



An $\mathcal{O}(3.82^k)$ Time \mathcal{FPT} Algorithm for Convex Flip Distance

Haohong Li 

Department of Computer Science, Lafayette College, Easton, PA, USA

Ge Xia 

Department of Computer Science, Lafayette College, Easton, PA, USA

Abstract

Let \mathcal{P} be a convex polygon in the plane, and let \mathcal{T} be a triangulation of \mathcal{P} . An edge e in \mathcal{T} is called a diagonal if it is shared by two triangles in \mathcal{T} . A *flip* of a diagonal e is the operation of removing e and adding the opposite diagonal of the resulting quadrilateral 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 CONVEX FLIP DISTANCE problem asks if the flip distance between two given triangulations of \mathcal{P} is at most k , for some given parameter $k \in \mathbb{N}$.

We present an \mathcal{FPT} algorithm for the CONVEX FLIP DISTANCE problem that runs in time $\mathcal{O}(3.82^k)$ and uses polynomial space, where k is the number of flips. This algorithm significantly improves the previous best \mathcal{FPT} algorithms for the problem.

2012 ACM Subject Classification Theory of computation \rightarrow Computational geometry; Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Flip distance, Rotation distance, Triangulations, Exact algorithms, Parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2023.44

Related Version *Full Version:* <https://arxiv.org/abs/2209.13134>

1 Introduction

Let \mathcal{P} be a convex polygon in the plane. A *triangulation* of \mathcal{P} adds edges between non-adjacent points in \mathcal{P} to produce a planar graph \mathcal{T} such that all faces except the outer face in \mathcal{T} are triangles. The edges in the triangulation \mathcal{T} not on the convex hull are called *diagonals*. The number of diagonals in \mathcal{T} is denoted by $\phi(\mathcal{T})$.

A *flip* of a diagonal e in \mathcal{T} removes e and adds the opposite diagonal of the resulting quadrilateral, thus transforming \mathcal{T} to another triangulation \mathcal{T}' of \mathcal{P} . Given two triangulations \mathcal{T}_{init} and \mathcal{T}_{final} of \mathcal{P} , the *flip distance* between them, denoted as $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final})$, is the minimum number of flips required to transform \mathcal{T}_{init} to \mathcal{T}_{final} (or equivalently from \mathcal{T}_{final} to \mathcal{T}_{init}). The CONVEX FLIP DISTANCE problem is formally defined as:

CONVEX FLIP DISTANCE

Given: Two triangulation \mathcal{T}_{init} and \mathcal{T}_{final} of a convex polygon \mathcal{P} in the plane.

Parameter: k .

Question: Is $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final})$ at most k ?

The number of ways to triangulate a convex $(n + 2)$ -gon is C_n , the n -th Catalan number. Consequently, there exists an isomorphism between triangulations of a convex polygon and a plethora of counting problems, such as binary trees, Dyck paths, etc [28]. In particular, there exists a bijection between the set of triangulations of a convex $(n + 2)$ -gon and the set of full binary trees with n internal nodes. Furthermore, flipping an edge in a convex polygon triangulation corresponds to rotating a node in a binary tree, which is an essential operation for maintaining balanced binary search trees. The *rotation distance* between two binary trees



© Haohong Li and Ge Xia;

licensed under Creative Commons License CC-BY 4.0

40th International Symposium on Theoretical Aspects of Computer Science (STACS 2023).

Editors: Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté;

Article No. 44; pp. 44:1–44:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



with the same number of nodes is the minimum number of rotations needed to transform one binary tree into the other. Thus, finding the flip distance between two triangulations of a convex $(n + 2)$ -gon is equivalent to finding the rotation distance between two binary trees with n internal nodes.

Determining the computational complexity of CONVEX FLIP DISTANCE (equivalently the rotation distance problem between binary trees) has been an important open problem. Despite extensive research on this problem [9, 3, 6, 22, 24, 27, 19, 12, 8, 7, 21, 5], it is still unknown whether it is NP-hard or not.

In 1982, Culik and Wood [9] first studied the rotation distance problem between binary trees and proved an upper bound of $2n - 2$, where n is the number of internal nodes of the trees. In 1988, Sleator, Tarjan, and Thurston [27] improved the upper bound to $2n - 6$, which is proven to be tight for $n \geq 11$ by Pournin [26].

In 1998, Li and Zhang gave two polynomial-time approximation algorithms [19] for computing the flip distance of convex polygon triangulations. One algorithm has an approximation ratio of $2 - \frac{2}{4(d-1)(d+6)+1}$ given that each vertex is an endpoint of at most d diagonals. Another algorithm has an approximation ratio of 1.97 provided that both triangulations do not contain internal triangles. Cleary and St. John [8] gave a linear-time 2-approximation algorithm for rotation distance in 2009.

In 2009, Cleary and St. John [7] gave a kernel of size $5k$ for CONVEX FLIP DISTANCE and presented an $\mathcal{O}^*((5k)^k)$ -time fixed-parameter tractable (\mathcal{FPT}) algorithm based on this kernel (the \mathcal{O}^* notation suppresses polynomial factors in the input size). The upper bound on the kernel size of CONVEX FLIP DISTANCE was subsequently improved to $2k$ by Lucas [21], who also gave an $\mathcal{O}^*(k^k)$ -time \mathcal{FPT} algorithm for this problem. Very recently, Celvo and Kelk improved the kernel size to $(1 + \epsilon)k$ for any $\epsilon > 0$ in 2021 [5], although their result does not lead to an improved approximation algorithm over [19] and [8].

The generalized version of the CONVEX FLIP DISTANCE problem, referred to as the GENERAL FLIP DISTANCE problem, is the flip distance between triangulations of a point set in general positions in the plane. The GENERAL FLIP DISTANCE problem is also a fundamental and challenging problem, and has also been extensively studied [18, 1, 2, 4, 7, 13, 14, 20, 25, 17, 11].

In 1972, Lawson [18] gave an $\mathcal{O}(n^2)$ upper bound on GENERAL FLIP DISTANCE, where n is the number of points in \mathcal{S} [18]. The complexity of the GENERAL FLIP DISTANCE problem was resolved in 2012 by Lubiw and Pathak [20] who showed the problem to be \mathcal{NP} -complete. Simultaneously, and independently, the problem was shown to be \mathcal{APX} -hard by Pilz [25]. In 2015, Aichholzer Mulzer, and Pilz [2] proved that the flip distance problem is \mathcal{NP} -complete for triangulations of a simple (but not convex) polygon.

Very recently, Kanj, Sedgwick, and Xia [16] presented the first \mathcal{FPT} algorithm for GENERAL FLIP DISTANCE that runs in $\mathcal{O}(n + kc^k)$, where $c \leq 2 \cdot 14^{11}$. Their approach defines a dependency relation for a sequence of flips: some flips required some other flips to be performed first. They proved that any topological sort of the directed acyclic graph (DAG) modeling this dependency relation yields the equivalent end result. Their algorithm simulates a “non-deterministic walk” that tries the possible flips to find a topological sort of the DAG representing an optimal solution. Refining this approach, Feng, Li, Meng, and Wang improved the \mathcal{FPT} algorithm to run in $\mathcal{O}(n + k \cdot 32^k)$ [11], which currently stands as the best \mathcal{FPT} algorithm for both GENERAL FLIP DISTANCE and CONVEX FLIP DISTANCE. No more efficient polynomial space algorithm was known for CONVEX FLIP DISTANCE despite its structural properties. We note in passing here that although a straightforward algorithm based on breadth-first search (BFS) can find the flip distance between triangulations of a convex polygon in time $\mathcal{O}^*(C_n) = \mathcal{O}^*(4^n)$, where C_n is the n^{th} Catalan number, it requires exponential space.

