above adjacency relation defined on the flips in F can be naturally represented by a directed acyclic graph (DAG), denoted $\mathcal{D}_F$, where the nodes of $\mathcal{D}_F$ are the flips in F, and its arcs represent the (directed) adjacencies in F. Note that by definition, if $f_i \to f_j$ then $i < j$. For simplicity, we will label the nodes in $\mathcal{D}_F$ with the labels of their corresponding flips in F.

▶ **Lemma 2.** *Every node in $\mathcal{D}_F$ has indegree at most 5. Therefore, $|E(\mathcal{D}_F)| \leq 5 \cdot |V(\mathcal{D}_F)|$ and $|\mathcal{D}_F| \leq 6 \cdot |V(\mathcal{D}_F)|$.*

▶ **Lemma 3.** *Let $\mathcal{T}_0$ be a triangulation and let $F = \langle f_1, \ldots, f_r \rangle$ be a sequence of flips such that $\mathcal{T}_0 \xrightarrow{F} \mathcal{T}_r$. Let $\pi(F)$ be a permutation of the flips in F such that $\pi(F)$ is a topological sorting of $\mathcal{D}_F$. Then $\pi(F)$ is a valid sequence of flips such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$.*

Proof. Proceed by induction on $|F|$. If $|F| \leq 1$, then the statement obviously holds true. Suppose that the statement is true for any F such that $|F| < r$, where $r > 1$, and consider a sequence F such that $|F| = r$.

Let f_s be the last flip in $\pi(F)$. Since $\pi(F)$ is a topological sorting of $\mathcal{D}_F$, f_s must be a sink in $\mathcal{D}_F$. It follows that no flip after f_s in F is adjacent to f_s in $\mathcal{D}_F$. Let Q be the quadrilateral associated with $\varphi(f_s)$ in triangulation $\mathcal{T}_s$. Then no flips after f_s in F has its underlying edge in Q (*i.e.*, as a boundary edge of Q or as a diagonal of Q), which means that the two adjacent triangles forming Q in $\mathcal{T}_s$ remain unchanged throughout the flips after f_s in F. Therefore, we can safely move the flip f_s to the end of the sequence F without affecting the other flips in F nor the validity of F. Let this new sequence be F'; then it follows from the previous argument that $\mathcal{T}_0 \xrightarrow{F'} \mathcal{T}_r$. Since f_s appears at the end of F', $F' - f_s$ is a valid sequence with respect to $\mathcal{T}_0$ that transforms $\mathcal{T}_0$ into some triangulation $\mathcal{T}$ such that $\mathcal{T} \xrightarrow{f_s} \mathcal{T}_r$. Note that since f_s is a sink in $\mathcal{D}_F$, $\pi(F) - f_s$ is a permutation of the flips in $F' - f_s$ that is a topological sorting of $\mathcal{D}_F - f_s$. By the inductive hypothesis, $\pi(F) - \{f_s\}$ transforms $\mathcal{T}_0$ into $\mathcal{T}$. Since $\mathcal{T} \xrightarrow{f_s} \mathcal{T}_r$, appending f_s to the end of $\pi(F) - \{f_s\}$ results in $\pi(F)$ such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$. This completes the inductive proof. ◀

▶ **Corollary 4.** *Let $\mathcal{T}_0$ be a triangulation and let $F = \langle f_1, \ldots, f_r \rangle$ be a sequence of flips such that $\mathcal{T}_0 \xrightarrow{F} \mathcal{T}_r$. For any given ordering $(C_1, \ldots, C_\ell)$ of the components in $\mathcal{D}_F$, there is a permutation $\pi(F)$ of the flips in F such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$, and such that for any two flips $f_i \in C_t$ and $f_j \in C_s$, where $1 \leq t < s \leq \ell$, f_i appears before f_j in $\pi(F)$. That is, all the flips in the same component appear as a consecutive block in $\pi(F)$, and the order of the blocks in $\pi(F)$ is the same as the given order of their corresponding components.*

▶ **Definition 5.** Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of Flip Distance. An edge in $\mathcal{T}_{initial}$ that is not in $\mathcal{T}_{final}$ is called a *changed edge*. If a sequence F is a solution to the instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$, we call a component in $\mathcal{D}_F$ *essential* if the component contains a flip f such that $\varepsilon(f)$ is a changed edge, otherwise, the component is called *nonessential*.

▶ **Lemma 6.** *Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of* Flip Distance*, and suppose that F is a solution to the instance. Then every component of $\mathcal{D}_F$ is essential.*

Proof. Suppose, to get a contradiction, that $\mathcal{D}_F$ contains a nonessential component C. Let F_C be the subsequence of F consisting of the flips that are in C. We will show that $F - F_C$ is a solution to the instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$. This will contradict the minimality of F because the number of flips in F is the flip distance between $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$.

