Reconfiguring triangulations

by

Vinayak Pathak

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The results in this thesis lie at the confluence of triangulations and reconfiguration. We make the observation that certain solved and unsolved problems about triangulations can be cast as reconfiguration problems. We then solve some reconfiguration problems that provide us new insights about triangulations. Following are the main contributions of this thesis:

1. We show that computing the flip distance between two triangulations of a point set is NP-complete. A flip is an operation that changes one triangulation into another by replacing one diagonal of a convex quadrilateral by the other diagonal. The flip distance, then, is the smallest number of flips needed to transform one triangulation into another. For the special case when the points are in convex position, the problem of computing the flip distance is a long-standing open problem.

2. Inspired by the problem of computing the flip distance, we start an investigation into computing shortest reconfiguration paths in reconfiguration graphs. We consider the reconfiguration graph of satisfying assignments of Boolean formulas where there is a node for each satisfying assignment of a formula and an edge whenever one assignment can be changed to another by changing the value of exactly one variable from 0 to 1 or from 1 to 0. We show that computing the shortest path between two satisfying assignments in the reconfiguration graph is either in P, NP-complete, or PSPACE-complete depending on the class the Boolean formula lies in.

3. We initiate the study of labelled reconfiguration. For the case of triangulations, we assign a unique label to each edge of the triangulation and a flip of an edge from e to e' assigns the same label to e' as e. We show that adding labels may make the reconfiguration graph disconnected. We also show that the worst-case reconfiguration distance changes when we assign labels. We show tight bounds on the worst case reconfiguration distance for edge-labelled triangulations of a convex polygon and of a spiral polygon, and edge-labelled spanning trees of a graph. We generalize the result on spanning trees to labelled bases of a matroid and show non-trivial upper bounds on the reconfiguration distance.

Table of Contents

List of Figures

Chapter 1

Introduction

Triangulations are a well-studied mathematical object, and, as a research area, are mature enough to have textbooks [31, 37] written about them. Reconfiguration is a new field of study and has gained significant attention over the last decade. This thesis is at the confluence of triangulations and reconfiguration. There are long-standing open problems about triangulations that can be stated as reconfiguration problems. New results in reconfiguration often provide new insights for triangulations and problems about triangulations inspire new questions about reconfiguration. It is this feedback loop that has provided most of the fuel for this thesis.

In this chapter, we provide an overview of results presented in the thesis. In Section 1.1, we provide a summary of the known and relevant results in the fields of reconfiguration and triangulations. In Section 1.2, we describe our contributions. The overall goal of this chapter is to provide the commentary underlying the collection of results in the thesis. Thus we do not attempt to provide a comprehensive background for the thesis in this chapter. Instead, we only discuss those parts of the literature that are necessary to understand the motivation behind the questions we study and the significance of the results we have obtained. More thorough literature surveys pertaining to each chapter can be found at the beginning of the respective chapters.

1.1 Background

1.1.1 Reconfiguration

Reconfiguration problems are a way to study solution spaces of optimization problems. Consider the problem of computing the maximum independent set, for example. Given a maximum independent set I of a graph $G = (V, E)$, one defines a *reconfiguration step* as replacing a vertex $v \in I$ with a vertex $u \in V \setminus I$ such that the set $I \setminus \{v\} \cup \{u\}$ is also independent. Such a step is also called a *token jump*. Then, given two maximum independent sets I_1 and I_2 of G, deciding whether there exists a sequence of token jumps that transforms I_1 to I_2 is an example of a reconfiguration problem.

Similar reconfiguration problems can be defined for other problems, including vertex cover, coloring, maximum matching, shortest path, minimum spanning tree, and satisfiability of boolean formulas [59, 76, 58, 55, 47]. For all these cases, we can define a *reconfiguration graph* that has a vertex for each solution and an edge whenever the two solutions can be transformed into one another with the application of one reconfiguration step. The following kinds of reconfiguration problems have been studied so far.

1. Given two solutions, does there exist a sequence of reconfiguration steps that converts one into the other? That is, we want to solve *st-connectivity* on the reconfiguration graph.

2. Given an instance of an optimization problem, is the reconfiguration graph connected? That is, we want to decide the *connectivity* of the reconfiguration graph.

3. Given two solutions, what is the smallest number of reconfiguration steps required to convert one to the other? We call this quantity the *reconfiguration distance* and this path the *shortest reconfiguration path (SRP)* between the two solutions.

4. What is the diameter of the reconfiguration graph in the worst case? Note that unlike the previous three problems, this one is not a computational problem.

Ideas related to reconfiguration have appeared in the mathematical literature for more than a century and thus it is difficult to identify the first work on reconfiguration. For example, one could consider computing any permutation of a given set as a trivial optimization problem and define a reconfiguration step as swapping two adjacent elements. The reconfiguration graph thus obtained is the permutohedron, which was first studied in 1911 [88]. The recent interest in reconfiguration, however, is from a more computational

point of view and it is reasonable to say that the first work of this flavor was published in 2009 by Gopalan et al. [47], where they studied reconfiguring between satisfying assignments of Boolean formulas. They studied st-connectivity, connectivity, and the worst-case diameter of the reconfiguration graph. Their work, combined with the more recent work of Schwerdtfeger [89], completely resolves these three problems for Boolean formulas. In particular, there exists a partition of the set of Boolean formulas into two mutually exclusive and exhaustive classes such that st-connectivity can be computed for one of them in polynomial time and is PSPACE-complete for the other. There exist similar dichotomies for the other two problems as well. In Chapter 3 (also [75]) we prove a similar partition result for the problem of computing the shortest reconfiguration path. Further background on reconfiguration of Boolean formulas can be found in Chapter 3.

Almost around the same time as Gopalan et al.'s work [47], Ito el al. [55] published another collection of important results. They considered the problem of st-connectivity for independent set, clique, vertex cover, set cover, and integer programming and proved them all to be PSPACE-complete. On the other hand, for matching and spanning trees, they provided polynomial time algorithms for st-connectivity.

One pattern that emerged from Ito et al.'s work was that the reconfiguration versions of tractable (in P) optimization problems happened to be in P and the reconfiguration versions of intractable (NP-hard) optimization problems happened to be PSPACE-complete. There exists a counterexample to the second pattern in the regime of Boolean formulas, i.e., there exists a class of Boolean formulas where deciding satisfiability is NP-complete but st-connectivity in the reconfiguration graph can be solved in polynomial time [47]. A counterexample to the first pattern was obtained by Kaminski et al. in 2011 [58] where they showed that deciding st-connectivity in the reconfiguration graph of shortest paths was NP-hard; it was later shown to be PSPACE-complete by Bonsma [12].

It was proved in the same paper [12] that reconfiguring between shortest paths in claw-free and chordal graphs can be done in polynomial time, that the diameter of the reconfiguration graph is linear in these cases, and that the connectivity of the reconfiguration graph can also be tested in polynomial time. Bonsma later also studied reconfiguration of shortest paths in planar graphs and showed that to be in P [13]. Many other reconfiguration problems have been studied and the interested reader is referred to the survey by van den Heuvel [98].

1.1.2 Triangulations

For a given planar point set P, let an *edge* be a line segment connecting two points in P with no other point lying on it. We will say that two edges *cross* if they intersect at a point that is not an end point. A triangulation of P is any maximal set of edges such that no two edges cross each other (Figure 1.1(a)). It is known that every maximal set of edges such that no two edges cross each other always contain the convex hull edges, and divides the interior of the convex hull into triangles.

It is clear from an easy application of Euler's formula that all triangulations of a given point set have the same number of edges. It is known that all point sets have at most $O(30^n)$ triangulations [90], and there exist point sets with at least $\Omega(2.43^n)$ triangulations [91]. Finding an asymptotically tight bound is still open.

One can also define a triangulation for a simple polygon by defining an *edge* to mean any diagonal of the polygon (as opposed to segments joining any two vertices). Any maximal set of diagonals such that no two of them cross constitutes a triangulation of the polygon (Figure 1.1). For a point set that is in convex position, any triangulation can be seen as a triangulation of the convex polygon formed by the convex hull of the point set.

A *flip* is an operation that changes one triangulation into another by replacing one edge by another. In all the cases above, when we remove an edge e there is at most one edge $e' \neq e$ that can be added back to form a triangulation. Replacing e by e' is called a *flip*. Edges on the convex hull cannot be flipped. Let $e = BC$ be an edge with two triangles ABC and BCD on either side (Figure 1.1(a)). If the quadrilateral $ABDC$ is convex, then BC can be replaced by AD and thus $e' = AD$. If $ABDC$ is not convex, then we say that e is not *flippable*.

Flips define a *flip graph* that has a node for each triangulation and edge whenever exactly one flip is required to convert between the two triangulations. As explained below, the flip graphs always happen to be connected [64, 63, 7], which makes flips a very fundamental operation. The connectedness implies for any two triangulations of the same underlying object, there always exists a flip sequence that transforms one to the other.