In this paper, we present an \mathcal{FPT} algorithm for the CONVEX FLIP DISTANCE problem that runs in time $\mathcal{O}(3.82^k)$ and uses polynomial space, where k is the number of flips. Instead of performing a “non-deterministic walk” as in [16, 11], our algorithm computes a topological sort of the DAG representing an optimal solution by repeatedly finding and removing its source nodes. This approach allows us to take advantage of the structural properties of CONVEX FLIP DISTANCE and design a simple yet significantly more efficient algorithm.

Our algorithm is closely related to algorithms in [16] and [11]. All three algorithms rely on the same structure results from [16] which state that (1) the flips in an optimal solution have a dependency relation that can be modeled as a DAG, and (2) any topological sort of this DAG yields an optimal solution. Therefore, three algorithms address the same problem: how to find a topological sort of this unknown DAG. The algorithm of Kanj, Sedgwick, and Xia [16] simulates a “non-deterministic walk”. For any diagonal e , the algorithm will either flip e (when this flip is a source in the DAG) or move on to one of e ’s neighbors (when a neighbor of e must be flipped before e can be flipped). Through a sequence of such “flip/move”-type local actions, a topological sort of the DAG (if exists) is found in time $\mathcal{O}(n + kc^k)$, where $c \leq 2 \cdot 14^{11}$. The algorithm of Feng, Li, Meng, and Wang [11] refines the “non-deterministic walk” approach by reducing the number of the action types and streamlining the backtracking in the walk, resulting in an improved algorithm that runs in time $\mathcal{O}(n + k \cdot 32^k)$. Different from the previous “walk”-based approach, our algorithm performs a topological sort of the DAG by repeatedly finding and removing source nodes in the DAG, similar to Kahn’s algorithm [15]. After the current source nodes of the DAG are removed, the new source nodes can be found among the neighbors of the flipped diagonals. Based on this more efficient approach and by exploiting the structural properties of CONVEX FLIP DISTANCE, we significantly improve the running time of our algorithm to $\mathcal{O}(3.82^k)$.

2 Preliminaries

2.1 Flips, triangulations, and flip distance

For any flip f , we use the notation f^{\leftarrow} to denote the underlying diagonal e on which f is performed, and the notation f^{\rightarrow} to denote the new diagonal \bar{e} added when f is performed. For any two diagonals e_1 and e_2 in \mathcal{T} , we say they are *neighbors* if they appear in the same triangle and say they are *independent* if they are not neighbors. Note that two independent diagonals in \mathcal{T} can share an endpoint, as long as they are not in the same triangle.

Let \mathcal{T} and \mathcal{T}' be two triangulations of \mathcal{P} . We refer to $(\mathcal{T}, \mathcal{T}')$ as a *pair of triangulations* of \mathcal{P} . Denote by $C(\mathcal{T}, \mathcal{T}')$ the number of common diagonals shared by \mathcal{T} and \mathcal{T}' . We say a sequence of flips $F = \langle f_1, \dots, f_r \rangle$ *transforms* a triangulation \mathcal{T} to \mathcal{T}' , denoted as $\mathcal{T} \xrightarrow{F} \mathcal{T}'$, if there exist triangulations $\mathcal{T}_0, \dots, \mathcal{T}_r$ such that $\mathcal{T}_0 = \mathcal{T}$, $\mathcal{T}_r = \mathcal{T}'$, and performing flip f_i in \mathcal{T}_{i-1} results in \mathcal{T}_i , for $i = 1, \dots, r$. Such a transformation $\mathcal{T} \xrightarrow{F} \mathcal{T}'$ is referred to as a *path* from \mathcal{T} to \mathcal{T}' following F . The *length* of F (equivalently, the length of the path $\mathcal{T} \xrightarrow{F} \mathcal{T}'$), denoted $|F|$, is the number of flips in it. The flip distance $\text{Dist}(\mathcal{T}, \mathcal{T}')$ is the length of the shortest path between \mathcal{T} and \mathcal{T}' . A sequence of flips F such that $\mathcal{T} \xrightarrow{F} \mathcal{T}'$ is a shortest path is called an *optimal* (or *minimum*) *solution* of the pair $(\mathcal{T}, \mathcal{T}')$.

Let f_i and f_j be two flips in $F = \langle f_1, \dots, f_r \rangle$ such that $1 \leq i < j \leq r$. The flip f_j is said to be *adjacent* to the flip f_i , denoted $f_i \rightarrow f_j$, if f_i^{\rightarrow} is a neighbor of f_j^{\leftarrow} in \mathcal{T}_{j-1} . This adjacency relation defines a partial order among flips in F : if $f_i \rightarrow f_j$ then f_i must precede f_j because f_i^{\rightarrow} is a neighbor of f_j^{\leftarrow} at the moment when f_j is performed. Therefore, the adjacency relation 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 .

44:4 An $\mathcal{O}(3.82^k)$ Time \mathcal{FPT} Algorithm for Convex Flip Distance

Recall that a topological sort of a DAG is *any* ordering of its nodes that satisfies: For any directed arc (u, v) in the DAG, u appears before v in the ordering. There could be many different topological sorts of \mathcal{D}_F , but the following lemma by Kanj, Sedgwick, and Xia [16] asserts that all of them yield the same outcome:

► **Lemma 1** ([16]). *Let \mathcal{T}_0 be a triangulation and let $F = \langle f_1, \dots, f_r \rangle$ 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 sort of \mathcal{D}_F . Then $\pi(F)$ is a valid sequence of flips such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$. Furthermore, the DAG $\mathcal{D}_{\pi(F)}$, defined based on the sequence $\pi(F)$, is the same directed graph as \mathcal{D}_F .*

► **Definition 2.** *Let $(\mathcal{T}_{init}, \mathcal{T}_{final})$ be a pair of triangulations. Let e be a diagonal in \mathcal{T}_{init} . We say e is a free-diagonal with respect to \mathcal{T}_{final} if flipping e creates a new diagonal \bar{e} that is in \mathcal{T}_{final} . When the context is clear, we simply refer to e as a free-diagonal.*

► **Lemma 3.** *If e_1 and e_2 are two free-diagonals in \mathcal{T}_{init} , then e_1 and e_2 are independent.*

Proof. Let \bar{e}_1 and \bar{e}_2 be the edges created by flipping e_1 and e_2 in \mathcal{T}_{init} , respectively. By Definition 2, both \bar{e}_1 and \bar{e}_2 are in \mathcal{T}_{final} . If e_1 and e_2 are neighbors, then \bar{e}_1 and \bar{e}_2 intersect each other, contradicting the fact that both \bar{e}_1 and \bar{e}_2 are in \mathcal{T}_{final} . ◀

The following lemma by Sleator, Tarjan and Thurston [27] shows that free-diagonals can be safely flipped and common diagonals will never be flipped in any shortest path.

► **Lemma 4** ([27]). *Let $(\mathcal{T}_{init}, \mathcal{T}_{final})$ be a pair of triangulations. (a) If \mathcal{T}_{init} contains a free-diagonal e , then there exists a shortest path from \mathcal{T}_{init} to \mathcal{T}_{final} where e is flipped first. (b) If \mathcal{T}_{init} and \mathcal{T}_{final} share a diagonal e in common, then every shortest path from \mathcal{T}_{init} to \mathcal{T}_{final} never flips e .*

In a sequence of flips, a previously non-free diagonal can become a free-diagonal only when one of its neighbors is flipped.