By Corollary 4, we can assume that all the flips in F_C appear consecutively (*i.e.*, as a single block) at the end of F. Let $\mathcal{T}'$ be the outcome of applying $F - F_C$ to $\mathcal{T}_{initial}$. It suffices to show that $\mathcal{T}' = \mathcal{T}_{final}$. Suppose that this is not the case. Since the number of edges in $\mathcal{T}'$ and $\mathcal{T}_{final}$ is the same, there must exist an edge $e \in \mathcal{T}'$ such that $e \notin \mathcal{T}_{final}$. Therefore, C must contain a flip f such that $\varepsilon(f) = e$; assume that f is the first such flip in C. Since C is nonessential, $e \notin \mathcal{T}_{initial}$, otherwise e would be a changed edge. Therefore, there must exist a flip f' in $F - F_C$ such that $\varphi(f') = e$; we can assume that f' is the last such flip in $F - F_C$. By the definition of adjacency in $\mathcal{D}_F$, there is an arc from node f' in $\mathcal{D}_F - C$ to node f in C, contradicting the assumption that C is a component of $\mathcal{D}_F$. ◀

Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of Flip Distance, and suppose that F is a solution for $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$. By Lemma 6, $\mathcal{D}_F$ does not contain nonessential components, and by Corollary 4, we can assume that all the flips in the same component of $\mathcal{D}_F$ appear as a consecutive block in F. We shall call such a solution F satisfying the above properties a *normalized* solution. Suppose that $F = \langle f_1, \ldots, f_k \rangle$ is a normalized solution to an instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ of Flip Distance, and let C be a component of $\mathcal{D}_F$. The following lemma provides several sufficient conditions for a directed path to exist between two flips in C:

▶ **Lemma 7.** *Let f_i and f_h, where $i < h$, be two flips in C. If one of the following conditions is true, then there is a directed path from f_i to f_h in C:*

(1) *$\varphi(f_h)$ crosses $\varepsilon(f_i)$.*
(2) *$\varphi(f_h) = \varepsilon(f_i)$.*
(3) *$\varepsilon(f_i) = \varepsilon(f_h)$.*
(4) *$\varphi(f_i) = \varepsilon(f_h)$, or $\varphi(f_i)$ and $\varepsilon(f_h)$ share a triangle T in $\mathcal{T}_j$, for some j satisfying $i \leq j < h$.*

4 The algorithm

Using the structural results in Section 3, it is not difficult to obtain an FPT algorithm for Flip Distance that runs in $\mathcal{O}^*(c^{k^2})$ time, for some constant c. For instance, starting from an edge in the current triangulation (which corresponds to a flip in the DAG representing the remaining solution), we can grow a BFS-like tree of size c^k searching for the next edge to flip (corresponding to a source node in the DAG), and flip this edge. Repeating this process k times gives an $\mathcal{O}^*(c^{k^2})$-time algorithm for the problem. Our goal, however, is to obtain an $\mathcal{O}^*(c^k)$-time algorithm for the problem, for some constant c. Achieving this goal turns out to be quite challenging, and requires a deep understanding of the structure of the problem. We did so by analyzing the relation between the DAG associated with a solution to a problem instance and the changing structure of the underlying triangulations.

4.1 Overview of the algorithm

In this subsection we give an intuitive description of how our algorithm works. Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of Flip Distance. In order to solve the instance, by Lemma 3, it suffices for the algorithm to perform a sequence of k flips that is a topological sorting of the DAG $\mathcal{D}_F$ associated with a normalized solution F to the instance. Needless to say, the difficulty is that we do not know F, nor do we know $\mathcal{D}_F$. By Lemma 6, each component of $\mathcal{D}_F$ is essential, and hence, must contain a changed edge. The algorithm starts by picking a changed edge e in $\mathcal{T}_{initial}$. There must exist a flip f in $\mathcal{D}_F$ such that $e = \varepsilon(f)$; let us refer to the component of $\mathcal{D}_F$ containing f by C. We explain next how the algorithm, starting at e in $\mathcal{T}_{initial}$, performs a sequence of flips that is a topological sorting of C; this can be easily extended to a sequence of flips that is a topological sorting of $\mathcal{D}_F$.