One of the earliest mentions of flips was in the context of Delaunay triangulations—there exists an $O(n^2)$ algorithm to construct the Deulaunay triangulation using flips. A Delaunay triangulation of a point set is defined as the triangulation which contains an edge between two points if and only if there exists a circle that contains only those two points and no other points. Delaunay triangulations satisfy some very nice properties, providing a list of which is out of scope for the thesis. However, the algorithm for their construction

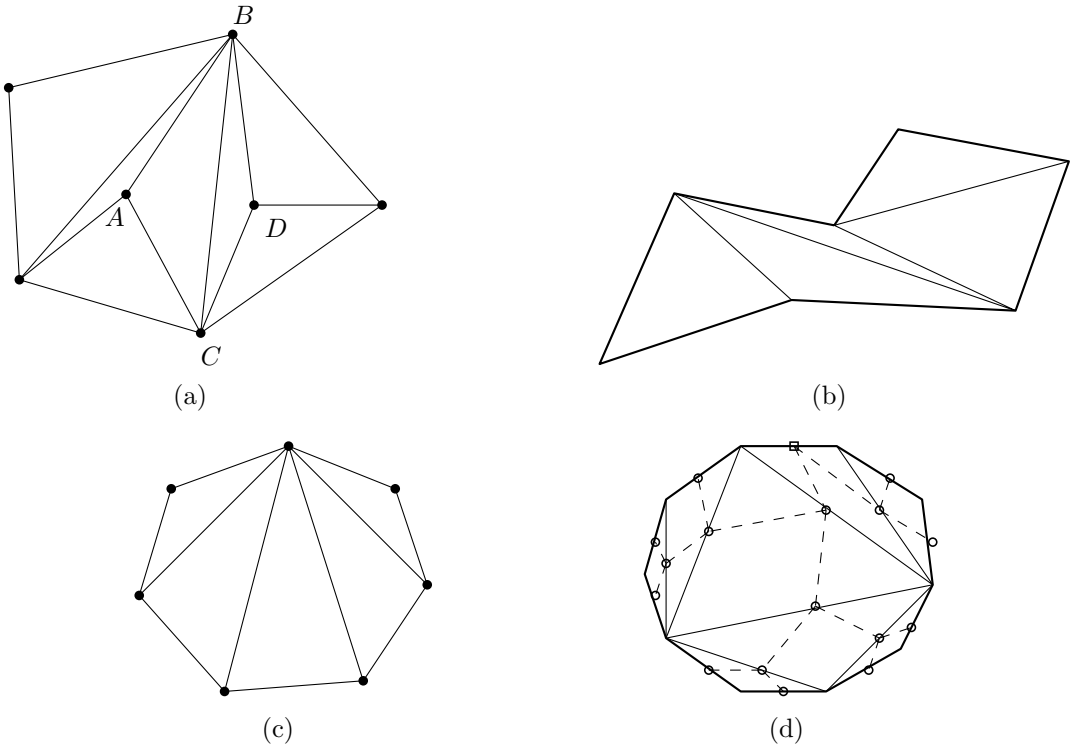


Figure 1.1: (a) Triangulation of a point set (b) Triangulation of a polygon (c) Canonical triangulation of a point set in convex position (d) Triangulation of a convex polygon and the corresponding BST (dashed lines indicate edges, circles indicate nodes, square indicates root)

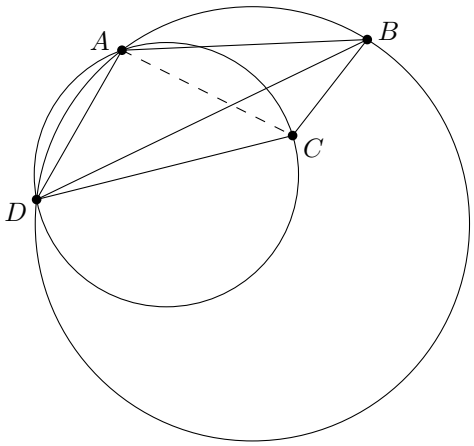


Figure 1.2: Flipping a non-Delaunay edge. BD is non-Delaunay since the circumcircle of triangle ABD contains C. Thus it can be flipped to AC, which is a Delaunay edge.

is instructive and explains why the flip graph is connected. Thus we provide a description of the algorithm in the next paragraph.

Consider any triangulation of a point set and consider any convex quadrilateral $ABCD$ formed by four edges of the triangulation. Let BD be the diagonal present in the triangulation. If the circumcircle of triangle ABD contains C, or if the circumcircle of triangle BCD contains A, then the triangulation is not Delaunay: since there exists a circle passing through B and D that contains either A or C, edge BD cannot be an edge of the Delaunay triangulation. We say that the edge BD is a non-Delaunay edge (Figure 1.1.2). A simple geometric argument shows that if BD is non-Delaunay, then AC must be Delaunay, i.e., the circumcircle of ABC does not contain D and the circumcircle of ACD does not contain B. Replacing BD by AC is exactly one flip. The algorithm is now simple to describe: as long as the triangulation contains a non-Delaunay edge, flip the edge. We will not go into the details of the proof, but it can be shown that this process always yields a Delaunay triangulation in $O(n^2)$ steps (see [37] for a proof). Since any triangulation can be transformed into the Delaunay triangulation in $O(n^2)$ flips, any two triangulations can be transformed into one another by going through the Delaunay triangulation in $O(n^2)$ flips.

This argument can be generalized to triangulations of simple polygons (see [7] for details) by going through the *constrained* Delaunay triangulation thus showing that the flip graph of triangulations of a simple polygon is always connected. When points are in convex position, this argument already shows the connectivity of the flip graph, but the $O(n^2)$ bound on the worst-case flip distance can be reduced to $2n + O(1)$ with a

more careful argument [93]. Both arguments produce flip sequences that pass through a canonical triangulation, which, for point sets, is the Delaunay triangulation, and for points in convex position, is a *fan* (Figure 1.1(c)).

The minimum number of flips required to transform one triangulation to another—the flip distance—acts as a measure of similarity between the two triangulations. It is often useful to compute the flip distance given the two triangulations as input. One context where such measures are useful is in the design of data structures [93] because of a connection with binary search trees (BST). The set of triangulations of a convex n-gon is in bijection with the set of binary search trees on $n-1$ leaves such that the flip operation on the triangulation corresponds to a rotation operation on the corresponding tree (Figure 1.1(d)). Thus the flip distance corresponds to the *rotation distance*—minimum number of rotations required to transform one tree to another. Measures of similarity between two trees, such as, the rotation distance, are also used in comparative genomics where evolution is represented by phylogenetic trees [40].

Thus computing flip distance is useful and fundamental. Interestingly, for the case of convex polygons, the complexity of this problem has been open since 1987 [93]. The problem is also important for planar point sets and has appeared in two textbooks and a survey [33, 31, 15]. We solved the problem in 2011 [66] and showed that for planar point sets computing the flip distance is NP-complete (see Chapter 2). It was later also shown to be APX-hard [84]. The problem on simple polygons is now known to be NP-complete [1]. A factor-2 approximation algorithm exists for convex polygons, but no constant-factor approximation algorithms are known for point sets or simple polygons. The factor-2 algorithm is easy, however, no improvements are known except for an algorithm that achieves factor 1.98 in some special cases [65].

A special case of point sets was considered by Eppstein [38], namely, point sets that contain no empty convex pentagons, that is, any convex pentagon that can be formed with points from the point set as its vertices must have another point inside it or on its boundary. Eppstein showed that the flip distance for such point sets can be computed in polynomial time. A complete rectangular grid of points is an example of such a point set and was independently studied by Caputo et al. [20]. Points in convex position and points with no empty convex pentagons lie on two opposite ends of a spectrum since for a point set in convex position every convex polygon formed by a subset of them is empty. Eppstein's algorithm uses the fact that the flip graph of triangulations for a point set with no empty convex pentagons is a partial cube, i.e., a subgraph of the hypercube where the shortest distance between two vertices is exactly equal to the Hamming distance between them.

1.2 Our contributions

The new results in this thesis fall under two general themes. The first theme is about computing the reconfiguration distance, which is a more general version of the problem about computing the flip distance between two triangulations. We study the problem for triangulations and for solutions of Boolean formulas. Our results on Boolean formulas settle the complexity of computing the reconfiguration distance for a large class of fundamental problems. We discuss more details in Section 1.2.1 and Chapters 2 and 3.

The second theme is about *labelled* reconfiguration. Consider any traditional (i.e., unlabelled) reconfiguration problem about transforming between different solutions of an optimization problem using reconfiguration steps. A reconfiguration step usually replaces a part of the solution with another part. Thus any solution can be divided into parts that a reconfiguration step operates on. In labelled reconfiguration, we assign a unique label to each part of a solution and a reconfiguration step preserves this labelling. For example, for a triangulation, each edge gets a label and if a flip replaces edge e with e', we assign the same label to e' as e. We study labelled reconfiguration of triangulations, and of independent sets of matroids. More details can be found in Section 1.2.2 and Chapters 4, 5, and 6.

1.2.1 Computing the reconfiguration distance

Given a planar point set P, consider the graph G that has a vertex for each edge of P and (u, v) is an edge of G if the edge corresponding to u and the edge corresponding to v cross. It is easy to see that any triangulation of P corresponds to a maximum independent set of G and a flip corresponds to a token jump. Thus studying flips in triangulations is exactly equivalent to studying independent set reconfiguration in a special class of graphs. In fact, the reconfiguration graph for this problem is exactly the flip graph. Having made this observation, we can think about triangulations in the context of reconfiguration problems and ask about the complexity of st-connectivity, connectivity and the worst-case bounds on diameter. The huge knowledge base about triangulations establishes the answers to three of the problems listed above—st-connectivity and connectivity are both in P since the reconfiguration graph is always connected, and the worst-case diameter is $\Theta(n^2)$ for point sets [64] and simple polygons [7] and $\Theta(n)$ for convex polygons [93]. In fact, for the case of convex polygons, the worst-case bounds are known to the exact constant, i.e., it is known to be $2n - 10$. On the other hand, the question of computing the shortest reconfiguration path is open for convex polygons. Our first result is that the problem is

NP-complete for point sets [66] and Chapter 2 is devoted to that result. Note that the case of simple polygons has also been proved to be NP-complete [1]. Nothing much is known about the best approximation factor achievable, and it is an interesting open question. Along similar lines, the parametrized complexity has been studied and all these versions are known to be fixed parameter tractable [25, 60].

Even though we claimed in Section 1.1 that shortest reconfiguration path is one of the four different kinds of reconfiguration problems studied in the literature, before our work, it was studied only implicitly, i.e., the only cases where a polynomial-time algorithm was known were the ones where the algorithm for st-connectivity happened to find the shortest reconfiguration path itself. Examples include reconfiguration of spanning trees, matchings, and of satisfying assignments of 2CNF formulas [55, 47]. Our next contribution is to initiate the study of the problem of computing the shortest reconfiguration paths by investigating it for satisfying solutions of Boolean formulas. Our results are similar to the results obtained in the first reconfiguration paper by Gopalan et al. [47] in the sense that we provide a complete classification of Boolean formulas into mutually exclusive and exhaustive classes based on the complexity of the problem of computing the reconfiguration distance. Chapter 3 is devoted to this result.