► **Lemma 5.** *Let $F = \langle f_1, \dots, f_r \rangle$ be a sequence of flips such that $\mathcal{T}_{init} = \mathcal{T}_0 \xrightarrow{F} \mathcal{T}_r = \mathcal{T}_{final}$ is a shortest path. Let e be a common diagonal of \mathcal{T}_{i-1} and \mathcal{T}_i , for $1 \leq i \leq r$. If e is not a free-diagonal in \mathcal{T}_{i-1} and is a free-diagonal in \mathcal{T}_i , then f_i^{\rightarrow} is a neighbor of e .*

Proof. Suppose that f_i^{\rightarrow} is not a neighbor of e . Then Q_e , the quadrilateral associated with e , remains the same in \mathcal{T}_i as in \mathcal{T}_{i-1} . Therefore, flipping e in \mathcal{T}_{i-1} creates the same diagonal as flipping e in \mathcal{T}_i , a contradiction to the fact that e is not a free-diagonal in \mathcal{T}_{i-1} and is a free-diagonal in \mathcal{T}_i . ◀

► **Definition 6.** *A pair of triangulations $(\mathcal{T}_{init}, \mathcal{T}_{final})$ of a convex polygon \mathcal{P} is called trivial if $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final}) = \phi(\mathcal{T}_{init}) - C(\mathcal{T}_{init}, \mathcal{T}_{final})$.*

► **Lemma 7.** *If a pair of triangulations $(\mathcal{T}_{init}, \mathcal{T}_{final})$ is trivial, then every flip in an optimal solution is a flip of a free-diagonal. Furthermore, it takes linear time to decide if a pair $(\mathcal{T}_{init}, \mathcal{T}_{final})$ is trivial.*

Proof. If $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final}) = \phi(\mathcal{T}_{init}) - C(\mathcal{T}_{init}, \mathcal{T}_{final})$, then every flip in an optimal solution F of $(\mathcal{T}_{init}, \mathcal{T}_{final})$ must create an additional common diagonal between \mathcal{T}_{init} and \mathcal{T}_{final} , which means that every flip in F is performed on a free-diagonal.

To decide if a pair $(\mathcal{T}_{init}, \mathcal{T}_{final})$ is trivial, first find all initial free-diagonals in \mathcal{T}_{init} and add them to a queue. Then flip the free-diagonals in the queue. By Lemma 5, new free-diagonals must be neighbors of previous flips and hence can be found and added to

the queue as the previous free-diagonals are flipped. The pair $(\mathcal{T}_{init}, \mathcal{T}_{final})$ is trivial if and only if there are always free-diagonals in the queue to be flipped until \mathcal{T}_{init} is transformed to \mathcal{T}_{final} . Since the flip distance between \mathcal{T}_{init} and \mathcal{T}_{final} is at most $2n - 4$, where n is the number of diagonals in \mathcal{T}_{init} and \mathcal{T}_{final} [26], this process takes linear time. \blacktriangleleft

Definition 8. Let $(\mathcal{T}_{init}, \mathcal{T}_{final})$ be a pair of triangulations. Let I be a set of diagonals in \mathcal{T}_{init} . We say I is a safe-set of diagonals with respect to \mathcal{T}_{final} , or simply safe-set, if

1. the diagonals in I are pair-wise independent, and
2. for any permutation $\pi(I)$ of I , there is a shortest path from \mathcal{T}_{init} to \mathcal{T}_{final} such that the diagonals in I are flipped first, in the same order as $\pi(I)$.

The set of diagonals corresponding to the source nodes of \mathcal{D}_F is a safe-set.

Lemma 9. Let $F = \langle f_1, \dots, f_r \rangle$ be a sequence of flips such that $\mathcal{T}_{init} = \mathcal{T}_0 \xrightarrow{F} \mathcal{T}_r = \mathcal{T}_{final}$ is a shortest path. Let $SC = \{f_{sc_1}, \dots, f_{sc_l}\}$ be the set of source nodes in \mathcal{D}_F . The set of diagonals $I = \{f_{sc_1}^{\leftarrow}, \dots, f_{sc_l}^{\leftarrow}\}$ is a safe-set in \mathcal{T}_{init} .

Proof. Let $f_{sc_i}, f_{sc_j} \in SC$, $i \neq j$, be two source nodes in \mathcal{D}_F . By Lemma 1, we may assume that f_{sc_i} is the first flip in F . If $f_{sc_i}^{\leftarrow}$ and $f_{sc_j}^{\leftarrow}$ share a triangle in \mathcal{T}_{init} , after flipping f_{sc_i} , $f_{sc_i}^{\rightarrow}$ and $f_{sc_j}^{\leftarrow}$ share a triangle in \mathcal{T}_1 , and hence by Lemma 1, there is a directed path from f_{sc_i} to f_{sc_j} in \mathcal{D}_F , contradicting to the fact that f_{sc_i} is a source in \mathcal{D}_F . Therefore, the diagonals in I are independent.

For an arbitrary permutation of $\pi(I)$, there is a topological sort of \mathcal{D}_F that begins with the flips in SC according to the order of $\pi(I)$. By Lemma 1, there is a shortest path between \mathcal{T}_{init} and \mathcal{T}_{final} that flips the diagonal in I first, in the order of $\pi(I)$. Therefore, I is a safe-set. \blacktriangleleft

2.2 Counting matchings in binary trees

The following lemma will be useful in the analysis of the running time of our algorithm.

Lemma 10. Let T_u be a binary tree rooted at a node u with n nodes and $n - 1$ edges. A matching in T_u is a subset of edges in T_u that do not share any endpoint (we consider an empty set to be a matching). Let \mathcal{E}_u be the set of matchings in T_u . Let \mathcal{E}_u^- be the set of matchings in T_u that exclude the edge(s) incident on u . Then $|\mathcal{E}_u^-| \leq F_n$ and $|\mathcal{E}_u| \leq F_{n+1}$, where F_n is the n -th Fibonacci number.

Proof. Let \mathcal{E}_u^+ be the set of matchings in T_u that include exactly one edge incident on u . Then $|\mathcal{E}_u| = |\mathcal{E}_u^+| + |\mathcal{E}_u^-|$. The lemma is proven by induction on n .

When $n = 1$, T_u is a leaf. Thus $\mathcal{E}_u^+ = \emptyset$ and $\mathcal{E}_u^- = \{\emptyset\}$. Therefore, $|\mathcal{E}_u^+| = 0 \leq F_0$, $|\mathcal{E}_u^-| = 1 \leq F_1$, and $|\mathcal{E}_u| = |\mathcal{E}_u^+| + |\mathcal{E}_u^-| = 1 \leq F_2$.

When $n = 2$, T_u has a single edge that connects u to a single leaf child v . Thus, $\mathcal{E}_u^+ = \{uv\}$ and $\mathcal{E}_u^- = \{\emptyset\}$. Therefore, $|\mathcal{E}_u^+| = 1 \leq F_1$, $|\mathcal{E}_u^-| = 1 \leq F_2$, and $|\mathcal{E}_u| = |\mathcal{E}_u^+| + |\mathcal{E}_u^-| = 2 \leq F_3$.

Now suppose that $n \geq 3$. We distinguish 2 cases.

