An Improved Kernel for the Flip Distance Problem on Simple Convex Polygons[☆]

Miguel Bosch Calvo[a,b], Steven Kelk[b]

[a] *IDSIA, USI-SUPSI, Switzerland.*
[b] *Department of Data Science and Engineering, Maastricht University, The Netherlands.*

Abstract

The complexity of computing the flip distance between two triangulations of a simple convex polygon is unknown. Here we approach the problem from a parameterized complexity perspective and improve upon the $2k$ kernel of Lucas [10]. Specifically, we describe a kernel of size $\frac{4k}{3}$ and then show how it can be improved to $(1+\epsilon)k$ for every constant $\epsilon > 0$. By ensuring that the kernel consists of a single instance our result yields a kernel of the same magnitude (up to additive terms) for the almost equivalent rotation distance problem on rooted, ordered binary trees. The earlier work of Lucas left the kernel as a disjoint set of instances, potentially allowing very minor differences in the definition of the size of instances to accumulate, causing a constant-factor distortion in the kernel size when switching between flip distance and rotation distance formulations. Our approach avoids this sensitivity. We have also undertaken experiments to understand how much reduction is achieved by our kernel in practice.

Keywords: Parameterized Complexity, Kernel, Triangulation, Simple Convex polygons

1. Introduction

Triangulating a set of points on a plane is a common operation in computational geometry. The operation of *flipping a diagonal* is defined as removing one edge of a triangulation, creating a convex quadrangle, and then adding to the triangulation the opposing diagonal of that quadrangle, as seen in Figure 1.

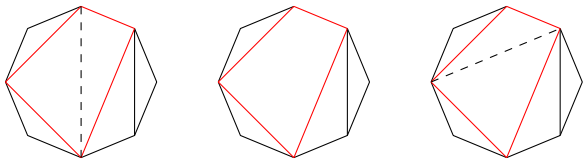

Figure 1: Flipping a diagonal of a triangulation of a simple convex polygon.

The *flip distance* between two triangulations of the same set of points on a plane is the minimum number of flips needed to transform one triangulation into another. Computing flip distance is NP-hard, even for the case of simple polygons [1].

In this article we will be working in a more restricted setup by considering only triangulations of simple *convex* polygons. The complexity of the problem is unknown. Indeed, there is a well known correspondence - essentially, an equivalence - between this problem and the computation of rotation distance between two rooted, ordered binary trees. It has been an open question for several decades whether rotation distance is polynomial-time solveable. Some of the results in this area have been obtained using the rotation distance formulation, but most of the work has been undertaken in the flip distance formulation.

Here we adopt a parameterized complexity perspective; in particular, a kernelization perspective [6]. Cleary et al. [3] proved that the problem is fixed parameter tractable, by providing a $5k$ kernel, where the parameter k is the flip distance. Lucas [10] employed different reduction strategies to obtain a kernel of size $2k$. In this article we will show how to improve upon the kernelization result of Lucas. We describe a $\frac{4k}{3}$ kernel and then extend

[☆] A preliminary version of this article, without experimental section, appeared in the proceedings of CCCG2021.

the approach to yield a $(1+\epsilon)k$ kernel for every constant $\epsilon > 0$; the running time grows sharply in $1/\epsilon$ but remains polynomial for fixed ϵ. Our article extends the decomposition-based approach of Lucas in two ways. We strengthen the bound on the size of the kernel, and potentially lower the flip distance, by solving small decomposed instances to optimality. Secondly, we show how to "reverse" the decomposition strategy adopted by Lucas, thus merging the separate instances into a single reduced instance at the end. This merging step ensures that the size of the kernel remains (up to additive terms) unchanged whether we view the problem from the flip distance or rotation distance perspective. As we note in the Discussion section, this is not as straightforward for Lucas' kernel result: there a subtle constant-factor distortion occurs when switching from one formulation to the other.

Finally, we conduct a number of experiments to understand the way our kernelization algorithm performs on synthetic data. We find that the reduced instances are substantially smaller than the worst-case theoretical bound, and that the $\frac{4k}{3}$ kernel achieves significantly more reduction than Lucas' kernel without requiring any extra investment in running time.