Out of a myriad of problems for which we could have studied the complexity of reconfiguration distance, our choice of Boolean formulas is justified for the following reasons. Satisfiabililty of Boolean formulas is fundamental from a complexity theoretic point of view since all problems in NP can be reduced to it. Boolean formulas are also amenable to a more systematic investigation due to a framework invented by Schaefer [87] to classify them. Due to this framework, researchers have been able to show a complete characterization for several problems about Boolean formulas (see [28] for a survey). Finally, reconfiguration on Boolean formulas shares some similarities with flips in triangulations. In particular, the class of NAND-free formulas (as defined by Gopalan et al. [47]) has the property that st-connectivity is trivial to solve on it even though computing the reconfiguration distance is not always trivial, which is exactly what happens with flips in triangulations. The similarity, in fact, goes deeper than that. The algorithm for st-connectivity on NAND-free formulas works as follows. Flip 0's to 1's in both satisfying assignments as long as you can while maintaining feasibility; return 'yes' if both assignments lead to the same solution and 'no' otherwise. This is reminiscent of the algorithm for flipping between two triangulations where we flipped non-Delaunay edges to Delaunay edges as long as we could and stopped when we reached the same triangulation starting from both the given triangulations.

An interesting property about triangulations of a convex polygon was first discovered in [93] and seems to be fundamental. For two triangulations of a convex polygon, it was shown that a shortest flip sequence that transforms one triangulation into another

never flips diagonals that are common between the two. This property does not hold for point sets and simple polygons. In fact, the NP-completeness proof relies crucially on this fact, which gives the impression that perhaps this is the property that distinguishes cases where computing flip distance is NP-complete from cases where it is in P. Our investigation into computing the reconfiguration distance for Boolean formulas throws some light on this issue. The only class of Boolean formulas where computing the reconfiguration distance was known to be in P before our result was 2CNF formulas and they satisfied the property that the shortest reconfiguration path left those variables unchanged that had the same values in the two assignments. We showed that the reconfiguration distance can be computed for a bigger class of formulas that includes classes where the shortest reconfiguration path flips variables that are assigned the same value.

Another uncanny similarity between reconfiguration of triangulations and of solutions of Boolean formulas is that before our result, the only known class of Boolean formulas where the reconfiguration distance could be computed in polynomial time—i.e., 2CNF formulas—had the property that its reconfiguration graph was a partial cube, and interestingly, the only class of point sets where computing the flip distance was known to be in P—i.e., Eppstein's point sets—also had the property that their reconfiguration graph was a partial cube. In fact, the polynomial time algorithm in both cases crucially used the fact that the graph was a partial cube. Our result on reconfiguration distance for Boolean formulas exhibits a class, namely, *navigable* formulas, where the reconfiguration graph is not always a partial cube. Thus it is not unreasonable to anticipate that there might exist a superclass of Eppstein's point set where the flip distance can also be computed in polynomial time.

1.2.2 Labelled reconfiguration

Traditionally, most reconfiguration problems are about reconfiguring between subsets of a bigger set. For example, all of independent set, vertex set, spanning tree, and matching are subsets of the set of vertices or edges of a graph. In all the reconfiguration problems studied so far, all representations of the subset are treated the same. In particular, the order in which the elements of the subset are listed is not important. We initiate the study of *labelled* reconfiguration problems where the subsets are ordered.

For example, we define an *edge-labelled* triangulation as a triangulation with a unique label assigned to each of its non-boundary edges. If a flip replaces edge e with edge e' we assign the same label to e' as e. Two edge-labelled triangulations are said to be the same if they have the same edges and the edges have the same labels. We can then ask the labelled versions of all the questions we consider in the unlabelled setting. More

generally, for independent set reconfiguration, we assign a unique label to each element of the independent set and if a token jump replaces v with u, then u gets the same label as v.

The following intuitive argument suggests that labeling edges might have some relevance to the problem of computing the flip distance in the unlabelled case. Given two unlabelled triangulations T_1 and T_2, assign labels to the edges of T_1. Any flip sequence that transforms T_1 to T_2 will induce a particular labelling to the edges of T_2 based on which edge of T_1 takes place of which edge of T_2, and different flip sequences that transform T_1 to T_2 may induce different labels on T_2. The shortest flip sequences will have the property that it will assign the same label to edges that can be flipped into each other easily. Providing the labels of both T_1 and T_2 as input is like providing the labelling induced on T_2 by the shortest flip sequence. Does this extra information make the problem easier to solve?

Eppstein has shown that the flip distance between triangulations of planar point sets that do not contain an empty convex pentagon can be computed in polynomial time [38]. These kinds of point sets have the property that given a particular labelling for T_1, any flip sequence induces the same labelling on the edges of T_2. Moreover, this labelling can be found in polynomial time given T_1 and T_2. Thus the only known case where computing the flip distance is easy is where finding the labelling induced on T_2 by the optimal flip sequence is also easy. This may be interpreted as evidence that the difficulty of computing the flip distance lies in computing the labels induced on T_2 by the optimal flip sequence.

There exists an example in the context of independent set reconfiguration where the labelled case has the same complexity as the unlabelled case. Reconfiguration of independent sets is known to be PSPACE-complete on perfect graphs [59]. Since perfect graphs have the property that the maximum independent set has the same size as the minimum cover by cliques, the set of vertices of a perfect graph can be partitioned into subsets such that any maximum independent set contains exactly one vertex from each subset. Thus a labelling of the vertices in I_1 induces a unique labelling on the vertices in I_2 for any reconfiguration sequence that converts I_1 to I_2 and this labelling can be found in polynomial time. Thus in the realm of independent set reconfiguration, having a unique induced labelling does not make computing shortest reconfiguration sequences any easier.

Assigning labels to edges leads to some interesting results. For example, the flip graph is no longer connected for the case of planar point sets! In Chapter 4 we initiate the study of labelled reconfiguration. We consider edge-labelled triangulations of convex polygons in Chapter 4, and of spiral polygons, simple polygons, and planar point in Chapter 5. We also consider labelled reconfiguration for independent sets of matroids in Chapter 6, which is equivalent to studying reconfiguration of edge-labelled spanning trees of a graph when

the matroid is graphic. In all the cases above, we consider the problems of st-connectivity, connectivity, and diameter of the reconfiguration graph. We show that in all cases except simple polygons and planar point sets, there exists a flip sequence that transforms one labelled object into another if and only if for each label, there exists a flip sequence that transforms the edge with that label in the initial object to the edge with that label in the final object. We conjecture that the above is true for simple polygons and planar point sets as well. We also provide bounds on the diameter of the flip graph.

Labelled reconfiguration has not been studied in the context of reconfiguration problems, but problems of a similar flavor have been studied in the past. For example, problems about permutations can be considered as labelled reconfiguration since a permutation is an ordered subset. We can define various reconfiguration steps and study all four kinds of reconfiguration problems for them. Interestingly, this area is of interest to the Bioinformatics community since genes can often be modelled as permutations and mutations by reconfiguration steps. Many problems about reconfiguring permutations have been studied and the central theme is that computing the reconfiguration distance happens to be NP-complete in most cases but lies in P for "signed" permutations. The reader is referred to the textbook by Fertin et al. [40] for a thorough survey.

Another place where one can find hints of labelled reconfiguration is in the area of puzzles. The 15-puzzle, which is a canonical example of a puzzle, can also serve as a canonical example of a labelled reconfiguration problem. The puzzle can be thought of as a certain reconfiguration problem on a "grid" graph and has been generalized to other kinds of graphs. See the survey by van den Heuvel [98] for details.

There exists a problem in the realm of puzzles where computing the reconfiguration distance is in P for the unlabelled version but is NP-complete for the labelled version. Consider the problem of reconfiguring arbitrary sets of vertices of a given graph. Here the reconfiguration step is *token sliding*, i.e., a vertex v can be replaced with a vertex u if and only if (u, v) is an edge. Computing the shortest sequence of token slides is in P, as shown in [98]. However, if we label the vertices of the set, Goldreich [46] showed the problem to be NP-complete. In fact, even if exactly one of the vertices is labelled, the problem remains NP-complete [97].

We do not have definitive answers about whether labelling the solution makes the problem of computing the reconfiguration distance easier or harder, but we provide the first results for labelled reconfiguration and formulate interesting open questions.

Chapter 2

Flip distance between triangulations of a planar point set[1]

In this chapter, we show that computing the flip distance between two triangulations of a planar point set is NP-complete.

2.1 Background

Flips have been studied in the geometric setting for triangulations of point sets and of polygons. In this context, a convex polygon is equivalent to a point set in convex position. The former generalizes to simple polygons, and the latter to planar point sets. Both of these are contained in the most general case of a polygon with holes (a "polygonal region"), so long as we consider a point as a one vertex polygonal hole. There is a survey on flips by Bose and Hurtado [15]. It also covers flips in the combinatorial setting of maximal planar graphs, which we will not discuss. Flips are often studied in terms of the *flip graph* which has a vertex for every triangulation and an edge when two triangulations differ by one flip, see e.g., [38].

The foundational result is that the flip graph is connected. This was proved first by Lawson [64] for the case of point sets. He then re-proved the result [63] by arguing that any triangulation can be flipped to the Delaunay triangulation, which then acts as a "canonical" triangulation from which any other triangulation can be reached. The constrained

[1]This chapter represents joint work with Anna Lubiw [66].

Delaunay triangulation can be used in the same way to argue that any polygonal region has a connected flip graph [7]. For more direct proofs see [36, 52, 79].

Regarding the number of flips needed to transform one triangulation to another, flipping via the [constrained] Delaunay triangulation takes $O(n^2)$ flips—in fact, a more exact bound is the number of visibility edges, see [7]. Hurtado, Noy and Urrutia [52] proved that $\Omega(n^2)$ flips may be required even for triangulations of a polygon. For the case of a convex polygon, Sleator et al. [95] proved that for large values of n, the flip distance between two triangulations of an n-gon is at most $2n - 10$, and that $2n - 10$ flips are sometimes necessary. A recent result by Pournin [86] proves the same bound for all values of n using combinatorial methods.

The problem of computing the exact flip distance between two given triangulations is especially interesting for convex polygons since it was first stated in 1982 and has been open since then [29]. Lucas [68] gave a polynomial time algorithm for special cases. The best approximation factor is trivially 2, and can be improved in some special cases [65]. Recently it was proved that the problem is fixed-parameter tractable in the flip distance [25]. Attempts have also been made to compute good upper and lower bounds on the flip distance efficiently. See, for example, [5, 81, 69, 32].