Clearly, the algorithm cannot start by performing f because other flips may precede f in the solution. Instead, the algorithm searches for an edge $\varepsilon(f_s)$ in $\mathcal{T}_{initial}$ that is the underlying edge of a source node f_s in C, and flips $\varepsilon(f_s)$. Now we explain how the algorithm searches for $\varepsilon(f_s)$ in $\mathcal{T}_{initial}$ without having access to C. The algorithm starts at edge e in $\mathcal{T}_{initial}$ and nondeterministically "takes a walk" in which each step/action consists of moving to an edge that shares a triangle with the edge that the algorithm is currently at; the number of such local actions is the length of the walk. We show (Lemma 10) that there exists a source node f_s in C such that, starting at the changed edge e, the algorithm can walk in the *current triangulation* $\mathcal{T}_{initial}$ from e to $\varepsilon(f_s)$, and that the length of this walk is at most the length of the path from f_s to f in C. Suppose that the algorithm nondeterministically guessed the right walk, and walked to $\varepsilon(f_s)$ in $\mathcal{T}_{initial}$. The algorithm then flips $\varepsilon(f_s)$, thus performing flip f_s in C, to obtain a new triangulation $\mathcal{T}_{current}$, and stays at the edge $\varphi(f_s)$ in $\mathcal{T}_{current}$. To continue the sequence of flips that corresponds to a topological sorting of C, the algorithm should flip next an edge in $\mathcal{T}_{current}$ that corresponds to a source node in the resulting DAG $C_{current} = C - f_s$. Hence, the algorithm needs to walk from $\varphi(f_s)$ in $\mathcal{T}_{current}$ to a source node in $C_{current}$, and to flip the edge corresponding to that source node in $\mathcal{T}_{current}$, and so on and so forth. To show how to perform this desired sequence of nondeterministic actions so that total number of actions remains linear in k, we define a spanning subgraph J_C of the underlying graph of C. We then show that there exists a sequence of local actions by the algorithm, in which the edge-flips is a topological sorting of C, that simulates a recursive traversal of J_C. This mapping of the actions of the algorithm to a specific traversal of J_C will allow us to "charge" the actions of the algorithm to the nodes and edges of J_C, thus obtaining the desired linear upper bound on the number of actions of the algorithm in terms of the size of J_C, and hence the size of C.

4.2 The nondeterministic actions of the algorithm

The algorithm is a nondeterministic algorithm that starts from a changed edge in a triangulation $\mathcal{T}_{initial}$ and performs a sequence of actions. The algorithm is equipped with a stack. Each action σ of the algorithm acts on some edge e in a triangulation that we refer to as the *current triangulation* (before σ), denoted $\mathcal{T}_{current}$. Initially $\mathcal{T}_{current} = \mathcal{T}_{initial}$, and $\mathcal{T}_{current}$ before action σ is the triangulation resulting from applying the sequence of actions preceding σ to $\mathcal{T}_{initial}$. Each action σ of the algorithm is of the following possible types:

(i) Move to one of the (at most 4) edges that share a triangle with e in $\mathcal{T}_{current}$.

(ii) Flip e, and move to one of the 4 edges that shared a triangle with e in $\mathcal{T}_{current}$.

(iii) Flip e, push the edge created by the flip into the stack, and move to one of the 4 edges that shared a triangle with e in $\mathcal{T}_{current}$.

(iv) Flip e, jump to the edge on the top of the stack.
(v) Flip e, jump to the edge on the top of the stack, and pop the stack.

A *walk* starting from an edge e in a triangulation is a sequence of actions all of which are of type (i). Since there are 4 choices for each action of types (i)-(iii) and 1 choice for each action of types (iv)-(v), we have:

▶ Proposition 8. The number of choices for any action by the algorithm is at most 14.

4.3 The sequence of actions on a component of $\mathcal{D}_F$

Let $F = \langle f_1, \ldots, f_k \rangle$ be a normalized solution to an instance of Flip Distance. Let C be a component of $\mathcal{D}_F$. By Corollary 4, we can assume that all the flips in C appear at the beginning of F, that is, form a prefix of F; let $F_C = \langle f_1, \ldots, f_t \rangle$ be the prefix of F corresponding to the flips in C. This subsection is dedicated to proving the following theorem:

▶ **Theorem 9.** *Let C be a component of $\mathcal{D}_F$. There is a sequence of actions for the nondeterministic algorithm of length at most $11|V(C)|$ that, starting from a changed edge $\varepsilon(f_h)$ for some $f_h \in C$, performs the flips in C in a topologically-sorted order.*

To prove the above theorem, we define a spanning subgraph J_C of the underlying graph of C recursively. We then exhibit a sequence of actions of the algorithm that can be depicted by a recursive traversal of J_C. By that we mean that the actions performed by the algorithm in the triangulations correspond to a traversal of the edges and nodes of J_C, and such that the sequence of edge-flips performed by the algorithm is a topological sorting of C. We initialize J_C to be empty, and we start the recursive definition of J_C at a node in C that corresponds to a changed edge in the current triangulation. We will then add edges and nodes to J_C, and recurse on the connected components of the graph resulting from C after a source node in C has been removed. Since during the recursion nodes and edges get removed from C, the resulting graph of C may consist of several connected components that we will refer to as *chunks*, in order to distinguish them from the components of $\mathcal{D}_F$. Assume that the current triangulation is $\mathcal{T}$ when we are recursing on a chunk H to define its spanning subgraph J_H. The recursive call starts at a node f_h in H that we call the *entry point* of H. At the top level of the recursion, C is the only chunk (in the recursive definition), and the entry point of $H = C$ is a node in C corresponding to a changed edge in the current triangulation. We will define in Lemma 10 a directed path $B = \langle b_1 = f_s, \ldots, b_\ell = f_h \rangle$ in H from a source node f_s in H to the entry point f_h of H. With the path B, we correspond a walk W, defined in Lemma 10, that the algorithm performs in the current triangulation from $\varepsilon(f_h)$ to $\varepsilon(f_s)$. We add B to J_H, we add the edges between f_s and the entry point of each chunk in $H - f_s$ to J_H, and we recurse on the chunks of $H - f_s$ to complete the recursive definition of J_H. The corresponding actions of the algorithm (with the recursive definition of J_H) consist of performing the walk W, flipping $\varepsilon(f_s)$, and recursively performing the sequence of actions corresponding to the traversals of the chunks in $H - f_s$. Note that to flip a single edge, the algorithm takes a walk in the current triangulation to a source node in C. Therefore, if we are not careful in how we do the traversal of C, the length of all these walks could be quadratic in k. To ensure that when the algorithm is done performing the sequence of actions in a chunk it can go back to continue with the other chunks, the algorithm uses a stack to store the edge $\varphi(f_s)$, resulting from flipping the "connecting node" f_s of all these chunks, so that the algorithm, after performing all the flips in a chunk of $H - f_s$, can go back by a single action to $\varphi(f_s)$. We start with the following lemma:

▶ **Lemma 10.** *Let f_h be a node in a chunk H such that $\varepsilon(f_h)$ is an edge in the current triangulation $\mathcal{T}$. There is a walk W in $\mathcal{T}$ from $\varepsilon(f_h)$ to an edge $\varepsilon(f_s)$ in $\mathcal{T}$, where f_s is a source node of H, such that there is a directed path B from f_s to f_h in H that we refer to as the backbone of H. Moreover, the length of the walk W is at most the length of B.*

Proof. If f_h is a source in H then $f_h = f_s$, the path B consists of f_s, and the length of the walk W is 0. The statement is trivially true in this case. Now assume that f_h is not a source node in H.

Since $\varepsilon(f_h)$ is an edge in $\mathcal{T}$, let Q be the quadrilateral associated with $\varepsilon(f_h)$ in $\mathcal{T}$. Since f_h is not a source in H and F is a minimal solution, one of the edges on the boundary of Q must be flipped before f_h; let f_p be the first such flip in H. Since $\varepsilon(f_p)$ and $\varepsilon(f_h)$ share a triangle in $\mathcal{T}_{p-1}$, $\varphi(f_p)$ and $\varepsilon(f_h)$ share a triangle in $\mathcal{T}_p$, and by Lemma 7, there is a directed path from f_p to f_h in the component C of $\mathcal{D}_F$. Since the nodes removed from C during the recursive definition are always source nodes in their current chunks, there is a directed path from f_p to f_h in the current chunk H. The edges $\varepsilon(f_h)$ and $\varepsilon(f_p)$ share a triangle in $\mathcal{T}$, and hence, in one action (of type (i)) the algorithm can go from $\varepsilon(f_p)$ to $\varepsilon(f_h)$ in $\mathcal{T}$.

If f_p is a source node in H, then we are done; otherwise, applying the above argument to f_p, we can find a flip f_q such that $\varepsilon(f_p)$ and $\varepsilon(f_q)$ share a triangle in $\mathcal{T}$ and there is a directed path from f_q to f_p in H. We can repeat this process until we reach a source node f_s in H. Going from $\varepsilon(f_h)$ to $\varepsilon(f_s)$ in $\mathcal{T}$ involves only actions of type (i), and hence, defines a walk W from $\varepsilon(f_h)$ to $\varepsilon(f_s)$ in $\mathcal{T}$. The length of W is at most the total number of flips in a directed path B from f_s to f_h in H, which is composed of the directed paths defined in the process described above (from f_p to f_h, f_q to f_p, and so on). ◀

We now formally give the recursive definition of J_C, described for a chunk H with entry point f_h of a graph resulting from C during the recursion. Recall that at the top level of the recursion $H = C$, and f_h is a node in C corresponding to changed edge in the starting triangulation.

▶ **Definition 11.** Let H be a chunk with entry point f_h. The subgraph J_H of H is defined recursively as follows.

(1) Let $B = \langle f_s = b_1, \ldots, b_\ell = f_h \rangle$, where f_s is a source node in H (possibly $f_h = f_s$), be the backbone of H defined in Lemma 10.

(2) Remove f_s from H and let $H_1, \ldots, H_x$ be the chunks of $H^- = H - f_s$; define f_s to be the *connecting point* to each of the chunks $H_1, \ldots, H_x$.

(3) For each chunk H_p, $p = 1, \ldots, x$, if H_p contains nodes from previously-defined backbones during the recursive definition, then let f_{min} be the node in H_p of minimum index (with respect to F) that belongs to a previously-defined backbone; define the *entry point* of H_p to be the node f_{h_p} in H_p that is adjacent to f_s and that has a path to f_{min} in H_p, and in case more than one neighbor of f_s satisfies this property pick the neighbor with the minimum index with respect to F (we will prove in Lemma 12 that the node f_{h_p} is well defined). Otherwise (H_p does not contains nodes from previously-defined backbones), define the entry point of H_p to be the flip f_{h_p} in H_p with the minimum index h_p (with respect to F) that is adjacent to f_s. (See Figure 1 for illustration.)