2. Preliminaries

We are working here with simple convex polygons. Such a polygon can be viewed without loss of generality as a simple cycle on n edges and n vertices. A triangulation of a simple convex polygon on n edges contains exactly $n-3$ diagonals. Hence a triangulation of a simple convex polygon can be represented as a list of $n-3$ edges and two triangulations are considered equal if the $n-3$ edges (i.e. the diagonals) are identical.

Thus, there is a finite number of triangulations of simple convex polygons of a given size. Precisely, the number of triangulations of a simple convex polygon of size n is given by the $(n-2)$th Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$.

We define $\mathcal{P}_n$ as the set containing all triangulations of simple convex polygons of size n. Thus $|\mathcal{P}_n| = C_{n-2}$. We abuse notation slightly and say that $(P, P') \in \mathcal{P}_n$ if both P and P' are triangulations of simple convex polygons of size n.

Henceforth, for the sake of brevity, we will refer to triangulations of simple convex polygons as triangulations or simply polygons.

Given $(P, P') \in \mathcal{P}_n$, we call a *shortest path* from P to P' to the sequence of polygons $P = P_0, P_1, P_2, \ldots, P_m = P'$ such that we can transform P_i into P_{i+1} by just flipping one diagonal and m is the minimum among all possible sequences. Given a pair of polygons $(P, P') \in \mathcal{P}_n$, the *flip distance* $d(P, P')$ between P and P' is the length of a shortest path from P to P'.

One of the earliest results in this area is the upper bound on flip distance proved by Culik et al. [4]. Precisely, the flip distance between two polygons $(P, P') \in \mathcal{P}_n$ is at most $2n-6$ for all $(P, P') \in \mathcal{P}_n$. Later, Sleator et al. [12] improved the bound to $2n-10$ for all $(P, P') \in \mathcal{P}_n, n > 12$, and by making use of hyperbolic geometry proved that the bound is tight.

Also, since every flip of a diagonal only affects one diagonal, the flip distance between (P, P') is at least the number of non-common diagonals of (P, P') [10].

There is another result from Sleator et al. [12] that is of importance to us. It implies that common diagonals belong to every polygon of every shortest path, and therefore that they should not be flipped at any point: Given $(P, P') \in \mathcal{P}_n$, if there is a common diagonal between P and P', then every shortest path from P to P' does not flip that diagonal.

We now present a formal definition of the Parameterized Flip Distance problem, which is the problem we will be addressing in this article:

Parameterized Flip Distance
Input: A pair of polygons $(P, P') \in \mathcal{P}_n$ and a parameter $k \in \mathbb{N}$.
Question: Is the flip distance between P and P' at most k?

As is standard in the study of kernelization, we will apply polynomial-time reduction rules to yield instances whose size is bounded by a function purely of k. We omit a formal definition of kernelization, referring to standard texts such as [6] for more details. We emphasize that the size of an instance, n, refers to the number of outer edges in the polygons.

The kernel we propose uses some of the ideas presented by Lucas at [10] combined with new reduction rules to tighten the bound on the kernel, plus a new merging step. Lucas' idea is based on dividing the original pair of polygons along their common diagonals by using the results by Sleator et al. [12]. An example of such division is shown in Figure 2.

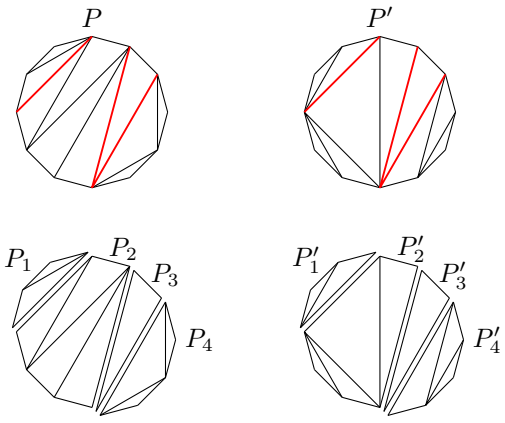

Figure 2: An example of splitting a polygon pair (P, P') along its common diagonals into m disjoint pairs. Here the instances (P, P') have size 12, so $(P, P') \in \mathcal{P}_{12}$, and they have 3 common diagonals, so they are divided into $m = 4$ disjoint pairs $(P_1, P'_1), (P_2, P'_2), (P_3, P'_3), (P_4, P'_4)$.