The more general problem of computing the flip distance between two triangulations of a point set is stated as an open problem in the survey by Bose and Hurtado [15], the book by Devadoss and O'Rourke [33, Unsolved Problem 12], and the book on triangulations by De Loera et al. [31, Exercise 3.18]. Hanke et al. [50] proved that the flip-distance is upper bounded by the total number of intersections between the overlap of the initial and final triangulations. Eppstein [38] provided an algorithm to compute a lower bound on the flip-distance efficiently. He also showed that the lower bound is equal to the flip-distance for certain special kinds of point sets.

Pilz [83] independently proved that computing the flip distance between triangulations of a point set is NP-complete and later extended the result to show that it is, in fact, APX-hard [84] as well. Aichholzer et al. [1] studied the problem of computing the flip distance between triangulations of a simple polygon and showed it to be NP-complete.

2.2 Triangulations of polygonal regions

Theorem 2.1. *The following problem is NP-complete: Given two triangulations of a polygon with holes and a number k, is the flip distance between the two triangulations at most k?*

2.2.1 Proof idea

Note that the problem lies in NP since the flip-sequence of size at most k is itself a polynomial-sized certificate. We prove hardness by giving a polynomial time reduction from vertex cover on 3-connected cubic planar graphs [8, 100], which is known to be NP-complete [8, 100].

The idea is to take a planar straight-line drawing of the graph and create a polygonal region by replacing each edge by a "channel" and each vertex by a "vertex gadget". We then construct two triangulations of the polygonal region that differ on the channels, and show that a short flip sequence corresponds to a small vertex cover in the original graph.

We begin by describing channels and their triangulations, because this gives the intuition for the proof. A *channel* is a polygon that consists of two 7-vertex reflex chains joined by two *end* edges, as shown in Figures 2.1(a) and 2.1(b). Note that every vertex on the upper reflex chain sees every vertex on the lower reflex chain and vice versa. We identify two triangulations of a channel: a *left-inclined triangulation* as shown in Figure 2.1(a); and a *right-inclined triangulation* as shown in Figure 2.1(b).

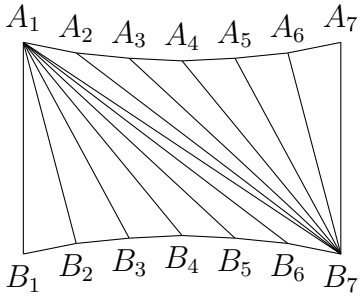
A channel is the special case $n = 7$ of the polygons H_n of Hurtado et al. [52]. They prove in Theorem 3.8 that the flip distance between the right-inclined and left-inclined triangulations of H_n is $(n-1)^2$. We include a different proof in order to generalize:

Property 2.1. *Transforming a left-inclined triangulation of a channel to a right-inclined triangulation takes at least 36 flips.*

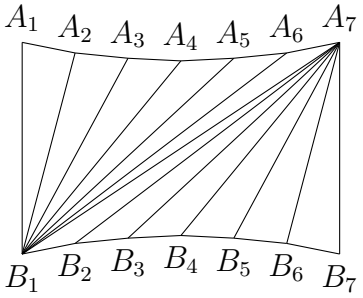
Proof. In any triangulation of a channel, each edge of the upper reflex chain is in a triangle whose apex lies on the bottom reflex chain. This apex must move from lower right (B_7) to lower left (B_1), in order to transform the left-inclined triangulation to the right-inclined triangulation. Similarly, each edge of the lower reflex chain is in a triangle whose apex lies on the upper reflex chain, and must move from upper left to upper right. However, one flip can only involve one edge of the upper chain and one edge of the lower chain (no other 4 vertices form a convex quadrilateral), and thus can only move one upper and one lower apex, and only by one vertex along the chain. Twelve triangles times six apex moves per triangle divided by two apex moves per flip gives a lower bound of 36 flips. □

We now show that the number of flips goes down if a channel has a *cap*, an extra vertex that is visible to all the channel vertices, as shown in Figure 1(c).

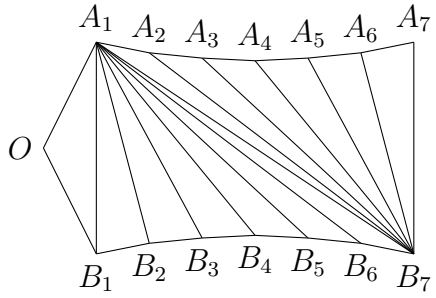
Property 2.2. *The flip distance from a left-inclined to a right-inclined triangulation of a capped channel is 24.*



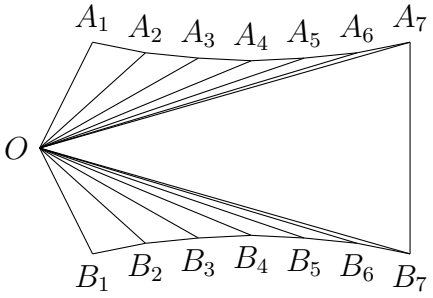
(a) A left-inclined channel.



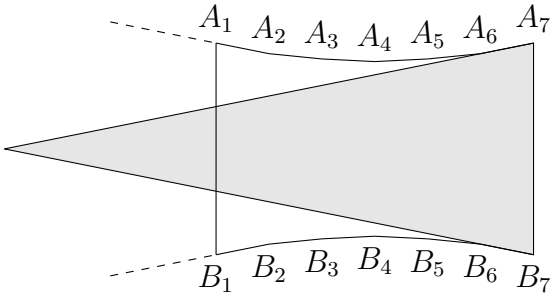
(b) A right-inclined channel.



(c) A capped channel.



(d) The canonical channel



(e) Narrow (shaded) and wide (dashed) mouths.

Figure 2.1: Channels

Proof. The canonical triangulation shown in Figure 2.1(d) is 12 flips away from both the left-inclined and the right-inclined triangulations of a capped channel: To flip the left-inclined triangulation to the canonical triangulation, flip edges $A_1B_1, \ldots, A_1B_7$ followed by edges $A_2B_7, \ldots, A_6B_7$ in that order. Similarly for the right-inclined triangulation.

For the lower bound, we follow the same idea as above. In any triangulation, each edge of the upper [lower] reflex chain is in a triangle whose apex is either the cap or a vertex of the lower [upper] chain. There are only two kinds of flips: (1) a flip involving the cap vertex, an edge of one chain, and a vertex of the other chain; and (2) a flip involving one edge of each chain. A flip of type (1) moves the apex of only one triangle, and moves the apex to or from the cap. If a triangle is altered by a flip of type (1) then at least two such flips are required, one to move the apex to the cap and one to move the apex from the cap. If a triangle is only altered by flips of type (2), then, as above, it costs 3 flips to get the apex to its destination. Thus the 12 triangles require at least 24 flips. □

We now elaborate on the idea of our reduction. We create a polygonal region by replacing each edge in the planar drawing by a channel, and each vertex by a vertex gadget. We make two triangulations of the polygonal region. In triangulation T_1 all edge channels are left-inclined and in T_2 all edge channels are right-inclined. The triangulations are otherwise identical. We design vertex gadgets so that making a few flips in a vertex gadget creates a cap for a channel connected to it. Since transforming a channel from left-inclined to right-inclined is less costly if it is capped, the minimum flip sequence that transforms all the channels is obtained by choosing the smallest set of vertices that covers all the edges and using them to cap all the channels. Thus, intuitively, a minimum flip sequence corresponds to a minimum vertex cover.

One complication is that we cannot construct a vertex gadget for a *sharp* vertex—a vertex of degree 3 where one of the three incident angles in the planar drawing is greater than or equal to π. Therefore, we first show how to eliminate sharp vertices. Let G be our given 3-connected cubic planar graph. Using a result of Bárány and Rote [4], we can find, in polynomial time, a *strictly convex* drawing of G on a polynomial-sized grid. *Strictly convex* means that each face is a strictly convex polygon. Thus the only sharp vertices of this drawing are the vertices of the outer face. We replace each sharp vertex v by a 3-vertex chain v_1, v_2, v_3 as shown in Figure 2.2. We claim that G has a vertex cover of size $\leq k$ if and only if the modified graph has a vertex cover of size $\leq k + t$, where t is the number of vertices on the outer face of G. This is because any minimum vertex cover of the modified graph can be adjusted to use either $\{v_1, v_3\}$ (corresponding to v being in the vertex cover of G), or $\{v_2\}$ (corresponding to v not being in the vertex cover of G).

Figure 2.2: Eliminating sharp vertices

We remark that Pilz's independent NP-hardness reduction [84] is from general (non-planar) vertex cover. His construction begins with the same channel gadgets, but then uses channels that overlap geometrically while flipping independently.

2.2.2 Details of the reduction

For the remainder of the proof we will assume that we have a graph G with vertices of degree 2 and 3, and a straight-line planar drawing, Γ, of the graph on a polynomial-sized grid with no sharp vertices.

We define the *narrow* and *wide* mouths of a channel as shown in Figure 2.1(e). Any point inside the narrow mouth but outside the channel can be a potential cap for the channel. We show below that a vertex outside the wide mouth does not reduce the flip distance.

We now describe the triangulated vertex gadgets. See Figures 2.3(a) and 2.3(b). Each of the 2 or 3 channels attached to the vertex gadget will have one potential cap. We place a convex quadrilateral $CDEF$ with diagonal CE, called the *lock*, that separates each channel from its potential cap. Thus the lock CE must be flipped, or "unlocked", in order to cap any channel.

For the degree-2 gadget (see Figure 2.3(a)), place point C in the smaller angular sector (of angle $< \pi$) between the two channels, so that C is outside the wide mouths of both channels. Place points D, E, and F in the other angular sector, with D inside channel 1's narrow mouth and outside channel 2's wide mouth, E outside the wide mouth of both channels, and F inside channel 2's narrow mouth and outside channel 1's wide mouth. Triangulate as shown. Thus D is a potential cap for channel 1 and F is a potential cap for channel 2.

For the degree-3 gadget (see Figure 2.3(b)), note that because the vertex is not sharp, the mouth of each channel exits between the other two channels. We place vertices in

the angular sectors as shown in the figure. Place D inside the intersection of the narrow mouths of channels 1 and 2, and outside the wide mouth of channel 3. Place F inside channel 3's narrow mouth and outside channel 1 and 2's wide mouths. Place C and E outside the wide mouths of all the channels. Triangulate as shown. Thus D is a potential cap for both channel 1 and 2 and F is a potential cap for channel 3.