(4) Define the subgraph J_p of H_p with entry point f_{h_p} recursively, for $p = 1, \ldots, x$.

(5) Define J_H to be the union of the edges in B, the edges in J_p and the edges between f_s and each entry point of H_p, for $p = 1, \ldots, x$.

Let J_C be the subgraph of the underlying graph of C resulting from applying the above recursive definition to C starting at a flip corresponding to a changed edge in C. We have:

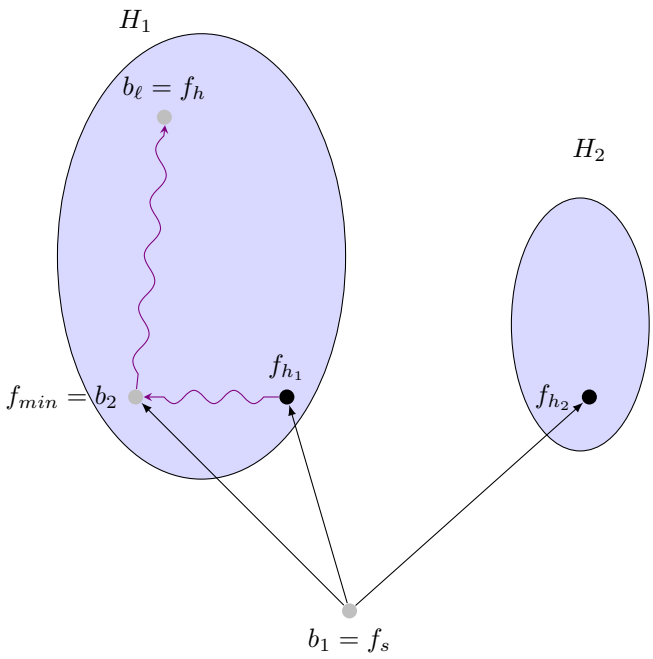

Figure 1 Illustration of the definition of entry points, where backbone nodes are colored gray. The entry point of H_1 is the node f_{h_1} in H_1 adjacent to f_s that has a path to the backbone node f_{min} with the minimum index (node b_2 in this case). The entry point of H_2, which does not contain backbone nodes in this case, is the node f_{h_2} of minimum index (h_2) that is adjacent to f_s.

▶ **Lemma 12.** *All the backbones, defined during the recursive definition of J_C, that exist in the same chunk are edge disjoint, and belong to a single (simple) path in the chunk; on this path the (remaining) nodes from each backbone appear consecutively. Moreover, the entry node of a chunk, defined in step 3 of Definition 11, is well-defined.*

Proof. The proof is by induction on the number of recursive steps (depth of the recursion) taken to form a chunk. The statement is clearly true at the top level of the recursion where the only chunk is C, whose entry point is defined to be a flip corresponding to a changed edge, and there is only one defined backbone. Suppose now that chunk H_p resulted from a chunk H in one recursive step, and that the statement is true for H (inductive hypothesis).

H_p was obtained by removing a source node f_s from H, which is a backbone node. By the inductive hypothesis, all the backbones in H are edge-disjoint and belong to a path P in H. Since f_s is a source node in H, f_s must be the tail of P. If H_p does not contain any previously-defined backbone nodes, then the entry point f_{h_p} of H_p is defined to be the node in H_p with the minimum index that is adjacent to f_s, and in this case f_{h_p} is well-defined. Moreover, there is only one backbone in H_p. Therefore, the statement of the lemma is true in this case. Suppose now that H_p contains at least one node from a previously-defined backbone. Because the underlying graph of H_p is connected and $P^- = P - f_s$ is a path, it follows that H_p contains P^-. Let b be the node adjacent to f_s on P^-, *i.e.*, the tail of P^-. By the inductive hypothesis, P^- contains all the previously-defined backbone nodes in H_p, and in particular, P^- contains the node f_{min} in H_p of minimum index (the minimum index is with respect to F) that belongs to a previously-defined backbone. Since b is adjacent to f_s, it follows from the preceding that node f_{h_p} is well-defined because b satisfies that it is

adjacent to f_s and there is a path from b to the backbone node in H with the minimum index, namely f_{min} (possibly b itself). Now let B_{H_p} be the backbone of H_p. Since B_{H_p} is a path whose head is f_{h_p}, and since — by the choice of f_{h_p} — there is a path from f_{h_p} to f_{min}, the indices of the nodes in B_{H_p} are not larger than the index of f_{min}, which is the backbone node on P^- of the minimum index. Therefore, the set of edges in B_{H_p} is disjoint from the set of backbone edges on P^- (and hence, from the set of backbone edges in H_p) that belong to previously-defined backbones. Since B_{H_p} is a path in H_p, and since there is a path from f_{h_p} to f_{min} in H_p, all the backbone edges in H_p form a path in which all the (remaining) nodes of each backbone appear consecutively. This completes the inductive proof. ◀