We will first present the operations that allows us to obtain a $\frac{4k}{3}$ kernel and then we will extend those ideas to derive the $(1+\epsilon)k$ kernel.

3. Results

3.1. $\frac{4k}{3}$ kernel

Lucas [10], using the results of Sleator et al. [12], showed that given two polygons $(P, P') \in \mathcal{P}_n$ with $m-1$ common diagonals, we can create m disjoint pairs of polygons $(P_i, P'_i), i \in [1, m]$ by dividing the original polygons along their common diagonals, so each common diagonal becomes an outer edge of one of the instances (P_i, P'_i) and each pair does not have any common diagonal. Thus we derive the following lemma:

Lemma 1. *The flip distance of $(P, P') \in \mathcal{P}_n$ is equal to the sum of the distances between all m pairs (P_i, P'_i) resulting from the division of (P, P') along its $m-1$ common diagonals, i.e. $d(P, P') = \sum_{i=1}^{m} d(P_i, P'_i)$.*

It is useful to apply the division along common diagonals into m pairs to the parameterized version of the problem, given by $(P, P') \in \mathcal{P}_n, k \in \mathbb{N}$.

Given a set of m pairs of polygons $(P_i, P'_i) \in \mathcal{P}_{n_i}$, let d_i be the number of diagonals of instance i, so that $d_i = n_i - 3$.

The upper bound of the problem of roughly $2n$ can be applied to every pair, and the pairs do not have any common diagonal, so we can deduce $d_i \leq d(P_i, P'_i) \leq 2d_i$. Since $d(P, P') = \sum d(P_i, P'_i)$ we can output a trivial YES answer if $\sum d_i \leq k/2$, and a trivial NO if $\sum d_i > k$, so for all non-trivial instances we have $k/2 < \sum_{i=1}^{m} d_i \leq k$.

Now we present a trivial observation and a lemma that will be useful to prove our final result.

Observation 1. *A pair of quadrilaterals with no common diagonals have distance 1. Similarly, a pair of pentagons with no common diagonals have distance 2.*

Lemma 2. *Given a set of m pairs of polygons $(P_i, P'_i) \in \mathcal{P}_{n_i}$, we can build a pair of polygons $(P, P') \in \mathcal{P}_n$ with $n = \sum_{i=1}^{m} d_i + m + 2$, and $d(P, P') = \sum_{i=1}^{m} d(P_i, P'_i)$.*

Proof. Given two pairs of polygons $(P_1, P'_1), (P_2, P'_2)$ with sets of outer edges $\{e^1_1, e^1_2, \ldots, e^1_{n_1}\}$ and $\{e^2_1, e^2_2, \ldots, e^2_{n_2}\}$ we can create a pair of polygons with edges:

$$\{e^1_1, e^1_2, \ldots, e^1_{n_1-1}, e^2_1, e^2_2, \ldots, e^2_{n_2-1}\}$$

We add to that polygon all diagonals present in both (P_1, P'_1) and (P_2, P'_2), plus a diagonal δ_s in place of $e^1_{n_1}$ and $e^2_{n_2}$. It is clear that we can add those diagonals and they will be non-crossing. We can see an example of this operation in Figure 3.

This way we have a new pair of polygons (P, P') in which the edges $\{e^1_1, e^1_2, \ldots, e^1_{n_1-1}, \delta_s\}$ induce the polygons (P_1, P'_1) and $\{e^2_1, e^2_2, \ldots, e^2_{n_2-1}, \delta_s\}$ the polygons (P_2, P'_2). Since δ_s is a common diagonal and thus is never flipped in a shortest path, $d(P, P') = d(P_1, P'_1) + d(P_2, P'_2)$. The size of (P, P') is $n_1 + n_2 - 2 = d_1 + d_2 + 4$, and it has $d_1 + d_2 + 1$ diagonals, of which at least one of them is common. Repeated applications of this operation complete the proof.