Observe that every channel is blocked from its unique potential cap by exactly 3 edges. (For example, in Figure 2.3(b), channel 1 is separated from its potential cap D by edges FA, FE, and CE.) Observe furthermore that for each vertex gadget, the sets of blocking edges of the channels have one edge in common, namely the locking edge CE, and are otherwise disjoint. These properties are crucial for correctness.

We will say that a vertex gadget is *locked* if the diagonal CE exists and *unlocked* otherwise. We first show what is possible with unlocked vertex gadgets.

Property 2.3. *If we unlock a vertex gadget then, for each channel attached to it, there is a sequence of 28 flips that transforms the channel triangulation and returns the vertex gadget to its (unlocked) state.*

Proof. We first claim that there is a 2-flip sequence that caps the channel. We enumerate the possibilities (refer to Figure 2.3). Note that we handle channels one at a time, not simultaneously. For the degree-2 gadget: flip CF followed by CA for channel 1; flip CD followed by CB' for channel 2. For the degree-3 gadget: flip FE followed by FA for channel 1; flip CF followed by CA' for channel 2; flip ED followed by EA'' for channel 3. Once the channel is capped, we can transform the left-inclined triangulation to the right-inclined triangulation in 24 flips by Property 2.2. Then we undo the 2 flips that capped the channel. The total number of flips is 28. □

Next we give lower bounds on the number of flips. First, note that the proof of Property 2.1 carries over to:

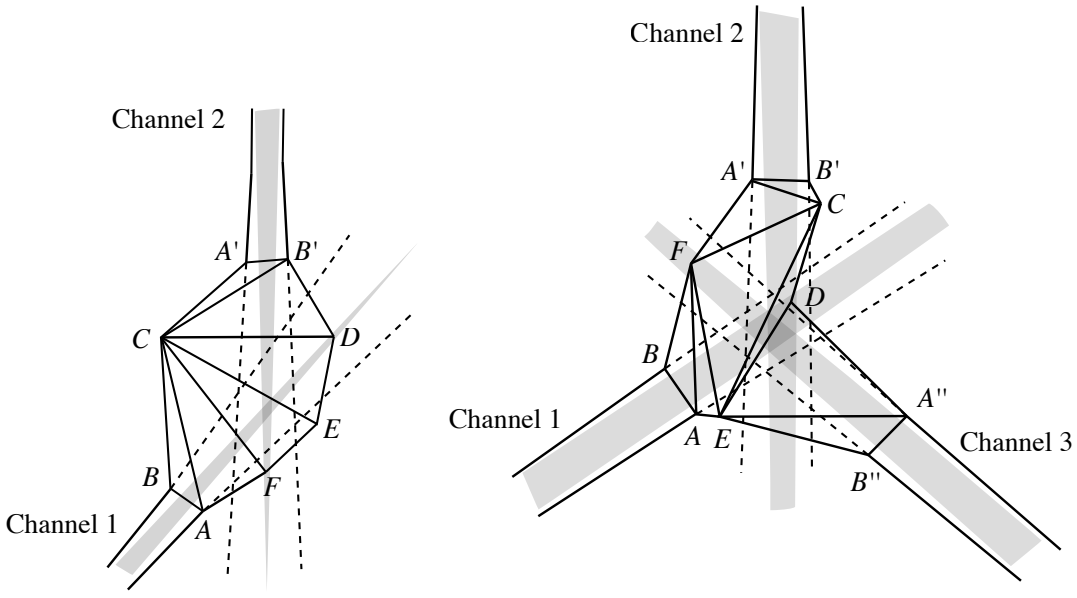
Property 2.4. *Transforming a left-inclined triangulation of a channel to a right-inclined triangulation takes at least 36 flips even in the presence of other vertices, so long as the other vertices lie outside the wide mouths at either end of the channel.*

We now consider what happens when we unlock some vertex gadgets. Let T_1' be the triangulation obtained from T_1 by unlocking some vertex gadgets. Let T_2' be the triangulation obtained from T_2 by unlocking the same vertex gadgets. Let C be the set of channels that have a locked vertex gadget at both ends. Then:

Property 2.5. *If the vertex gadgets at the ends of the channels of C remain locked, then the number of flips required to transform T_1' to T_2' is at least $28|E - C| + 36|C|$.*

Proof. Consider a channel of C, with a locked vertex gadget at both ends. The cap vertices of the channel are not useable. By construction, the other vertices are outside the wide mouths of the channel. Therefore, by Property 2.4, we need 36 flips to transform it.

Consider the channels with an unlocked vertex gadget at one end. We only save flips by capping the channel. To do this, we must flip the two edges that block the channel from its cap. Because the edges that block one channel are disjoint from the edges that block any other channel, we must do two flips per channel, and we must re-flip those edges to return to the original state. Finally, by Property 2.2 it takes at least 24 flips to transform a capped channel. (Note that the proof of Property 2.2 carries over even if the channel is capped at both ends.) The total number of flips is 28 per channel. □



(a) Degree 2 (b) Degree 3

Figure 2.3: Gadgets for vertices

2.2.3 Putting it all together

Lemma 2.1. *G has a vertex cover of size $\leq k$ if and only if the flip distance between the two triangulations T_1 and T_2 is $\leq 2k + 28|E|$.*

Proof. Suppose that G has a vertex cover of size k. Unlock the corresponding k vertex gadgets. Each edge channel has an unlocked gadget at one end, so by Property 2.3 we can transform between the two triangulations of the channel in 28 flips. When all channels have been transformed, we relock the k vertex gadgets. The total number of flips is $2k + 28|E|$.

For the other direction, suppose that there is a flip sequence between T_1 and T_2 of length $\leq 2k + 28|E|$. Let L be the set of vertices whose gadgets are unlocked in the flip sequence. Let C be the set of edges not covered by vertex set L. By adding one vertex to cover each edge of C, we observe that G has a vertex cover of size $|L| + |C|$. Thus it suffices to show that $|L| + |C| \leq k$. By Properties 2.4 and 2.5 the number of flips is at least $2|L| + 36|C| + 28(|E - C|) \geq 2|L| + 28|E| + 8|C|$. By assumption, the number of flips was $\leq 2k + 28|E|$. Therefore $2|L| + 8|C| \leq 2k$, which implies that $|L| + |C| \leq k$, as required. □

The last ingredient of the NP-completeness proof is to show that the reduction takes polynomial time. We need the following claim.

Claim 2.1. *The size of the coordinates used in the construction is bounded by a polynomial in n.*

Proof. We begin with a straight line drawing of G on a polynomial-sized grid. Expand the grid, and allocate a square region around each vertex for the vertex gadget (Figure 2.4). Expand each edge to two parallel line segments. These line segments will become the channel, but for now, the reflex vertices of the channel are all collinear, which means that the channel's wide mouth is equal to its narrow mouth. The points C, D, E, F of the vertex gadget go in *feasible* regions defined by the wide and narrow mouths (e.g. in the 3-channel gadget, point D lies in the narrow mouth of channels 1 and 2, but outside the wide mouth of channel 3). We make the channels narrow enough so that all the feasible regions intersect the region allocated to the gadget.

To do this, note that the edges incident to the vertex corresponding to the gadget in the straight line drawing and their extensions (the dotted lines in Figure 2.4) intersect the square at points whose coordinates have polynomial size. Let S be the set of intersection points and corners of the square. For the edge corresponding to channel 1, consider the

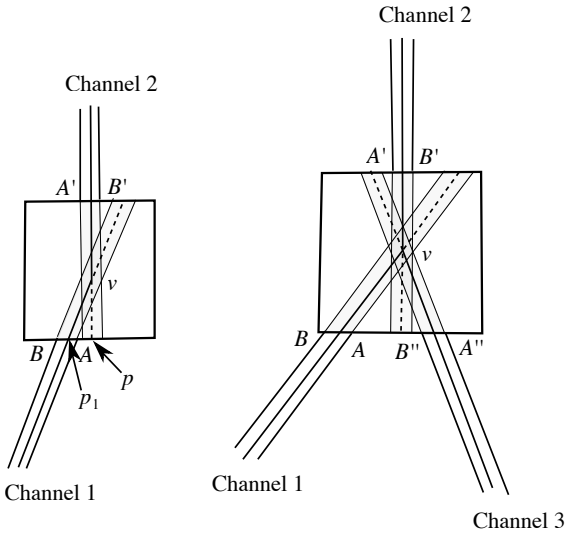
point p_1 where it intersects the square and find the point p other than itself in S that lies on the same edge of the square and is closest to it. Setting A to be the point on the boundary of the square a distance $pp_1/3$ away from p_1 towards p and B the symmetric point on the opposite side determines the channel and its width. Do the same thing at the other end of the edge corresponding to channel 1 and obtain another width. Finally, pick the narrower of the two options for channel 1. Since A and B lie on the edge of the square and their distance to p_1 is polynomial, we need a polynomial number of bits to express the coordinates of A and B as well. Repeat the above for A', B', A'' and B''. Since all the possible intersection points between the upper and lower chains of the channels occur inside the square, all the feasible regions have non-empty intersections with the interior of the square.

Now we pick points C, D, E, F inside the appropriate regions. Because the boundaries of the feasible regions are determined by pairs of points on the expanded grid, the new points can be chosen to have polynomial size (because solutions to linear programs have polynomial size as shown in Theorem 10.1 of [92]).

Finally we place the reflex points of each channel. The feasible region wherein each set of reflex points can be placed is bounded by lines through pairs of points already placed. Thus, we can choose reflex points of polynomial size. □

2.3 Triangulations of point sets

We prove the NP-hardness of computing the flip distance between two triangulations of a point set by reducing from computing the flip distance between two triangulations of a polygonal region. Given two triangulations T_1 and T_2 of a polygonal region R with n vertices, we triangulate all the holes and pockets of R the same way in both triangulations. Next, we repeat each edge on the boundary of the holes and pockets n^2 times (as shown in Figure 2.5). This gives two triangulations T_1' and T_2' of a point set. We will prove that the flip distance between T_1' and T_2' is the same as the flip distance between the original T_1 and T_2. Details are below, but the intuition is as follows. The one danger is that in T_1' and T_2' we have the freedom to flip the repeated edges that model the fixed edges in the original polygonal regions. However, the flip distance between T_1 and T_2 is less than n^2 because the flip distance to the constrained Delaunay triangulation is at most $\binom{n}{2}$ [7]. This means that the flip distance between T_1' and T_2' is also less than n^2. Thus "taking apart" a set of n^2 repeated edges does not offer any savings.