- (a) u has only one child v . In this case, T_v , the subtree rooted at v , has $n - 1$ nodes and $n - 2$ edges. Every matching in \mathcal{E}_u^+ includes uv and hence excludes edges in T_v incident on v . Thus $|\mathcal{E}_u^+| = |\mathcal{E}_v^-| \leq F_{n-1}$ by the induction hypothesis. Every matching in \mathcal{E}_u^- excludes uv and hence is also a matching in T_v . Therefore, $|\mathcal{E}_u^-| = |\mathcal{E}_v| \leq F_{(n-1)+1} = F_n$ by the induction hypothesis. Finally, $|\mathcal{E}_u| = |\mathcal{E}_u^+| + |\mathcal{E}_u^-| \leq F_{n-1} + F_n = F_{n+1}$. The statement is true.
- (b) u has two children v, w . Let the number of nodes in T_v and T_w be n_1 and n_2 , respectively. Then $n = n_1 + n_2 + 1$. We consider \mathcal{E}_u^- and \mathcal{E}_u^+ separately:

- (i) The matchings in \mathcal{E}_u^- exclude both uv and uw . Therefore, $|\mathcal{E}_u^-| = |\mathcal{E}_v| \cdot |\mathcal{E}_w| \leq F_{n_1+1}F_{n_2+1} \leq F_{n_1+n_2+1} \leq F_n$ by the induction hypothesis and the Honsberger's Identity of Fibonacci numbers.
- (ii) The matchings in \mathcal{E}_u^+ include exactly one of the edges uv and uw . The number of matchings that include uv and exclude uw is $|\mathcal{E}_v^-| \cdot |\mathcal{E}_w| \leq F_{n_1}F_{n_2+1}$ by the induction hypothesis. The number of matchings that include uw and exclude uv is $|\mathcal{E}_v| \cdot |\mathcal{E}_w^-| \leq F_{n_1+1}F_{n_2}$ by the induction hypothesis. Therefore, $|\mathcal{E}_u^+| \leq F_{n_1}F_{n_2+1} + F_{n_1+1}F_{n_2}$. Finally, $|\mathcal{E}_u| = |\mathcal{E}_u^-| + |\mathcal{E}_u^+| \leq F_{n_1+1}F_{n_2+1} + F_{n_1}F_{n_2+1} + F_{n_1+1}F_{n_2} = F_{n_1+2}F_{n_2+1} + F_{n_1+1}F_{n_2} = F_{n_1+n_2+2} = F_{n+1}$, where the second last equality is the Honsberger's Identity of Fibonacci numbers.

This completes the proof. \blacktriangleleft

2.3 Parameterized complexity

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* (\mathcal{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 [10, 23] for more information about parameterized complexity.

3 The algorithm and its analysis

3.1 The basic ideas

Given an instance $(\mathcal{T}_{init}, \mathcal{T}_{final}, k)$ of CONVEX FLIP DISTANCE, our algorithm decides whether there is a sequence of flips $F = \langle f_1, \dots, f_r \rangle$, $r \leq k$ that transforms \mathcal{T}_{init} to \mathcal{T}_{final} .

By Lemma 4, common diagonals will never be flipped and free diagonals can be safely flipped. Thus, we can assume that $(\mathcal{T}_{init}, \mathcal{T}_{final})$ does not have any common diagonals, i.e., $C(\mathcal{T}_{init}, \mathcal{T}_{final}) = 0$, and that there are no free-diagonals in the initial triangulation \mathcal{T}_{init} . Therefore, we can assume $n \leq k \leq 2n - 4$, where $n = \phi(\mathcal{T}_{init})$.

In Preliminaries, we showed that we can represent a minimum solution F to an instance $(\mathcal{T}_{init}, \mathcal{T}_{final})$ using a DAG \mathcal{D}_F , and that any topological sort of \mathcal{D}_F is a minimum solution (Lemma 1). Therefore, to solve an instance of CONVEX FLIP DISTANCE, it suffices to compute a topological sort of \mathcal{D}_F .

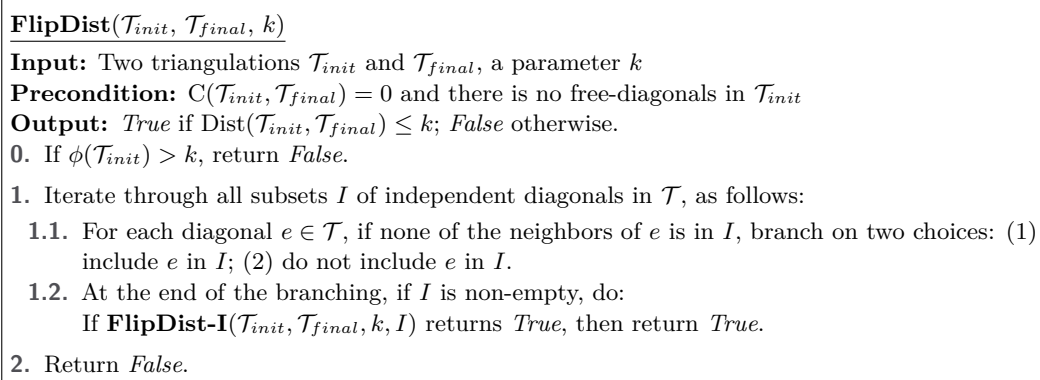
Intuitively, our algorithm computes a topological sort of \mathcal{D}_F by repeatedly finding and removing its source nodes, similar to Kahn's algorithm [15]. The algorithm uses a branch-and-bound approach to find and flip the source nodes in \mathcal{D}_F .

This seemingly simple approach faces two technical challenges. The design of our algorithm revolves around addressing them.

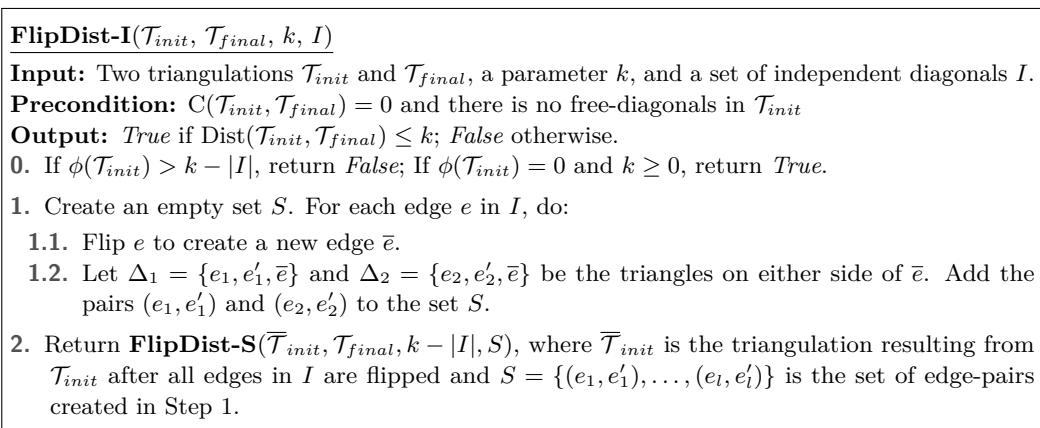
First, how to find the source nodes of the unknown \mathcal{D}_F without trying all possible subsets of the diagonals? The set I of initial source nodes of \mathcal{D}_F must be a subset of independent diagonals in \mathcal{T}_{init} . Our algorithm enumerates all subsets of independent diagonals in \mathcal{T}_{init} , whose number is at most F_{n+1} , the $(n+1)$ -th Fibonacci number (Lemma 14). Afterward, any new source node f_j of \mathcal{D}_F must be adjacent to a previous source node f_i . This means that the underlying diagonal of f_j is a neighbor of the new diagonal created by f_i . Therefore, when flipping a source node, our algorithm adds all neighbors of the new diagonal to a candidate pool S from which new source nodes are chosen. In fact, the neighbors of the new diagonal are added as *pairs* in S because at most one diagonal in each pair may be chosen as a new source node. The next set of new source nodes of \mathcal{D}_F will be chosen by branching on the edge-pairs in S . This method significantly reduces the search space of the source nodes.

Second, how to flip free-diagonals without increasing the branching factor of the algorithm? By Lemma 4, any free-diagonals can be safely flipped. If there is a free-diagonal e in \mathcal{T}_{init} , our algorithm flips it to create a new diagonal \bar{e} . Since \bar{e} is a common diagonal and hence will never be flipped again, the instance $(\mathcal{T}_{init}, \mathcal{T}_{final})$ is then partitioned along \bar{e} into two smaller instances. The candidate pool S is also partitioned accordingly and passed on to the two smaller instances. This method, through careful analysis, allows the free-diagonals to be flipped “for free” essentially.