Theorem 3. *There is a kernel of size $\frac{4k}{3} + \mathcal{O}(1)$ for the* Parameterized Flip Distance *problem. Specifically, given a pair of polygons $(P, P') \in \mathcal{P}_n$ and a parameter k we can output in polynomial time another pair of polygons $(P^*, P^{*\prime})$ of size at most $\frac{4k}{3} + 2$, and a parameter $k' \leq k$ such that:*

$$d(P, P') \leq k \iff d(P^*, P^{*\prime}) \leq k'$$

Proof. Given $(P, P') \in \mathcal{P}_n$, and a parameter k, the following algorithm outputs a kernel of the problem of size at most $\frac{4k}{3} + 2$.

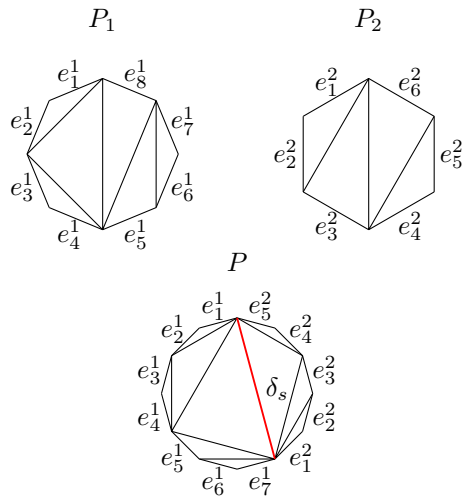

Figure 3: Given two pairs of polygons (P_1, P_1') and (P_2, P_2') we can generate a new pair (P, P') that has distance equal to the sum of the distances of the original pair. In this example (P_1, P_1') has 8 outer edges $\{e_1^1, e_2^1, e_3^1, e_4^1, e_5^1, e_6^1, e_7^1, e_8^1\}$ and (P_2, P_2') has 6 outer edges $\{e_1^2, e_2^2, e_3^2, e_4^2, e_5^2, e_6^2\}$, resulting in a pair (P, P') with 12 edges $\{e_1^1, e_2^1, e_3^1, e_4^1, e_5^1, e_6^1, e_7^1, e_1^2, e_2^2, e_3^2, e_4^2, e_5^2\}$ (in this figure only one of the polygons of the pair is shown since we operate identically with the other).

1. Divide (P, P') along their common diagonals to obtain m pairs of polygons (P_i, P_i') and discard all pairs that have only three edges, because their distance is 0. Now we have m' pairs, with $m' \leq m$, so we re-number the pairs to have $(P_i, P_i'), i \in [1, m']$.
2. Making use of Observation 1, discard all pairs with four edges and reduce the parameter k by one per each pair removed that way. Proceed the same way with pairs of five edges reducing the parameter by two instead and renumber the pairs as we did in the previous step. We get a new parameter $k' \leq k$.
3. If $\sum d_i > k'$ output NO. If $\sum d_i \leq k'/2$ output YES.
4. Use Lemma 2 to create a new polygon $(P^*, P^{*\prime})$ from all the remaining pairs. The new instance is defined by $(P^*, P^{*\prime})$ and k'.

Since we have removed all pairs with $d_i < 3$, each pair has at least 3 diagonals, none of them common, so m is at most $k'/3$, or otherwise $\sum d_i > k'$, and we could have output a trivial NO answer. Also, we have that $k'/2 < \sum_{i=1}^{m} d_i \leq k'$, and by making use of Lemma 2 to obtain the pair of polygons $(P^*, P^{*\prime})$, they will be of size $\sum_{i=1}^{m} d_i + m + 2 \leq k' + k'/3 + 2 \leq 4k/3 + 2$.

Also, from Lemma 1, Lemma 2 and Observation 1 it is clear that $d(P, P') \leq k$ if and only if $d(P^*, P^{*\prime}) \leq k'$, and Lucas [9] showed that the first step can be done in $\mathcal{O}(n^2)$ time, while the last step can be done in time $\mathcal{O}(n)$, completing the proof.