(a) Degree-2 (b) Degree-3

Figure 2.4: Constraints for vertex gadgets.

We now describe the details about repeating the edges. Consider a triangulated hole or a pocket, say, $v_1v_2v_3v_4$. For each vertex on the boundary, draw the angle bisector of the angle formed by the two edges incident on that vertex. Thus at v_2, we draw the angle bisector of $\angle v_1v_2v_3$. Next, we choose $n^2/2$ equally spaced points on a polynomially small portion of the bisector inside the hole. We claim that we can choose points with polynomial-sized coordinates. Next we repeat each edge on the boundary of the hole or pocket n^2 times by adding a zig-zag between the new vertices as shown in Figure 2.5.

Note that flipping edges incident to one of the new vertices will not help because the new vertices behave like the vertex whose angle bisector they were drawn on. This intuition is captured in the following lemma, which then implies the NP-completeness:

Lemma 2.2. *The flip distance between T_1 and T_2 is equal to the flip distance between T_1' and T_2'.*

Proof. The flip distance between T_1' and T_2' is at most the flip distance between T_1 and T_2 because imitating a flip sequence that transforms T_1 to T_2 gives a flip sequence that transforms T_1' to T_2'.

It remains to show that the flip distance between T_1 and T_2 is at most the flip distance between T_1' and T_2'. For each vertex v_i on the boundary of a hole, there is a set of $n^2/2$

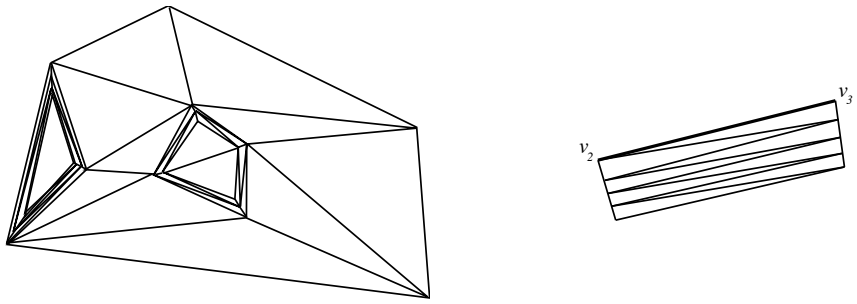


Figure 2.5: Repeating edges on the boundary of pockets and holes. On the right is the detail of how an edge v_2v_3 is repeated n^2 times.

points associated with it. Call the union of the set together with v_i itself the *cluster corresponding to* v_i. Define two triangulations to lie in the same *equivalence class* if they are the same when we collapse each cluster into one point.

Consider the smallest flip sequence that transforms T_1' to T_2'. It has a length of at most n^2. At each step, consider the triangulation obtained by collapsing each cluster into one point. Since the number of flips is at most n^2, we cannot flip all the n^2 repetitions of any edge on the boundary of a hole or pocket. Thus, after collapsing clusters, we have a triangulation that includes all boundary edges of holes and pockets. We can therefore convert the flip sequence to a flip sequence that transforms T_1 to T_2. During the conversion we ignore flips that happen inside holes and pockets. The resulting flip sequence from T_1 to T_2 is not longer than the original. □

Theorem 2.2. *The following problem is NP-complete: Given two triangulations of a point set in the plane, and a number k, is the flip distance between the triangulations at most k?*

2.4 Discussions and open problems

The channel provides an instance of the flip distance problem where the shortest flip sequence flips edges that are common between the two triangulations. In fact, the NP-completeness proof exploits the fact that it is computationally hard to determine exactly which of the common diagonals need to be flipped. It may then appear that the necessity of flipping common diagonals makes the problem hard. We further explore this theme in Chapter 3 for the problem of reconfiguring solutions to Boolean formulas.

Our NP-completeness proof results in problem instances that contain a polynomial number of points strictly inside the convex hull. Reducing this number would take us gradually closer to the case of convex polygons. Incidentally, the independent proof of NP-completeness by Pilz [84] also uses a polynomial number of points strictly inside the convex hull and it is not clear how to reduce it for either of the proofs.

It will also be interesting to design approximation algorithms for the case of point sets and simple polygons. For point sets, there exists an APX-hardness proof [84] but for simple polygons even a PTAS is not ruled out. However, no approximation algorithms are known other than trivial polynomial-factor ones.

Chapter 3

Shortest Reconfiguration Paths in the Solution Space of Boolean Formulas[1]

3.1 Introduction

This chapter presents the first example of the feedback loop between reconfiguration and triangulations that we briefly talked about in Chapter 1. We will not repeat the background on reconfiguration here, since we covered that in Chapter 1. After proving the NP-hardness of computing the flip distance between triangulations of planar point sets and failing to resolve the same question about convex polygons, we ask the obvious next question: What kinds of reconfiguration problems exhibit a polynomial time algorithm for finding reconfiguration distances?

Although this problem had not been explicitly studied in the past, many algorithms for deciding whether a reconfiguration path exists implicitly found the shortest such path. Examples include reconfiguration on 2CNF formulas [47], matchings, and spanning trees [55]. All these cases also happened to satisfy the *symmetric difference property* that the shortest path only changed the parts of the solution that were different in the two target configurations. For example, the shortest reconfiguration path between two satisfying assignments of a 2CNF formula only flips variables that have different values in the two assignments. Similarly, for the case of matchings and spanning trees, the SRP only changes the edges

[1] This chapter represents joint work with Amer Mouawad, Naomi Nishimura, and Venkatesh Raman [75].

that are in the symmetric difference of the two solutions. More recently, some work on the parametrized complexity of reconfiguration problems [76, 77] explored the complexity of computing the reconfiguration distance for some cases all of which adhered to the symmetric difference property.

The pattern we observe in these cases is that the problem has the symmetric difference property (a shortest reconfiguration path does not change parts of the solution that are the same in the source and target configurations) if and only if shortest reconfiguration paths can be computed in polynomial time.

It is known that the symmetric difference property is also satisfied for triangulations of a convex polygon: edges that are common are never flipped by the shortest flip sequence. However, no one knows a polynomial time algorithm to find the flip distance. Triangulations of general point sets do follow the above pattern since computing the flip distance in NP-complete (as shown in Chapter 1) and the symmetric difference property fails: an edge that is in both the initial and final triangulations may need to flip in a shortest reconfiguration path. Eppstein's results [38] on point sets without empty convex pentagons also satisfy the pattern: the symmetric difference property holds and he provided a polynomial time algorithm for computing the flip distance. So is this pattern universal, or is it just a matter of chance that all the problems studied so far happened to obey it?

To be able to make any mathematically precise statements about the universality of something that looks like a pattern to our intuitive senses, we need to tackle reconfiguration problems with more rigour than what we have seen so far. We want to prove or disprove the statement: "Computing the reconfiguration distance is in P if and only if the SRP only flips the symmetric difference". We cannot mathematically reason about such a hypothesis until we have precisely defined what a reconfiguration problem is; otherwise it will be easy to disprove any claim: just construct a new problem where the claim is false and expand the definition of reconfiguration problems to include the new problem.

Our approach is to limit our attention to a large set of problems where reconfiguration already has a precise meaning, namely, reconfiguration of satisfying assignments of Boolean formulas. The size of the set of problems it provides is sufficiently large and should pass all reasonable scrutiny since any decision problem in NP can be written as the satisfiability of Boolean formulas. Moreover, this set provides several natural classes of problems to study that are defined by the corresponding classes of Boolean formulas. Thus, instead of studying patterns in reconfiguration on various different problems such as independent set, graph coloring, graph matching, etc., we study patterns in reconfiguration problems defined on various different classes of Boolean formulas, such as 2CNF formulas, Horn formulas, etc. This line of investigation is appropriate for our goals because of a framework developed

by Schaefer [87] that provides a way to divide the class of Boolean formulas into mutually exclusive and exhaustive classes. Thus we can, at least for reconfiguration on Boolean formulas, make statements about intuitive patterns without having to be mathematically imprecise. In particular, as we will see later, the pattern "Computing the reconfiguration distance is in P if and only if the SRP only flips the symmetric difference" breaks down for Boolean formulas.

3.1.1 Schaefer's framework

The problem of deciding whether a given Boolean formula has a satisfying assignment, also known as SAT, has played an important role in the history of computer science. It was the first problem that was proved to be NP-complete. Moreover, the complexity of SAT, for several subclasses of Boolean formulas is also known. For example, 3SAT is NP-complete, and 2SAT, Horn-SAT, dual-Horn-SAT (as defined in Section 3.2.1) are all in P. Moreover, since systems of linear equations over finite fields can be solved in polynomial time using Gaussian elimination, SAT is also in P for formulas that can be written as the conjunction of several clauses of the form $(x_1 \oplus \ldots \oplus x_k)$ where '$\oplus$' is the XOR operator. In 1978, Schaefer [87] proved that these are the only classes of formulas for which satisfiability is in P, and that for every other class it is NP-complete. This result is interesting in light of Ladner's theorem that states that if P$\neq$NP, then there are an infinite number of problems that are neither in P nor NP-complete.

The proof of Ladner's theorem constructs a subclass of SAT and shows it to be neither in P nor NP-complete under the assumption that P$\neq$NP. The apparent contradiction with Schaefer's result, then, is due to the fact that Schaefer has a restricted way of defining a "class" of formulas. Arguably, Schaefer's definition of a class is natural and the class exhibited by Ladner's theorem does not fit into the framework.

Since Schaefer's original paper, a myriad of problems about Boolean formulas have been analyzed, and similar divisions into equivalence classes obtained (see [28] for an excellent survey). We explain Schaefer's framework below.

A *k-ary Boolean logical relation* (or *relation* for short) R is defined as a subset of $\{0,1\}^k$, where $k \geq 1$. Each $i \in \{1,\ldots,k\}$ can be interpreted as a variable of R such that R specifies exactly which assignments of values to the variables are to be considered satisfying.