▶ **Corollary 13.** *All the backbones defined in the recursive definition of J_C are edge-disjoint.*

Proof. It suffices to show that when a backbone B of a chunk H is defined during the recursive definition of J_C, the edges of B are different from the edges of all previously-defined backbones. Clearly, the edges of B are different from those of the backbones in chunks other than H, and from the edges of previously-defined backbones that have been previously removed during the recursive definition of J_C. By Lemma 12, the edges of B are different from those of the backbones other than B that (may) exist in H. The statement follows. ◀

▶ **Lemma 14.** *Let C be a component of $\mathcal{D}_{\mathcal{F}}$. The subgraph J_C formed by applying Definition 11 to C is a spanning subgraph of the underlying graph of C.*

Proof. The statement follows from the connectedness of C and Definition 11 by a simple inductive argument: J_C contains a source node f_s of C and an edge from f_s to each chunk in $C - f_s$. ◀

We define next a sequence of actions that the algorithm performs starting at a changed edge (corresponding to a node in C) in the current triangulation and that corresponds to a traversal of J_C. Let f_i, f_h be the connecting and the entry points to a chunk $H \neq C$, respectively. At the top level of the recursion, where $H = C$ is a component of $\mathcal{D}_F$, define f_h to be a flip in C whose underlying edge $\varepsilon(f_h)$ is a changed edge (f_i need not be defined).

▶ **Definition 15.** Let H be a chunk with entry point f_h. The sequence of actions of the nondeterministic algorithm on H is defined as follows.

- **(a)** The nondeterministic algorithm performs the walk W from $\varepsilon(f_h)$ to $\varepsilon(f_s)$ (in the current triangulation $\mathcal{T}$) defined in Lemma 10 that corresponds to the backbone $B = \langle f_s = b_1, \ldots, b_\ell = f_h \rangle$ of H.
- **(b)** The nondeterministic algorithm flips the edge $\varepsilon(f_s)$.
- **(c)** The algorithm nondeterministically pushes $\varphi(f_s)$ into the stack if there is more than one chunk in $H^- = H - f_s$, and moves to the entry point of the first chunk in H^-.
- **(d)** The nondeterministic algorithm recursively performs the sequence of actions on each chunk of H^-, nondeterministically moving to the edge $\varphi(f_s)$ on the top of the stack when performing the last action in each chunk, and following that with a move (if needed) to the underlying edge of the entry point of a new chunk, which shares a triangle with $\varphi(f_s)$ (or is identical to it) by Lemma 16 below.
- **(e)** The algorithm nondeterministically moves to the top of the stack and pops the stack after performing the last action in the last chunk of H^- (in case there is more than one).

▶ **Lemma 16.** *Let f_i, f_h be the connecting and entry points to a chunk $H \neq C$, respectively. Suppose that the current triangulation is $\mathcal{T}$ when the sequence of actions of the algorithm defined in Definition 15 is applied on H. Then either $\varphi(f_i) = \varepsilon(f_h)$, or $\varphi(f_i)$ and $\varepsilon(f_h)$ share a triangle in $\mathcal{T}$.*

Proof (Theorem 9). It is clear that the order of the flips performed by the algorithm in the sequence of actions described in Definition 15 corresponds to a topological sorting of C because every flip corresponds to the removal of a source node from a DAG resulting from C in the recursive definition of J_C, and because J_C is a spanning subgraph of C by Lemma 14. Therefore, it suffices to show that this sequence has length at most $11|V(C)|$. To do so, we charge the actions of the algorithms to the nodes and edges of J_C.

When invoked on a chunk H, the algorithm starts at an entry node f_h of H; initially $H = C$ and f_h is a node in C whose underlying edge is a changed edge in the current triangulation $\mathcal{T}$. In Lemma 10 we showed that there is a path $B = \langle f_s = b_1, \ldots, b_\ell = f_h \rangle$ from a source node f_s in H to f_h that corresponds to a walk by the algorithm from edge $\varepsilon(f_h)$ to $\varepsilon(f_s)$ in $\mathcal{T}$; moreover, the length of this walk is at most the length of B. The algorithm can perform this walk using actions of type (i) (as defined in Subsection 4.2), and the number of such actions is at most the length of B. When the algorithm is at edge $\varepsilon(f_s)$ in $\mathcal{T}$, it flips edge $\varepsilon(f_s)$, which is one action either of type (ii) or (iii). Next, the algorithm recurses on each chunk H_p of $H - f_s$, starting at the entry point f_{h_p} of H_p. In Lemma 16, we showed that the edges $\varphi(f_s)$ and $\varepsilon(f_{h_p})$ are either identical, or they share a triangle in the current triangulation when the algorithm is recursively called on H_p. Hence, in at most one action the algorithm can move from $\varphi(f_s)$ to $\varepsilon(f_{h_p})$. If there is more than one chunk in $H - f_s$ (the algorithm nondeterministically decides), the algorithm pushes $\varphi(f_s)$ into the stack after flipping f_s. In case there is only one chunk left, the algorithm also pops the stack after jumping to the top of the stack. It is not difficult to see that each of the steps corresponds to one action of the algorithm from types (i)-(v).