3.2. $(1+\epsilon)k$*-kernel*

In this section we will show a procedure that allows us to obtain a kernel of size $(1+\epsilon)k + \mathcal{O}(1)$ in time that is polynomial in n when $\epsilon > 0$ is constant. The procedure is based on the algorithm of the previous section and on the trivial exponential-time algorithm that allows us to solve an instance of size n in time $\mathcal{O}(n^{2n})$ by trying all possible diagonal flips recursively.

Theorem 4. *Given a pair of polygons $(P, P') \in \mathcal{P}_n$ and a parameter k, we can output another pair of polygons $(P^*, P^{*\prime})$ of size at most $(1+\epsilon)k + 2$ and a parameter $k' \leq k$ such that:*

$$d(P, P') \leq k \iff d(P^*, P^{*\prime}) \leq k'$$

In time $\mathcal{O}(n^2 + f(\epsilon)n)$, where $f(\epsilon)$ is a function that only depends on ϵ.

Proof. Given two polygons $(P, P') \in \mathcal{P}_n$, a parameter $k \in \mathbb{N}$ and some $\epsilon > 0$, apply steps 1 and 2 of the algorithm described in Theorem 3. Then solve all instances of size less than $1/\epsilon + 3$, i.e. instances that have fewer than $1/\epsilon$ diagonals, using the trivial exponential-time algorithm. We can do this in time $\mathcal{O}((1/\epsilon)^{2/\epsilon} \cdot n)$, (because there can be at most $n-2$ instances after splitting common diagonals) and discard all pairs solved this way, reducing the parameter k by the sum of the flip distances of the pairs solved this way. Finally, apply steps 3 and 4 of the algorithm.

Now each remaining pair before step 3 will have at least $1/\epsilon$ diagonals, so m must be at most $\epsilon k'$, by a similar reasoning as in Theorem 3. Then we have that the polygons $(P^*, P^{*\prime})$ will have size at most $\sum_{i=1}^{m} d_i + m + 2 \leq k' + \epsilon k' + 2 \leq (1+\epsilon)k + 2$.

Since the steps common with the $\frac{4k}{3}$ kernelization algorithm can be done in time $\mathcal{O}(n^2)$, and the additional time spent on solving small instances is at most $\mathcal{O}((1/\epsilon)^{2/\epsilon} \cdot n)$, the total time required to produce the kernel is $\mathcal{O}(n^2 + (1/\epsilon)^{2/\epsilon} \cdot n) = \mathcal{O}(n^2 + f(\epsilon)n)$, as we wanted to prove.

We note that by using the recent FPT algorithm of [5] instead of the trivial exponential-time algorithm, we can solve an instance of size less than $1/\epsilon$

in time $\mathcal{O}(32^{1/\epsilon} \cdot \text{poly}(1/\epsilon))$ instead of $\mathcal{O}((1/\epsilon)^{2/\epsilon})$, which is a significant improvement.

As mentioned earlier there is a near equivalence between the flip distance problem on simple convex polygons, and the rotation distance problem on two ordered, rooted binary trees. The definition of the rotation distance problem is rather technical so we omit details. In any case, it is well-known that an instance of rotation distance of size n (where the size here denotes the number of non-leaf nodes in one of the input trees) can be easily mapped to an instance of flip distance of size $n + 2$, such that the distance is preserved. The mapping goes both ways [12]. Hence, the kernel obtained in Theorem 4 (and that of Theorem 3) also goes through for rotation distance, up to additive terms.

Corollary 5. *For each $\epsilon > 0$ there is a kernel of size $(1 + \epsilon)k$ for the rotation distance problem.*

4. Experimental results

It is natural to ask how the kernels described in Theorems 3 and 4 function in practice. In this section we undertake a simple experiment to partially answer this question.

4.1. Experimental setup

The dataset for our experiment consists of pairs of random polygons of a certain size n and maximum distance D. The polygon pairs have been obtained through the following procedure:

1. Generate a random polygon by selecting uniformly at random two non-consecutive vertices to form a diagonal. Then split the polygon through that diagonal and recursively apply the same operation to both subpolygons until no more diagonals can be added.
2. Starting from the previous polygon we perform D random flips to it to obtain a second polygon. By random flip we mean that the diagonal we flip is selected uniformly at random.