Roughly speaking, Schaefer's framework uses relations as a template. A class is then defined by a set of relations and a CNF formula is said to belong to that class if all of its clauses are "instantiations" of the template as defined below.

For any k-ary relation R and positive integer $k' \leq k$, we define a k'-ary *restriction* of R to be any k'-ary relation R' that can be obtained from R by substitution with constants and identification of variables. More precisely, let $X : \{1, \ldots, k\} \to \{1, \ldots, k'\} \cup \{c_0, c_1\}$ be a mapping from the variables of R to the variables of R' and the constants 0 and 1. Any such X defines a mapping $f_X : \{0,1\}^{k'} \to \{0,1\}^k$ as follows. For $r \in \{0,1\}^{k'}$, let $f_X(r)$ be the k-bit vector whose i^{th} bit is 0 if $X(i) = c_0$, 1 if $X(i) = c_1$ and equal to the $X(i)^{\text{th}}$ bit of r otherwise. We say that a k'-ary relation R' is a restriction of R with respect to $X : \{1, \ldots, k\} \to \{1, \ldots, k'\} \cup \{c_0, c_1\}$ if $r \in R' \Leftrightarrow f_X(r) \in R$.

A Boolean formula ϕ over a set $\{x_1, \ldots, x_n\}$ of variables defines a relation R_ϕ as follows. For any n-bit vector $v \in \{0,1\}^n$, we interpret v as the assignment to the variables of ϕ where x_i is set to be equal to the i^{th} bit of v. We then say that $v \in R_\phi$ if and only if v is a satisfying assignment.

A *CNF formula* is a Boolean formula of the form $C_1 \wedge \cdots \wedge C_m$, where each C_i, $1 \leq i \leq m$, is a *clause* consisting of a finite disjunction of *literals* (variables or negated variables). A *kCNF formula*, $k \geq 1$, is a CNF formula where each clause has at most k literals. A CNF formula is *Horn* (*dual Horn*) if each clause has at most one positive (negative) literal.

For a finite set of relations $\mathcal{S}$, a *CNF($\mathcal{S}$) formula* over a set of n variables $\{x_1, \ldots, x_n\}$ is a finite collection $\{C_1, \ldots, C_m\}$ of clauses. Each C_i, $1 \leq i \leq m$, is defined by a tuple (R_i, X_i), where R_i is a k_i-ary relation in $\mathcal{S}$ and $X_i : \{1, \ldots, k_i\} \to \{1, \ldots, n\} \cup \{c_0, c_1\}$ is a function. Each X_i defines a mapping $f_{X_i} : \{0,1\}^n \to \{0,1\}^{k_i}$ and we say that an assignment v to the variables satisfies ϕ if and only if for all $i \in \{1, \ldots, m\}$, $f_{X_i}(v) \in R_i$. For any variable x_j, we say that x_j *appears in* clause C_i if $X_i(q) = j$ for some $q \in \{1, \ldots, k_i\}$ and for any assignment v to the variables of ϕ, we say that $f_{X_i}(v)$ is the assignment induced by v on R_i.

For example, to represent the class 3CNF in Schaefer's framework, we specify $\mathcal{S}$ as follows. Let $R^0 = \{0,1\}^3 \backslash \{000\}$, $R^1 = \{0,1\}^3 \backslash \{100\}$, $R^2 = \{0,1\}^3 \backslash \{110\}$, $R^3 = \{0,1\}^3 \backslash \{111\}$, and $\mathcal{S} = \{R^0, R^1, R^2, R^3\}$. Since R^i can be used to represent all 3-clauses with exactly i negative literals (regardless of the positions in which they appear in a clause), clearly CNF($\mathcal{S}$) is exactly the class of 3CNF formulas.

Consider POSITIVE 1-IN-3CNF, which is the class of CNF formulas where each clause contains a list of three variables and the formula is said to be satisfied if and only if exactly one out of the three variables in each clause is set to 1. We can represent it in Schaefer's framework by CNF($\mathcal{S}$) where $\mathcal{S} = \{\{100, 010, 001\}\}$.

3.2 Computing shortest reconfiguration paths

3.2.1 Preliminaries

Below we define some classes of relations used in the literature and relevant to our work. Note that componentwise bijunctive, OR-free and NAND-free were first defined by Gopalan et al. [47]. Schwerdtfeger [89] later modified them slightly by adding "safely" in front of each. We adopt Schwerdtfeger's definition but drop the word "safely".

Definition 3.1. *For a k-ary relation R:*

- *R is* bijunctive *if it is the set of satisfying assignments of a 2CNF formula.*
- *R is* Horn *(*dual Horn*) if it is the set of satisfying assignments of a Horn (dual Horn) formula.*
- *R is* affine *if it is the set of satisfying assignments of a formula $x_{i_1} \oplus \ldots \oplus x_{i_h} \oplus c$, with $i_1, \ldots, i_h \in \{1, \ldots, k\}$ and $c \in \{0, 1\}$. Here $\oplus$ denote the exclusive OR operation which evaluates to 1 when exactly one of the values it operates on is 1 and evaluates to 0 otherwise.*
- *R is* componentwise bijunctive *if every connected component of the reconfiguration graph of R and of the reconfiguration graph of every restriction R' of R induces a bijunctive relation.*
- *R is* OR-free *(*NAND-free*) if there does not exist a restriction R' of R such that $R' = \{01, 10, 11\}$ ($R' = \{01, 10, 00\}$).*

Using his framework, Schaefer showed that SAT($\mathcal{S}$)—the problem of deciding if a CNF($\mathcal{S}$) formula has a satisfying assignment—is in P if every relation in $\mathcal{S}$ is bijunctive, Horn, dual Horn, or affine, and is NP-complete otherwise.

Gopalan et al.'s work [47], with corrections presented by Schwerdtfeger [89], shows a dichotomy for the problem of deciding whether a reconfiguration path exists between two satisfying assignments of a CNF($\mathcal{S}$) formula.

They call a set $\mathcal{S}$ of relations *tight* if

- all relations in $\mathcal{S}$ are componentwise bijunctive, or
- all relations in $\mathcal{S}$ are OR-free, or

- all relations in $\mathcal{S}$ are NAND-free.

They showed that the st-connectivity problem on CNF($\mathcal{S}$) formulas is in P if $\mathcal{S}$ is tight and PSPACE-complete otherwise.

Our trichotomy relies on a new class of formulas that subdivides the tight classes into those for which computing the shortest reconfiguration path can be done in polynomial time and those for which it is NP-complete.

Definition 3.2. *For a k-ary relation R:*

- *R is* Horn-free *if there does not exist a restriction R' of R such that $R' = \{0,1\}^3 \setminus \{011\}$, or equivalently, R' is the set of all satisfying assignments of the clause $(x \vee \overline{y} \vee \overline{z})$ for some three variables x, y, and z.*
- *R is* dual-Horn-free *if there does not exist a restriction R' of R such that $R' = \{0,1\}^3 \setminus \{100\}$, or equivalently, R' is the set of all satisfying assignments of the clause $(\overline{x} \vee y \vee z)$ for some three variables x, y, and z.*

Definition 3.3. *We call a set $\mathcal{S}$ of relations* navigable *if one of the following holds:*

(1) All relations in $\mathcal{S}$ are OR-free and Horn-free.

(2) All relations in $\mathcal{S}$ are NAND-free and dual-Horn-free.

(3) All relations in $\mathcal{S}$ are component-wise bijunctive.

It is clear that if $\mathcal{S}$ is navigable, then it is also tight. Our main result is the following trichotomy.

Theorem 3.1. *For a CNF($\mathcal{S}$) formula ϕ and two satisfying assignments s and t, the problem of computing the shortest reconfiguration path between s and t is in P if $\mathcal{S}$ is navigable, NP-complete if $\mathcal{S}$ is tight but not navigable and PSPACE-complete otherwise.*

In the next section, we establish the hardness results; the rest of the chapter is devoted to develop our polynomial time algorithm for navigable formulas. Interestingly, unlike previous classification results, while the NP-completeness result in our case turns out to be easier, the polynomial time algorithm is quite involved.

3.2.2 The hardness proofs

Gopalan et al. [47] showed that if $\mathcal{S}$ is not tight, then st-connectivity is PSPACE-complete for CNF($\mathcal{S}$). This implies that finding the shortest reconfiguration path is also PSPACE-complete for such classes of formulas.

Theorem 3.2. *If $\mathcal{S}$ is tight but not navigable, then finding the shortest reconfiguration path on CNF($\mathcal{S}$) formulas is NP-complete.*

Proof. The problem is in NP because the diameter of the reconfiguration graph is polynomial for all tight formulas, as shown by Gopalan et al. [47]. We now prove that it is, in fact, NP-complete.

As $\mathcal{S}$ is tight but not navigable, all relations in $\mathcal{S}$ are OR-free or all relations in $\mathcal{S}$ are NAND-free. Let us assume that all relations in $\mathcal{S}$ are NAND-free (we handle the other case later). Then, as $\mathcal{S}$ is not navigable, there exists a relation which is dual-Horn.

We show a reduction from Vertex Cover to such a CNF($\mathcal{S}$) formula. Given an instance $(G = (V, E), k)$ of Vertex Cover, we create a variable x_v for each $v \in V$. For each edge $e = (u, v) \in E$, we create two new variables y_e and z_e and the clauses $(y_e \vee \overline{z_e} \vee x_u)$ and $(\overline{y_e} \vee z_e \vee x_v)$. The resulting formula $F(G)$ has $|V| + 2|E|$ variables and $2|E|$ clauses.

It is easy to see that all the relations of $F(G)$ are NAND-free (as we cannot set the values of all but two of their variables to get a NAND relation), however none of them is dual-Horn-free (as each clause has two positive literals). Hence the formula $F(G)$ is tight but not navigable.

Let s be the satisfying assignment for the formula with all variables set to 0, and let t be the satisfying assignment with all the variables x_v with $v \in V$ set to 0 and the rest set to 1. If G has a vertex cover S of size at most k, then we can form a reconfiguration sequence of length at most $2|E| + 2k$ from s to t by flipping each x_v with $v \in S$ from 0 to 1, flipping the y_e and z_e variables, and then flipping each $x_v, v \in S$ back from 1 to 0. To show that such a reconfiguration sequence exists only if there exists such a vertex cover, we observe that if neither x_u nor x_v has been flipped to 1, neither y_e nor z_e can be flipped to 1 while keeping the formula satisfied at the intermediate steps.