3.2 The algorithm

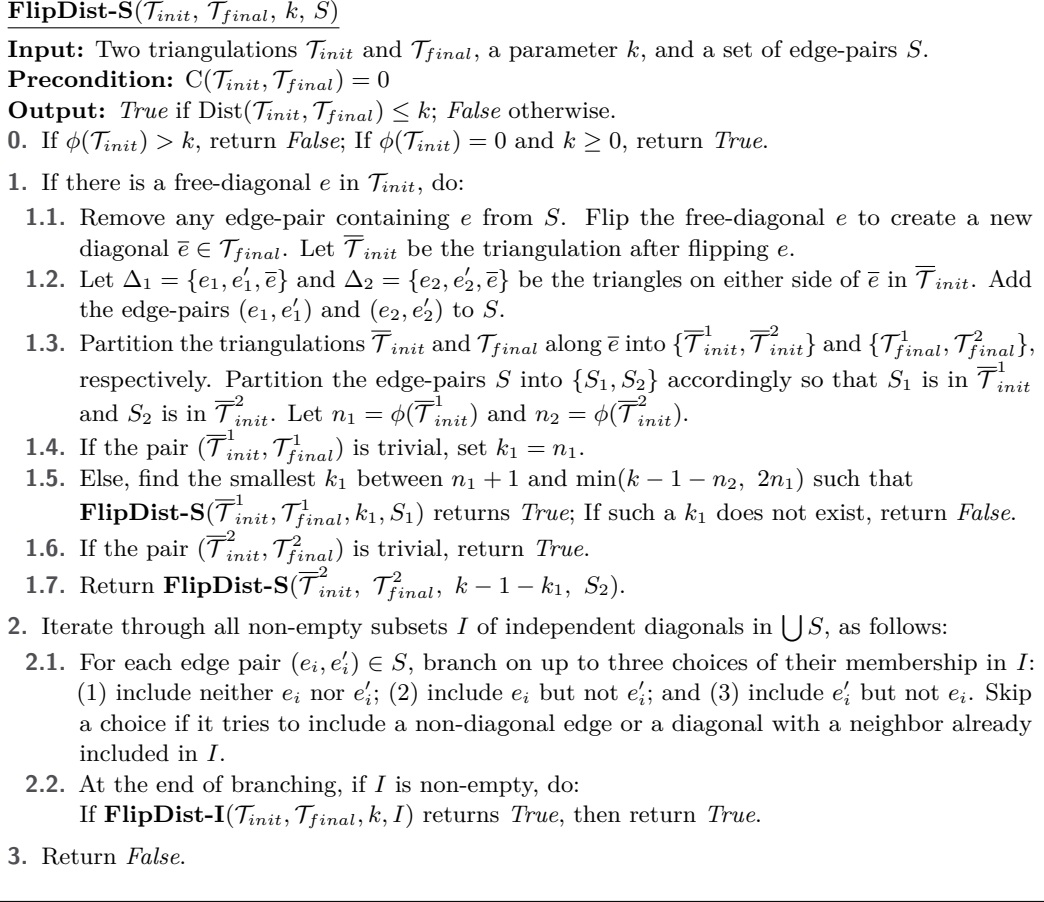


■ **Figure 1** The function **FlipDist**.



■ **Figure 2** The function **FlipDist-I**.

The algorithm’s main function **FlipDist** (Fig. 1) iterates through all subsets of independent diagonals in \mathcal{T} . For each non-empty subset I of independent diagonals that represents the set of initial source nodes of a DAG \mathcal{D}_F , two mutually recursive functions **FlipDist-I** (Fig. 2) and **FlipDist-S** (Fig. 3) are invoked to repeatedly remove the source nodes and find the set of new source nodes in \mathcal{D}_F . Along the way, whenever a free-diagonal is flipped to create a common diagonal, the instance is partitioned into smaller isolated instances.



■ **Figure 3** The function **FlipDist-S**.

Specifically, **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) performs flips on all diagonals in I . Suppose that a diagonal $e \in I$ is flipped to create a new edge \bar{e} . There are two triangles $\Delta_1 = \{e_1, e'_1, \bar{e}\}$ and $\Delta_2 = \{e_2, e'_2, \bar{e}\}$ on opposite sides of \bar{e} . The edges e_1, e'_1, e_2, e'_2 are candidates of the new source nodes in \mathcal{D}_F . Since (e_1, e'_1) are neighbors and so are (e_2, e'_2) , at most one edge can be chosen from each pair as a new source node. Therefore, the pairs (e_1, e'_1) and (e_2, e'_2) are added to a candidate pool S . After all diagonals in I are flipped, the current triangulations are $\bar{\mathcal{T}}_{init}$ and \mathcal{T}_{final} , the current parameter is $k - |I|$ since $|I|$ flips are already performed, and the candidate pool is S , all of which are passed to **FlipDist-S** as parameters.

FlipDist-S($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) first flips free-diagonals (if any) in \mathcal{T}_{init} . When a free-diagonal e is flipped, a common diagonal \bar{e} is created, and by Lemma 4, the instance can be safely partitioned along \bar{e} into two smaller isolated sub-instances, which can be solved recursively in a divide-and-conquer approach. In order to determine the parameters of the sub-instances, it is necessary to find the smallest parameter k_1 such that the first sub-instance returns *True*. The parameter of the second sub-instance is then set to be $k - k_1$. If there are no free-diagonals, **FlipDist-S** branches on the pairs of edges in S to form the safe-set I for the next round. For each edge-pair $(e_i, e'_i) \in S$ in S , I may include neither, only e_i , or only e'_i , forming a three-way branching.

3.3 Analysis of the algorithm

In the following, we will prove the correctness of the algorithm and analyze its running time. We start by proving the following invariant for both **FlipDist-I** and **FlipDist-S**:

▷ **Claim 11.** For every new diagonal created in **FlipDist-I** and **FlipDist-S**, its neighbors are contained in $\bigcup S$, which is the union of the edge-pairs in S .

Proof. In **FlipDist-I**, after each diagonal $e \in I$ is flipped, the neighbors of the new diagonal \bar{e} are included in the edge-pairs in S . Thus in **FlipDist-I**, $\bigcup S$ contains all neighbors of the new diagonals in $\bar{\mathcal{T}}_{init}$.

FlipDist-S($\mathcal{T}_{init}^i, \mathcal{T}_{final}^i, k, I$) starts by flipping free-diagonals (if any) in \mathcal{T}_{init} . If there is a free-diagonal e in \mathcal{T}_{init} , then by Lemma 4, e can be safely flipped to create a new diagonal $\bar{e} \in \mathcal{T}_{final}$. Before e is flipped to create a new diagonal \bar{e} , any pair (e, e') containing e is removed from S . Let $\bar{\mathcal{T}}_{init}$ be the triangulation after flipping e . Let $\Delta_1 = \{e_1, e'_1, \bar{e}\}$ and $\Delta_2 = \{e_2, e'_2, \bar{e}\}$ be the triangles on either side of \bar{e} in $\bar{\mathcal{T}}_{init}$. Two edge-pairs (e_1, e'_1) and (e_2, e'_2) are added to S . Note that if (e, e') is a pair in S before the free-diagonal e is flipped, then e' must be contained in one of two new pairs added to S after e is flipped. Therefore, after flipping e , $\bigcup S$ still contains all neighbors of the new diagonals in $\bar{\mathcal{T}}_{init}$, and hence the claim is true. ◁

► **Lemma 12.** Let $(\mathcal{T}_{init}, \mathcal{T}_{final})$ be a pair of triangulations that do not share any common edge. Let I be a set of independent diagonals in \mathcal{T}_{init} . Let S be a set of edge-pairs in \mathcal{T}_{init} and $\bigcup S$ be the union of the edge-pairs in S .