This way we have an easy to compute lower bound on the distance, which is the number of non-common diagonals of the pair, and an upper bound on the distance which is given by D.

For each $n \in \{10, 50, 100, 500, 1000\}$, maximum distance $D \in \{0.1,\ 0.5,\ 1,\ 1.5,\ 2\}n$ and $\epsilon \in \{1, 1/3, 1/6,\ 1/9, 1/12, 1/15,\ 1/18, 1/21, 1/24,\ 1/27, 1/30\}$ we have generated 100 random pairs of polygons and executed a slightly modified version of the $(1 + \epsilon)k$ kernel algorithm described in Theorem 4. Note that $\epsilon = 1$ actually corresponds to the previous best kernel by Lucas, and $\epsilon = 1/3$ to the kernel from Theorem 3. Note also that we have chosen to express the maximum distance D as a fraction of the instance size n.

We modify the $(1+\epsilon)k$ kernel as follows. First, we do not compute the flip distances of the small polygons that are created by the decomposition procedure, i.e. those of size less than $1/\epsilon + 3$. We simply remove them. Removing them is reasonable because we are primarily interested in the reduction achieved by the kernel, not its running time. To obtain this information it is not necessary to solve the small instances. In particular, as explained in Theorem 4, the size of the kernel will be the result of merging all the polygons that remain. Of course, in a practical setting the running time of the kernel, which increases for decreasing ϵ, would have to be taken into account; we return to this point later.

Second, we will not use a parameter k, since the parameter k is actually only used to determine whether an early YES or NO conclusion can be reached (see Step 3 of Theorem 3). By leaving k out of the experiments we therefore potentially underestimate the power of the kernel: we might allow large NO instances to survive. However, this simplification is safe in the sense that, if the simplified kernel we have implemented performs well, a full-blown implementation of the kernel (that does take k into account) would perform at least as well.

For each of the mentioned combinations we measure the average, over the 100 pairs generated, of the following statistics:

- Kernel size. The size of the single pair of polygons after applying the kernel.
- Kernel size as a fraction of n. That is, kernel size/n.
- Theoretical kernel size. Let NC be the number of non-common diagonals before applying the kernel, then, the theoretical kernel size is defined as $(1 + \epsilon)NC + 2$. When applying the kernel for a certain k, we know that the size is at most $(1 + \epsilon)k + 2$. Since NC is a lower bound for the distance, we have $(1+\epsilon)k+2 \geq (1+\epsilon)NC+2$, so that the theoretical kernel size is lower than the upper bound for the size for every k. We do this because we do not know the true distance. Note that

comparing the kernel size obtained in practice, to the theoretical kernel size, is safe in the following sense: if the obtained kernel is smaller than the theoretical kernel size, then it will definitely be smaller than the true kernel bound i.e. the bound that is a function of the actual distance.

- Number of non-common diagonals before applying the kernel.
- Number of non-common diagonals after applying the kernel.
- Number of sub-instances removed. This is the number of small, i.e. of size less than $1/\epsilon+3$, instances removed by the kernelization algorithm before the merging step.

Not all these values are elaborated as results, for a detailed report on these values we refer to the repository[1] that contains all the code and data from the experiments.

4.2. Results

4.2.1. Kernel size

In Figure 4 we show for $n = 1000$ and $D = 1n = 1000$ how the size of the kernel evolves as ϵ decreases. We only show this n and D combination because, as can be observed in the full results in the repository, a similar pattern is observed regardless of D and n.

A number of observations can be made. First, the kernel size (blue line) is significantly smaller than the theoretical kernel size (orange line), meaning that the gap between the achieved reduction and the real upper bound will be even larger. Second, the kernel size is also significantly smaller than the natural lower bound of the number of non-common diagonals NC (dashed green line). Third, the $4k/3$ kernel (blue line at $\epsilon = 1/3$) achieves more reduction than Lucas' kernel (red dashed line), despite requiring no additional computational time.

4.2.2. Computational effort required to solve small instances

As mentioned earlier the running time of our kernelization algorithm increases as ϵ decreases due to the need to exactly solve small instances i.e. those of size less than $1/\epsilon + 3$ (see Theorem 4). In Table 1 we can see the maximum size of small instances that *in theory* would have to be computed exactly by the kernel, compared to the largest such instances we encountered during the experiment for $D = n = 1000$.