To prove that the length of the sequence of actions is at most $11|V(C)|$, we charge these actions to the nodes and edges of J_C. The sequence of actions can be classified into two categories: actions with flips and actions without flips. The number of actions with flips is at most the number of nodes in J_C, which is $|V(C)|$. Note that actions that involve moving to the top of the stack, or popping the stack, or both, are combined with flips, and hence have been accounted for. The actions without flips are all of type (i), and can be further divided into two groups: (I) those done in a walk W corresponding to a backbone B of a chunk H, and (II) those done when the algorithm moves from an underlying edge $\varphi(f_s)$ (on the top of the stack) of a source node f_s in a chunk H to an edge whose corresponding node is an entry point of a chunk resulting from removing f_s from H. The number of actions in group (I) is at most $|E(C)|$; this is because, by Corollary 13, the edges of different backbones are distinct, and hence the total number of such edges (and hence actions in group (I)) is at most $|E(J_C)| \leq |E(C)|$. To bound the number of actions in group (II), observe that each such action corresponds to an edge in C from f_s to the entry point of a chunk resulting from removing f_s from H. Since f_s is removed from J_C upon making the recursive calls to the resulting chunks, we can charge each such action in a one-to-one fashion to edges of $E(J_C)$. Therefore, the number of actions in group (II) is at most $|E(J_C)| \leq |E(C)|$. Therefore, the total number of actions of type (i) is at most $2|E(J_C)| \leq 2|E(C)|$. It follows that the total number of actions performed by the algorithm when applied to C is at most $|V(C)| + 2|E(J_C)| \leq 11|V(C)|$ (by Lemma 2).

◀

4.4 Putting all together: the whole algorithm

Let F be a normalized solution to the instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$. Order the changed edges arbitrarily, and denote this ordering by $\mathcal{O}$. The algorithm starts by guessing the number of components t, where $t \leq k$, in $\mathcal{D}_F$. The algorithm then guesses the number of flips

$k_1, \ldots, k_t$ in the components $C_1, \ldots, C_t$, respectively, of $\mathcal{D}_F$ satisfying $k_1, \ldots, k_t \geq 1$ and $k_1 + k_2 + \ldots + k_t = k$. Fix such a guess $(k_1, \ldots, k_t)$.

The algorithm performs t iterations: $\ell = 1, \ldots, t$. We define $\mathcal{T}^{\ell}_{initial}$, $\ell = 1, \ldots, t$, to be the triangulation that resulted from $\mathcal{T}_{initial}$ after the flips in the first ℓ iterations are performed. We define $\mathcal{T}^{0}_{initial} = \mathcal{T}_{initial}$. For each $\ell = 1, \ldots, t$, do the following: Pick the next edge $e \in \mathcal{O}$. If e is not a changed edge anymore with respect to $\mathcal{T}^{\ell-1}_{initial}$ and $\mathcal{T}_{final}$, then skip to the next edge in $\mathcal{O}$. Otherwise (e is in $\mathcal{T}^{\ell-1}_{initial}$ but not in $\mathcal{T}_{final}$), perform a sequence of actions starting from e in $\mathcal{T}^{\ell-1}_{initial}$ until either the number of flips performed is k_ℓ, or the number of actions performed reaches $11k_\ell$. Let F_ℓ be the sequence of flips performed in the current iteration, and note that $\mathcal{T}^{\ell-1}_{initial} \xrightarrow{F_\ell} \mathcal{T}^{\ell}_{initial}$. After the last iteration $\ell = t$, if $\mathcal{T}^{t}_{initial} = \mathcal{T}_{final}$ then accept; otherwise reject.

▶ **Theorem 17.** *Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of* Flip Distance. *The above nondeterministic algorithm decides $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ correctly, and it can be simulated by a deterministic algorithm that runs in time $\mathcal{O}(n + k \cdot c^k)$.*

Proof. It is easy to see that the correctness of the algorithm follows from the following: (1) there is a guess for the algorithm of the correct number of components t, and of $(k_1, \ldots, k_t)$ such that k_i is the exact number of flips in C_i, $i = 1, \ldots, t$; and (2) by Theorem 9, there is a nondeterministic sequence of actions by the algorithm of length at most $11k_i$ that, starting from a changed edge in C_i, performs the k_i flips in C_i in a topologically-sorted order.