- (a) If there is a minimum solution F of $(\mathcal{T}_{init}, \mathcal{T}_{final})$ such that I is the set of underlying diagonals of the source nodes of \mathcal{D}_F , then **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) returns *True* if and only if F has at most k flips.
- (b) If there is a minimum solution F of $(\mathcal{T}_{init}, \mathcal{T}_{final})$ such that the set of underlying diagonals of the source nodes of \mathcal{D}_F is a subset of $\bigcup S$, then **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) returns *True* if and only if F has at most k flips.

Proof. The proof is by mutual induction on k . For the base case when $k = 0$, both functions will return true if and only if $\mathcal{T}_{init} = \mathcal{T}_{final}$. The statements are true.

Based on the inductive hypothesis, the inductive step is proven in two parts.

- (a) First consider **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$). By Lemma 9, I is a safe-set and hence for any permutation $\pi(I)$ of I , there is a shortest path from \mathcal{T}_{init} to \mathcal{T}_{final} such that the diagonals in I are flipped first according to the order of $\pi(I)$. Therefore, $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final}) \leq k$ if and only if $\text{Dist}(\bar{\mathcal{T}}_{init}, \mathcal{T}_{final}) \leq k - |I|$, where $\bar{\mathcal{T}}_{init}$ is the resulting triangulation after the diagonals in I are flipped. After the set of diagonals I corresponding to the source nodes of \mathcal{D}_F are flipped and removed from \mathcal{D}_F , the set of diagonals corresponds to the new source nodes in \mathcal{D}_F must be neighbors of the new diagonals in $\bar{\mathcal{T}}_{init}$ and hence by Claim 11 is a subset of $\bigcup S$. Therefore, when **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) calls **FlipDist-S**($\bar{\mathcal{T}}_{init}, \mathcal{T}_{final}, k - |I|, S$), by the inductive hypothesis, it returns *True* if and only if $\text{Dist}(\bar{\mathcal{T}}_{init}, \mathcal{T}_{final}) \leq k - |I|$, as required.
- (b) Now consider **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) in two cases:
 - (i) If \mathcal{T}_{init} has a free-diagonal e , then by Lemma 4, there is a shortest path from \mathcal{T}_{init} to \mathcal{T}_{final} such that e is flipped first to create a new diagonal \bar{e} that is shared by $\bar{\mathcal{T}}_{init}$ and \mathcal{T}_{final} . Again by Lemma 4, no shortest paths from $\bar{\mathcal{T}}_{init}$ to \mathcal{T}_{final} will flip \bar{e} . Therefore, the triangulations $\bar{\mathcal{T}}_{init}$ and \mathcal{T}_{final} can be safely partitioned along \bar{e} into $\{\bar{\mathcal{T}}_{init}^1, \bar{\mathcal{T}}_{init}^2\}$ and $\{\mathcal{T}_{final}^1, \mathcal{T}_{final}^2\}$, respectively. By Claim 11, after e is flipped, the set S still contains all neighbors of the new diagonals, which is partitioned

accordingly to S_1 and S_2 . There is a path of length at most k from \mathcal{T}_{init} to \mathcal{T}_{final} if and only if there is a path of length k_1 from \mathcal{T}_{init}^1 to \mathcal{T}_{final}^1 and a path of length k_2 from \mathcal{T}_{init}^2 to \mathcal{T}_{final}^2 such that $k_1 + k_2 = k - 1$. It is easy to see that S_1 and S_2 satisfy the condition of Part (b) for $\overline{\mathcal{T}}_{init}^1$ and $\overline{\mathcal{T}}_{init}^2$, respectively. By the inductive hypothesis, **FlipDist-S**($\overline{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1, k_1, S_1$) returns *True* if and only if there is a path of length k_1 from \mathcal{T}_{init}^1 to \mathcal{T}_{final}^1 , and **FlipDist-S**($\overline{\mathcal{T}}_{init}^2, \mathcal{T}_{final}^2, k - 1 - k_1, S_2$) returns *True* if and only if there is a path of length $k - 1 - k_1$ from $\overline{\mathcal{T}}_{init}^2$ to \mathcal{T}_{final}^2 . Therefore, **FlipDist-S**($\overline{\mathcal{T}}_{init}, \mathcal{T}_{final}, k - |I|, S$) returns *True* if and only if there is a path of length at most k from \mathcal{T}_{init} to \mathcal{T}_{final} , as required.

- (ii) If \mathcal{T}_{init} has no free-diagonals, then **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) enumerates all subsets I of independent diagonals and calls **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$). For each edge-pair $(e_1, e_2) \in S$, the algorithm branches on up to three possible choices: 1) include neither e_1 nor e_2 , 2) include e_1 but not e_2 , and 3) include e_2 but not e_1 . Since e_1 and e_2 are neighbors, they cannot both belong to I . **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) returns *True* if and only if there is an enumerated independent subset I of $\bigcup S$ such that **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) returns *True*. If $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final}) \leq k$, then by Claim 11, the set of the underlying diagonals of the source nodes of \mathcal{D}_F is a subset of $\bigcup S$ and hence, will be enumerated. Consequently, by Part (a) of the inductive step proven above, **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) returns *True*. Conversely, if there is an enumerated set I such that **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) returns *True*, then by Part (a) of the inductive step, there is a sequence of at most k flips that transforms \mathcal{T}_{init} to \mathcal{T}_{final} . Therefore, **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) returns *True* if and only if $\text{Dist}(\mathcal{T}_{init}, \mathcal{T}_{final}) \leq k$.

This completes the proof. \blacktriangleleft

The following lemma bounds the number of leaves in the search trees of **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) and **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$).

► **Lemma 13.** *Let $n = \phi(\mathcal{T}_{init})$. Let $L_I(n, k)$ be the number of leaves in the search tree of **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$). Let $L_S(n, k, s)$ be the number of leaves in the search tree of **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$), where $s = |S|$. Then $L_I(n, k) \leq 3^{2(k-n)}$ and $L_S(n, k, s) \leq 3^{s+2(k-n)}$.*

Proof. The proof is by mutual induction on k . For the base case when $k = 0$, we have $n = 0$ and both search trees have only 1 leaf. The statements are true.

Based on the inductive hypothesis, the inductive step is proven in two parts.

- (a) First consider **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$), which flips $|I|$ edges and create a set S of $2|I|$ edge-pairs. It then calls **FlipDist-S**($\overline{\mathcal{T}}_{init}, \mathcal{T}_{final}, k - |I|, S$) where $\phi(\overline{\mathcal{T}}_{init}) = \phi(\mathcal{T}_{init}) = n$. By the inductive hypothesis, the number of leaves in **FlipDist-S**($\overline{\mathcal{T}}_{init}, \mathcal{T}_{final}, k - |I|, S$) is at most $3^{|S|+2(k-|I|-n)} = 3^{2|I|+2(k-|I|-n)} = 3^{2(k-n)}$. Since there is no branching in **FlipDist-I**, we have $L_I(n, k) \leq 3^{2(k-n)}$ as required.
- (b) Now consider **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) in two cases:
- (i) If there is no free-diagonal in \mathcal{T}_{init} , then **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) branches into up to 3 subtrees for each pair in S , corresponding to three possible choices that include at most one edge in the pair. Therefore, at most $3^{|S|}$ subtrees are created after all edge-pairs in S are branched on and each subtree is a call to **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$). By Part (a) of the inductive step proven above, **FlipDist-I**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, I$) has at most $3^{2(k-n)}$ leaves. Therefore, the total number of leaves in the search tree of **FlipDist-S**($\mathcal{T}_{init}, \mathcal{T}_{final}, k, S$) is at most $3^{|S|} 3^{2(k-n)} = 3^{s+2(k-n)}$, as required.