We see that here the observed maximum size of the small instances is somewhat smaller than the theoretical maximum. This was not always the case: for example, when D becomes very large relative to n the observed maximum often approaches the theoretical maximum. In any case, the table gives an idea of how future speed-ups of algorithms for exactly solving (small) flip distance instances can contribute to a pre-processing framework. Note also that, as we can see in Figure 4, the size of the kernel approaches zero for our smallest choices of ϵ, meaning that we have effectively reduced the problem of solving an instance of size 1000 to that of solving instances of size at most 32 (and often much smaller).

5. Discussion

Our kernel makes use of the fact that two polygons of size n with no common diagonals have $n-3$ non-common diagonals, which is a lower bound on the flip distance. Hence, for such "fully reduced" instances the ratio of the instance size to the flip distance is at most $\frac{n}{n-3}$ which is $1 + o(1)$. In this sense, our $(1 + \epsilon)k$ kernel feels like a natural result for this problem. We would like to remark that, unlike previous kernels, our approach manages to reduce the number of *non-common* diagonals in some instances, when discarding small instances of size larger than 3.

It would be interesting to explore alternative, less inflated parameterizations of the problem. For example, if we let d denote the number of non-common diagonals in an instance, we could ask: is the flip distance $\leq d + k$? We also note that there has been quite sophisticated parameterized complexity work undertaken on the flip distance problem in recent years, although most of it has been done on more general versions of it: in triangulations of point sets on the plane [5, 8]. We wonder whether those results can be strengthened in our more restricted setting i.e. the simple convex polygon case.

We note in passing that our improved kernel does not lead to an improvement of the polynomial-time approximation algorithm by Cleary et al. [2].

[1] See: `https://github.com/mbosch95/tri_kernel_exp`

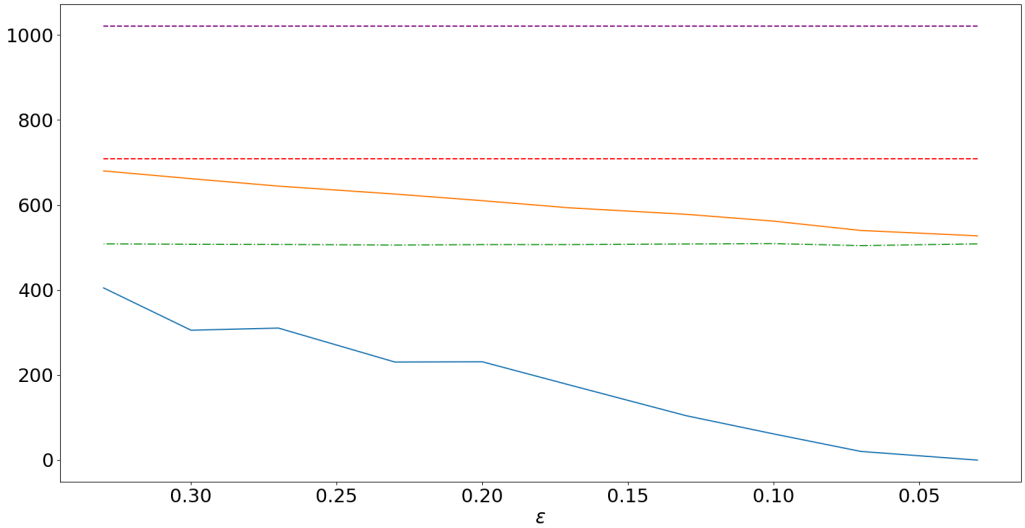

Figure 4: Results for the experiment when $n = 1000$ and $D = 1n = 1000$. Each datapoint is the average of the 100 polygon pairs generated for that specific combination of experimental parameters. The blue line shows the empirically observed kernel size, while the orange line indicates the theoretical kernel size. The dashed red line shows the observed size of the previous best kernel (by Lucas), and the dashed purple line shows its theoretical kernel size. Finally, the number of non-common diagonals of the original pair (before kernelization) is shown by the dashed green line.