We only need to analyze the deterministic running time needed to simulate the nondeterministic algorithm. The initial processing of the triangulations to find the changed edges takes $\mathcal{O}(n)$ time. The total number of sequences $(k_1, \ldots, k_t)$, for $t = 1, \ldots, k$, satisfying $k_1 + \ldots + k_t = k$ and $k_1, \ldots, k_t \geq 1$, is known as the *composition number* of (integer) k, and is equal to 2^{k-1}. For each such sequence $(k_1, \ldots, k_t)$, we iterate through the numbers $k_1, \ldots, k_t$ in the sequence. For a number k_i, $1 \leq i \leq t$, by Theorem 9, we need to try every sequence of at most $11k_i$ actions, and in which each action is one of 14 choices (by Proposition 8). Therefore, the number of such sequences is at most 14^{11k_i}. It follows that the total number of sequences that the algorithm needs to enumerate to find a witness to the solution (if it exists) is at most: $\sum_{t=1}^{k} \sum_{(k_1, \ldots, k_t)} (14^{11k_1} \times \ldots \times 14^{11k_t}) = \mathcal{O}(2^{k-1} 14^{11k}) = \mathcal{O}(c^k)$, where $c \leq 2 \cdot 14^{11}$. Since each sequence of actions can be carried out in time $\mathcal{O}(k)$, and the resulting triangulation at the end of the sequence can be compared to $\mathcal{T}_{final}$ in $\mathcal{O}(k)$ time as well, the running time for each enumerated sequence is $\mathcal{O}(k)$. It follows from the above that the running of the deterministic algorithm is $\mathcal{O}(n + k \cdot c^k)$. Finally we point out that the algorithm needs to decide whether k is the flip distance between $\mathcal{T}_{initial}$ and $\mathcal{T}_{final}$, which means that no sequence of flips of length smaller than k exists that transforms $\mathcal{T}_{initial}$ to $\mathcal{T}_{final}$. This can be decided by invoking the algorithm on each of the instances $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k')$, for $k' = 0, \ldots, k$. The running time remains $\mathcal{O}(n + k \cdot c^k)$ because $\sum_{k'=0}^{k} \mathcal{O}(k' \cdot c^{k'}) = \mathcal{O}(k \cdot c^k)$. ◀

5 Concluding remarks

Improving the upper bound $2 \cdot 14^{11}$ on the constant c in the running time of our algorithm to a small value is an important open problem. Another important open problem is investigating the kernelization of Flip Distance. One can obtain an exponential-size kernel based on the results in this paper, but the question of whether there is a polynomial-size kernel is important and challenging. Recall that a kernel of size $2k$ was given by Lucas [12] for the convex case. Finally, we note that Flip Distance falls broadly into the category of reconfiguration problems, for which several parameterized complexity results appeared very recently (see [13, 14, 15]).

References

1 O. Aichholzer, F. Hurtado, and M. Noy. A lower bound on the number of triangulations of planar point sets. *Computational Geometry*, 29(2):135–145, 2004.

2 O. Aichholzer, W. Mulzer, and A. Pilz. Flip distance between triangulations of a simple polygon is NP-complete. In *Proceedings of ESA*, volume 8125 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2013.

3 P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009.

4 S. Cleary and K. St. John. Rotation distance is fixed-parameter tractable. *Information Processing Letters*, 109(16):918–922, 2009.

5 R. Diestel. Graph Theory. Springer, Berlin, 4th edition, 2010.

6 R. Downey and M. Fellows. Parameterized Complexity. Springer, New York, 1999.

7 S. Hanke, T. Ottmann, and S. Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.

8 F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

9 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

10 A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. In *Proceedings of CCCG*, pages 119–124, 2012.

11 A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. arXiv.org e-Print archive, paper cs.CG/1205.2425, May 2012.

12 J. Lucas. An improved kernel size for rotation distance in binary trees. *Information Processing Letters*, 110(12):481–484, 2010.

13 A. Mouawad, N. Nishimura, and V. Raman. Vertex cover reconfiguration and beyond. In *Proceedings of ISAAC*, volume 8889 of *Lecture Notes in Computer Science*, pages 452–463. Springer, 2014.

14 A. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. In *Proceedings of IPEC*, volume 8246 of *Lecture Notes in Computer Science*, pages 281–294. Springer, 2013.

15 A. Mouawad, N. Nishimura, V. Raman, and M. Wrochna. Reconfiguration over tree decompositions. In *Proceedings of IPEC*, volume 8894 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2014.

16 R. Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press, USA, 2006.

17 A. Okabe, B. Boots, and K. Sugihara. Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons, New York, NY, USA, 1992.

18 A. Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014.

19 A. Saalfeld. Joint triangulations and triangulation maps. In *Proceedings of SoCG*, pages 195–204. ACM, 1987.

20 L. Schumaker. Triangulations in CAGD. *IEEE Computer Graphics and Applications*, 13(1):47–52, 1993.

21 R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1978.

22 D. Sleator, R. Tarjan, and W. Thurston. Rotation distance, triangulations, and hyperbolic geometry. In *Proceedings of STOC*, pages 122–135. ACM, 1986.

23 K. Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

24 D. Watson and G. Philip. Systematic triangulations. *Computer Vision, Graphics, and Image Processing*, 22(2):310, 1983.