(ii) Suppose that a free-diagonal e is flipped in Step 1 of **FlipDist-S** to create a new diagonal $\bar{e} \in \mathcal{T}_{final}$. The algorithm partitions the triangulations $\bar{\mathcal{T}}_{init}$ and \mathcal{T}_{final} along \bar{e} into $\{\bar{\mathcal{T}}_{init}^1, \bar{\mathcal{T}}_{init}^2\}$ and $\{\mathcal{T}_{final}^1, \mathcal{T}_{final}^2\}$, respectively. This creates two smaller instances $(\bar{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1)$ and $(\bar{\mathcal{T}}_{init}^2, \mathcal{T}_{final}^2)$. We may assume both are non-trivial because, by Lemma 7, trivial instances are solvable in linear time. Let $n_1 = \phi(\bar{\mathcal{T}}_{init}^1)$. The algorithm calls **FlipDist-S** $(\bar{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1, \hat{k}_1, S_1)$ for $\hat{k}_1 = n_1 + 1, \dots, \min(k-1-n_2, 2n_1)$, until $\hat{k}_1 = k_1$, where $k_1 = \text{Dist}(\bar{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1)$. If no such k_1 is found, then the search tree terminates and it is easy to verify that the statement is true. Since $\hat{k}_1 < k$, by the inductive hypothesis, **FlipDist-S** $(\bar{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1, \hat{k}_1, S_1)$ has at most $3^{|S_1|+2(\hat{k}_1-n_1)}$ leaves. This means for $\hat{k}_1 = n_1 + 1, \dots, k_1$, the subtrees of **FlipDist-S** $(\bar{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1, \hat{k}_i, S_1)$ has at most $3^{|S_1|+2}, 3^{|S_1|+4}, \dots, 3^{|S_1|+2(k_1-n_1)}$ leaves, respectively. Observing that this is a geometric sequence with a common ratio of 9, their sum is at most $\frac{9}{8} \cdot 3^{|S_1|+2(k_1-n_1)}$.

Finally, the algorithm calls **FlipDist-S** $(\bar{\mathcal{T}}_{init}^2, \mathcal{T}_{final}^2, k-1-k_1, S_2)$, which by the inductive hypothesis has at most $3^{|S_2|+2(k-1-k_1-n_2)}$ leaves.

Thus, in total, the number of leaves in the search tree of **FlipDist-S** $(\bar{\mathcal{T}}_{init}, \mathcal{T}_{final}, k, S)$ is at most $\frac{9}{8} \cdot 3^{|S_1|+2(k_1-n_1)} + 3^{|S_2|+2(k-1-k_1-n_2)}$. Let $x = 3^{|S_1|+2(k_1-n_1)}$ and $y = 3^{|S_2|+2(k-1-k_1-n_2)}$. Since both instances $(\bar{\mathcal{T}}_{init}^1, \mathcal{T}_{final}^1)$ and $(\bar{\mathcal{T}}_{init}^2, \mathcal{T}_{final}^2)$ are non-trivial, we have $k_1 - n_1 \geq 1$ and $k - 1 - k_1 - n_2 \geq 1$, and hence $x, y \geq 9$. Therefore, the number of leaves in the search tree of **FlipDist-S** $(\bar{\mathcal{T}}_{init}, \mathcal{T}_{final}, k, S)$ is at most

$$\frac{9}{8} \cdot 3^{|S_1|+2(k_1-n_1)} + 3^{|S_2|+2(k-1-k_1-n_2)} = \frac{9}{8}x + y \quad (1)$$

$$= \left(\frac{27}{8y} + \frac{3}{x}\right) \cdot \frac{xy}{3} \quad (2)$$

$$\leq \left(\frac{27}{8 \cdot 9} + \frac{3}{9}\right) \cdot \frac{xy}{3} \quad (3)$$

$$\leq \frac{xy}{3} \quad (4)$$

$$= \frac{3^{|S_1|+2(k_1-n_1)} \cdot 3^{|S_2|+2(k-1-k_1-n_2)}}{3} \quad (5)$$

$$= 3^{|S_1|+|S_2|-1+2(k-1-n_1-n_2)}. \quad (6)$$

We have $|S_1| + |S_2| \leq |S| + 1$ because when a free-diagonal e is flipped, at least one edge-pair in S that contains e is removed and two new edge-pairs are added to S . We also have $n_1 + n_2 = n - 1$ because after the $\bar{\mathcal{T}}_{init}$ is partitioned along the diagonal \bar{e} , the total number of diagonals is reduced by 1.

Therefore, the total number of leaves in the search tree of **FlipDist-S** $(\bar{\mathcal{T}}_{init}, \mathcal{T}_{final}, k, S)$ is $L_S(n, k, s) \leq 3^{|S_1|+|S_2|-1+2(k-1-n_1-n_2)} \leq 3^{|S|}3^{2(k-n)} = 3^{s+2(k-n)}$, as required.

This completes the proof. \blacktriangleleft

When the algorithm initially starts, the set of source nodes may be any subsets I of independent diagonals in \mathcal{T}_{init} . We will prove in Lemma 14 that there are at most F_{n+1} such subsets, where F_n is the n -th Fibonacci number, and they can be enumerated using polynomial space and in time $\mathcal{O}(n)$ each.

44:12 An $\mathcal{O}(3.82^k)$ Time \mathcal{FPT} Algorithm for Convex Flip Distance

► **Lemma 14.** *Let \mathcal{T} be a triangulation of a convex polygon and $\phi(\mathcal{T}) = n$. The number of subsets of independent diagonals in \mathcal{T} is at most F_{n+1} , the $(n+1)$ -th Fibonacci number. Furthermore, all such subsets can be iterated in time $\mathcal{O}(n)$ each using polynomial space.*

Proof. First, observe that the set of subsets of independent diagonals in \mathcal{T} is in bijection with the set of matchings in the binary tree T corresponding to \mathcal{T} . Each triangle in \mathcal{T} corresponds to a node in T and each diagonal shared by two triangles in \mathcal{T} corresponds to an edge between the two nodes in T representing the two triangles. A set of independent diagonals in \mathcal{T} corresponds to a subset of edges in T that do not share any endpoint, referred to as a *matching* (we consider an empty set to be a matching). By Lemma 10, the number of matchings in T is at most F_{n+1} and hence the number of subsets of independent diagonals in \mathcal{T} is at most F_{n+1} .

To iterate all such subsets I , consider all diagonals in an arbitrarily fixed order. For each diagonal e , if any of its neighbors has already been included in I , do not include e ; otherwise, branch on e to include or exclude e in I . When all diagonals have been branched on, I is a subset of independent diagonals in \mathcal{T} . Since each leaf of this branching tree corresponds to a unique subset of independent diagonals in \mathcal{T} and the depth of the branching tree is n , the running time is $\mathcal{O}(n)$ for each subset. The iteration uses polynomial space because only a subset of diagonals in \mathcal{T} is maintained at each step of the branching. ◀

Finally, we have the correctness and complexity of our algorithm.