ϵ	1/3	9/30	8/30	7/30	6/30	5/30	4/30	3/30	2/30	1/30
observed max n	5.00	6.00	6.00	7.00	7.00	8.00	9.98	11.82	15.26	19.11
theoretical max n	5	6	6	7	7	8	10	12	17	32

Table 1: (Average of) the observed maximum size n of small instances that need to be solved exactly when $n = D = 1000$ for each ϵ, compared to the theoretical maximum size (largest integer strictly less than $1/\epsilon + 3$) of small instances that we are required to solve.

That article uses a similar technique to Lucas, but the limiting factor there is the algorithmic upper bound, which is an algorithm that takes in the worse case two flips to fix each non-common diagonal.

Finally, we return to rotation distance. As stated in Corollary 5, we obtain (up to an additive difference of 2) the same kernel result for rotation distance. For us, the additive term is insignificant, but for Lucas [10] it can be of importance. Lucas uses the correspondence with rotation distance to derive the $2k$ kernel. The bound there is based on the observation that, after splitting at common diagonals and deleting distance-0 subinstances, and letting d be the total number of non-common diagonals, there can be at most d subinstances of pairs of polygons, each with a corresponding pair of trees, and each such subinstance (i.e. pair of trees) has at least one non-root interior node. The worst case is when there are d subinstances, each with exactly one non-root interior node. (In the rotation distance problem non-root interior nodes correspond to diagonals in the flip distance problem.) In the rotation distance literature the size of an instance is usually taken to be the number of interior nodes, *including* the root ([4, 9, 11, 12] among others). This yields a bound of $2d \leq 2k$. However, when translated to flip distance, the worst case corresponds to d subinstances, each of which has exactly one non-common diagonal (and no common diagonals). Such subinstances are squares, and in the vast majority of the literature the size of the polygons is regarded as the number of outer edges [5, 7, 8, 12]. Taking that metric, Lucas' kernel would yield $4d \leq 4k$ for flip distance, not $2k$, so the kernel distorts when using the usual sizes of the problems. In a nutshell: Lu-

cas left the kernel as a set of subinstances, but this can cause small additive terms to accumulate when switching between frameworks. Our kernel avoids such problems by merging the subinstances into a single instance at the end; this is the significance of the merging step.

6. Acknowledgements

We thank Steve Chaplick for useful discussions.

References

[1] O. Aichholzer, W. Mulzer, and A. Pilz. Flip distance between triangulations of a simple polygon is NP-Complete. *Discrete Comput. Geom.*, 54:368–389, 09 2015.

[2] S. Cleary and K. John. A Linear-Time Approximation Algorithm for Rotation Distance. *J. Graph Algorithms Appl.*, 14, 03 2009.

[3] S. Cleary and K. St. John. Rotation distance is fixed-parameter tractable. *Inf. Process. Lett.*, 109(16):918–922, 07 2009.

[4] K. Culik and D. Wood. A note on some tree similarity measures. *Inf. Process. Lett.*, 15(1):39–42, 1982.

[5] Q. Feng, S. Li, X. Meng, and J. Wang. An improved FPT algorithm for the flip distance problem. *Inf. Comput.*, page 104708, 2021.

[6] F. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing.* Cambridge University Press, 2019.

[7] F. Hurtado and M. Noy. Graph of triangulations of a convex polygon and tree of triangulations. *Comput. Geom.*, 13(3):179–188, 1999.

[8] I. Kanj, E. Sedgwick, and G. Xia. Computing the flip distance between triangulations. *Discrete Comput. Geom.*, 58:313–344, 2017.

[9] J. M. Lucas. Untangling Binary Trees via Rotations. *Comput. J.*, 47(2):259–269, 01 2004.

[10] J. M. Lucas. An improved kernel size for rotation distance in binary trees. *Inf. Process. Lett.*, 110(12):481–484, 2010.

[11] J. Pallo. An efficient upper bound of the rotation distance of binary trees. *Inf. Process. Lett.*, 73(3):87–92, 2000.

[12] D. Sleator, R. Tarjan, and W. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Am. Math. Soc.*, 1(3):647–681, July 1988.