► **Theorem 15.** *The algorithm $\mathbf{FlipDist}(\mathcal{T}_{init}, \mathcal{T}_{final}, k)$ runs in time $\mathcal{O}(3.82^k)$ using polynomial space and returns *True* if and only if there is a sequence of at most k flips that transforms \mathcal{T}_{init} to \mathcal{T}_{final} .*

Proof. By Lemma 14, $\mathbf{FlipDist}(\mathcal{T}_{init}, \mathcal{T}_{final}, k)$ branches into at most F_{n+1} subsets of independent diagonals. For each such subset I , the function $\mathbf{FlipDist-I}(\mathcal{T}_{init}, \mathcal{T}_{final}, k, I)$ is called, which has at most $3^{2(k-n)}$ leaves by Lemma 13. Therefore, the search tree of $\mathbf{FlipDist}(\mathcal{T}_{init}, \mathcal{T}_{final}, k)$ has at most $F_{n+1}3^{2(k-n)}$ leaves, which means there are at most $F_{n+1}3^{2(k-n)}$ root to leaf paths in the search tree.

Along each root-to-leaf path, the algorithm does the following: (a) enumerates the initial independent set I in time $\mathcal{O}(n)$ when charged to each path; (b) performs at most k flips that take time $\mathcal{O}(1)$ each; (c) performs at most n partitions that take time $\mathcal{O}(n)$ each; (d) perform at most k rounds of branching, where each round takes time $\mathcal{O}(k)$ when charged to each end-branch. Therefore, the time spent on each root-to-leaf path is $\mathcal{O}(k^2 + n^2) = \mathcal{O}(n^2)$. Since $\mathcal{O}(n^2 F_{n+1}) = \mathcal{O}(1.618^n)$, the overall running time is $\mathcal{O}(n^2 F_{n+1} 3^{2(k-n)}) = \mathcal{O}(1.618^n 3^{2(k-n)}) = \mathcal{O}(9^k \cdot (\frac{1.618}{9})^n)$.

Since $k/2 \leq n \leq k$, the overall running time $\mathcal{O}(9^k \cdot (\frac{1.618}{9})^n)$ is maximized when $n = k/2$. Therefore, the total running time of the algorithm is $\mathcal{O}(9^k \cdot (\frac{1.618}{9})^n) = \mathcal{O}(9^k \cdot (\frac{1.618}{9})^{k/2}) = \mathcal{O}(3.82^k)$.

$\mathbf{FlipDist}(\mathcal{T}_{init}, \mathcal{T}_{final}, k)$ enumerates all subsets I of independent diagonals as the initial set of source nodes and calls $\mathbf{FlipDist-I}(\mathcal{T}_{init}, \mathcal{T}_{final}, k, I)$. By Lemma 12, $\mathbf{FlipDist-I}(\mathcal{T}_{init}, \mathcal{T}_{final}, k, I)$ returns *True* if and only if there is a sequence of at most k flips that transforms \mathcal{T}_{init} to \mathcal{T}_{final} .

Finally, the algorithm uses polynomial space because every step of it, including the iteration of the initial independent sets (Lemma 14), uses polynomial space.

This proves the correctness and complexity of our algorithm. ◀

4 Concluding remarks

Both CONVEX FLIP DISTANCE and GENERAL FLIP DISTANCE are important problems. The current paper presents a simple \mathcal{FPT} algorithm for CONVEX FLIP DISTANCE that runs in time $\mathcal{O}(3.82^k)$ and uses polynomial space, significantly improving the previous best \mathcal{FPT} algorithm the problem, which runs in time $\mathcal{O}(n+k \cdot 32^k)$ and is the same \mathcal{FPT} algorithm for GENERAL FLIP DISTANCE [11].

Our algorithm takes advantage of the structural properties of CONVEX FLIP DISTANCE regarding the common diagonals and the free-diagonals. However, the general approach of our algorithm, namely finding a topological sort of the DAG \mathcal{D}_F by repeatedly removing the source nodes, seems applicable to GENERAL FLIP DISTANCE. It remains to be seen if can be used to derive an improved algorithm for GENERAL FLIP DISTANCE.

The recent progress on both CONVEX FLIP DISTANCE and GENERAL FLIP DISTANCE, including [16, 11] and this work, all rely on the DAG that models dependency relation among the flips. More research along this line will likely produce further improved algorithms. However, deciding whether the CONVEX FLIP DISTANCE problem is \mathcal{NP} -hard remains a challenging open problem and may require new insights into the structural properties of the problem.

References

- 1 O. Aichholzer, F. Hurtado, and M. Noy. A lower bound on the number of triangulations of planar point sets. *Computational Geometry: Theory and Applications*, 29(2):135–145, 2004.
- 2 O. Aichholzer, W. Mulzer, and A. Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015.
- 3 A. Bonnin and J.-M. Pallo. A shortest path metric on unlabeled binary trees. *Pattern Recognition Letters*, 13(6):411–415, 1992.
- 4 P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry: Theory and Applications*, 42(1):60–80, 2009.
- 5 M. B. Calvo and S. Kelk. An improved kernel for the flip distance problem on simple convex polygons. In Meng He and Don Sheehy, editors, *Proceedings of the 33rd Canadian Conference on Computational Geometry, CCCG 2021*, pages 195–199, 2021.
- 6 Y.-J. Chen, J.-M. Chang, and Y.-L. Wang. An efficient algorithm for estimating rotation distance between two binary trees. *International Journal of Computer Mathematics*, 82:1095–1106, 2005.
- 7 S. Cleary and K. St. John. Rotation distance is fixed-parameter tractable. *Information Processing Letters*, 109(16):918–922, 2009.
- 8 S. Cleary and K. St. John. A linear-time approximation algorithm for rotation distance. *J. Graph Algorithms Appl.*, 14:385–390, 2010.
- 9 K. Culik and D. Wood. A note on some tree similarity measures. *Information Processing Letters*, 15(1):39–42, 1982.
- 10 R. Downey and M. Fellows. *Parameterized Complexity*. Springer, New York, 1999.
- 11 Q. Feng, S. Li, X. Meng, and J. Wang. An improved fpt algorithm for the flip distance problem. *Information and Computation*, 281, 2021.
- 12 S. Fordham and S. Cleary. Minimal length elements of thompson’s groups $f(p)$. *Geometriae Dedicata*, 141:163–180, 2007.
- 13 S. Hanke, T. Ottmann, and S. Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.
- 14 F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

- 15 A. B. Kahn. Topological sorting of large networks. *Commun. ACM*, 5(11):558–562, November 1962.
- 16 I. Kanj, E. Sedgwick, and G. Xia. Computing the flip distance between triangulations. *Discret. Comput. Geom.*, 58(2):313–344, 2017.
- 17 I. Kanj and G. Xia. Flip Distance Is in FPT Time $\mathcal{O}(n + k \cdot c^k)$. In *proceedings of STACS*, volume 30 of *LIPIcs*, pages 500–512, 2015.
- 18 C. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972.
- 19 M. Li and L. Zhang. Better approximation of diagonal-flip transformation and rotation transformation. In *Proceedings of the 4th Annual International Conference on Computing and Combinatorics*, COCOON '98, pages 85–94, 1998.
- 20 A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry: Theory and Applications*, 49:17–23, 2015.
- 21 J. Lucas. An improved kernel size for rotation distance in binary trees. *Information Processing Letters*, 110(12):481–484, 2010.
- 22 F. Luccio and L. Pagli. On the upper bound on the rotation distance of binary trees. *Information Processing Letters*, 31(2):57–60, 1989.
- 23 R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, USA, 2006.
- 24 J. Pallo. On the rotation distance in the lattice of binary trees. *Information Processing Letters*, 25(6):369–373, 1987.
- 25 A. Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry: Theory and Applications*, 47(5):589–604, 2014.
- 26 L. Pournin. The diameter of associahedra. *Advances in Mathematics*, 259:13–42, 2014.
- 27 D. Sleator, R. Tarjan, and W. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1:647–681, 1988.
- 28 R. P. Stanley. *Enumerative Combinatorics*, volume 2. Cambridge University Press, 1999.