To show hardness when all relations in $\mathcal{S}$ are OR-free but not Horn-free, we give a reduction from Independent set. Given $G = (V, E)$ and an integer k, we create, as before, a variable x_v for each $v \in V$ and two variables y_e and z_e for each $e \in E$. For each edge $e = (u, v) \in E$, we create the clauses $(y_e \vee \overline{z_e} \vee \overline{x_u})$ and $(\overline{y_e} \vee z_e \vee \overline{x_v})$. Clearly, all the relations of the formula are OR-free, and none of them is Horn-free.

Let s be the satisfying assignment that sets all the variables to 1, and t be the satisfying assignment that sets all the variables to 0 except the variables $x_v, v \in V$ that are set to 1. If G has an independent set of size at least k, then it has a vertex cover S of size at most $n-k$, then we can form a reconfiguration sequence of length at most $2|E| + 2(n-k)$ from s to t by flipping each $x_v, v \in S$ from 1 to 0, flipping the y_e and z_e variables, and then flipping each $x_v, v \in S$ back from 0 to 1. To show that such a reconfiguration sequence exists only if there exists such an independent set (of size $n-k$), we observe that if neither x_u nor x_v has been flipped to 0, neither y_e nor z_e can be flipped to 0 while keeping the formula satisfied at the intermediate steps. □

3.2.3 The polynomial-time algorithm for navigable formulas

In this section, we give the polynomial time algorithm to find the shortest reconfiguration sequence between two satisfying assignments of a navigable formula.

Gopalan et al. [47] gave a polynomial-time algorithm for finding the shortest reconfiguration path in component-wise bijunctive formulas. The path, in this case, flips only variables that have different values in s and t. The NP-completeness proof from the previous section crucially relies on the fact that we need to flip variables with common values; in fact, the hardness lies in deciding precisely which common variables need to be flipped. Thus it is tempting to conjecture that hardness for shortest reconfiguration path is caused by relations where the shortest distance is not always equal to the Hamming distance.

Interestingly, this intuition is wrong. The reconfiguration graph for the relation $P_4 = \{000, 001, 101, 111, 110\}$ is a path of length four, where for 000 and 110 the shortest path is of length four but the Hamming distance is two. However, we can find shortest reconfiguration paths for formulas built out of R in polynomial time, the exact reason for which will become clear in our general description of the algorithm. The intuitive reason is that there are very few candidates for shortest paths; if we restrict our attention to a single clause built out of R, then there exists a unique path to follow. It then suffices to determine whether there exist two clauses for which the prescribed paths are in conflict. In general, our proof relies on showing that even if there does not exist a unique path, the set of all possible paths between two satisfying assignments of a navigable formula is not diverse enough to make the problem computationally hard. We show that the set of all possible paths can be characterized using a partial order on the set of flips.

Notation

Our results make use of two different views of the problem (graph theoretic and algebraic), and hence two sets of notation.

The graph-theoretic view consists of the reconfiguration graph G_R that has a node for each Boolean string $s \in R$ and an edge whenever the Hamming distance between the two strings is exactly one. We call a path from s to t *monotonically increasing* if the Hamming weights of the vertices on the path increase monotonically as we go from s to t, and define a *monotonically decreasing* path similarly. A path is *canonical* if it consists of a monotonically increasing path followed by a monotonically decreasing path.

The algebraic view consists of a finite state machine, or a *token system* [39], representing the graph G_R. The token system consists of a set $\mathscr{S}$ of states and a set τ of tokens. The tokens specify the rules of transition between states. Each token $t \in \tau$ is a function that maps $\mathscr{S}$ to itself. Given a k-ary relation R, we define a token system for it as follows. The set $\mathscr{S}$ of states consists of all the elements of R and a special state s^* called the *invalid state*. The set τ of tokens is the set $\{x_1^+, \ldots, x_k^+\} \cup \{x_1^-, \ldots, x_k^-\}$, where x_i^+ denotes a flip of variable x_i from 0 to 1, which we call a *positive flip*, and x_i^- denotes a flip of variable x_i from 1 to 0, which we call a *negative flip*.

To complete the description of the token system, we need to specify the function to which each token corresponds. For $x_i^+ \in \tau$ and $s \in \mathscr{S}$,

- $x_i^+(s^*) = s^*$ for all i.
- $x_i^+(s) = s^*$ if the value of variable x_i in s is already 1.
- $x_i^+(s) = s'$ if the value of variable x_i in s is 0 and the bit string s' obtained on flipping it to 1 lies in R.
- $x_i^+(s) = s^*$ if the value of variable x_i in s is 0 and the bit string s' obtained on flipping it to 1 does not lie in R.

The function x_i^- is defined analogously. In the rest of this article, we will use the word "flip" instead of "token", and we will use the words "state," "vertex," and "satisfying assignment" (for all states except s^*) interchangeably.

A sequence of flips also defines a function, that is, the composition of all the functions in the sequence. We call a flip sequence *invalid* at a given state s if the sequence applied to s results in invalid state s^*, and *valid* otherwise. Starting from a state s, two flip sequences

are said to be *equivalent* if they result in the same final state when applied to s. Finally, we call a flip sequence *canonical* if all positive flips in it occur before all the negative flips. That is, the path from its first state (node) to the last is a canonical path. Note that in any valid canonical flip sequence, each flip occurs at most once. Given two states $s, t \in \mathscr{S}$, we say that a set $\mathscr{C}$ of flips *transforms s to t* if the elements of $\mathscr{C}$ can be arranged in some order such that the resulting flip sequence transforms s to t. For a given state s and flip set $\mathscr{C}$, we say $\mathscr{C}$ is *valid* if the elements of $\mathscr{C}$ can be arranged in some order such that the resulting flip sequence applied to s results in a valid state.

We will work with sequences of flips. If $\mathcal{F} = f_1, \ldots, f_q$ is a sequence of flips, we write $\mathcal{F}(s)$ for the state that results from applying f_1, then $f_2, \ldots$, then f_q to state s, i.e. $\mathcal{F}(s) = f_q(f_{q-1}(\cdots f_1(s)) \cdots)$. Note that this is well-defined. The flip sequence formed by removing flip f from $\mathcal{F}$ is denoted $\mathcal{F} \setminus f$. The flip sequence obtained by reversing $\mathcal{F}$ is $\mathcal{F}^{-1}$, and by performing $\mathcal{F}_1$ followed by $\mathcal{F}_2$ is $\mathcal{F}_1 \cdot \mathcal{F}_2$. We use $\mathscr{C}(\mathcal{F})$ to denote the set of flips that appear in $\mathcal{F}$. A flip sequence (set) consisting of only positive flips will be called a *positive flip sequence (set)*. We use $\mathcal{F}_0$ to denote an empty flip sequence and, by convention, define it to be valid for all states except s^*. For a flip sequence $\mathcal{F}$, if $f \in \mathcal{F}$ appears before $f' \in \mathcal{F}$ in the sequence, then we say $f <_{\mathcal{F}} f'$. For a tuple $t = (x_{i_1}, \ldots, x_{i_d})$ of variables and a state s, we use s^t to denote the string of values restricted to $x_{i_1}, \ldots, x_{i_d}$.

We will repeatedly use the following observation.

Observation 3.1. *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two positive valid flip sequences starting at s such that $\mathscr{C}(\mathcal{F}_2) = \mathscr{C}(\mathcal{F}_1) \setminus \{x^+\}$. Then x^+ is a valid flip at $\mathcal{F}_2(s)$.*

Proof. Clearly, the Hamming distance between $\mathcal{F}_1(s)$ and $\mathcal{F}_2(s)$ is 1, the value of x at $\mathcal{F}_2(s)$ is 0, and the value of x at $\mathcal{F}_1(s)$ is 1. □

Overview of the algorithm

Consider, once again, the relation $P_4 = \{000, 001, 101, 111, 110\}$ from Section 3.2.3, which we claimed to be navigable. A satisfying assignment to the formula induces, on each clause, a boolean string that consists of the values of the variables appearing in that clause. Similarly, a flip sequence $\mathcal{F}$ induces a flip sequence for each clause C, which is the subsequence of $\mathcal{F}$ that flips a variable that appears in C. Note that $\mathcal{F}$ is valid if and only if the sequence induced on each clause is valid.

The relation P_4 satisfies two nice properties:

1. Any valid flip sequence of P_4 is canonical.

2. Let x, y, z be the three variables that represent the three bits of P_4, then there is a total order, namely, $z^+ < x^+ < y^+$, such that any valid positive flip sequence must satisfy this order.

With this observation, formulating an algorithm is easy. Given two satisfying assignments s and t of the formula, find the Boolean string induced on each clause. The shortest flip sequence inside each clause can be computed in constant time. Each clause prescribes a unique order in which the flips of its corresponding flip sequence must be performed. If no two clauses prescribe conflicting orders, then their sequences can be combined into a sequence for the entire formula. If there is a conflict, then we know that no path exists. In general, for navigable formulas, we show, in Lemma 3.2, that any flip sequence can be transformed into an equivalent canonical flip sequence by rearranging the flips, and, in Lemma 3.9, that each clause prescribes a partial order instead of a total order. The task of combining the partial orders prescribed by each clause becomes more involved, but can still be done efficiently, as shown in Lemma 3.10.

The token system of NAND-free relations

We begin by proving some useful properties of the token system formed by NAND-free relations. We will first show that valid flip sequences can be made canonical, by moving the positive flips ahead of the negative flips.

Lemma 3.1. *Let R be a NAND-free relation and $\mathcal{F} = f_1 \dots f_q$ be a valid flip sequence at $s \in R$. If there exists $i \in \{1, \dots, q-1\}$ such that $f_i = x^-$ is a negative flip and $f_{i+1} = y^+$ is a positive flip, with $x \neq y$, then the sequence $\mathcal{F}' = f_1 \dots f_{i-1} f_{i+1} f_i \dots f_q$ is also valid at s and is equivalent to $\mathcal{F}$, i.e., swapping f_i and f_{i+1} results in an equivalent flip sequence.*

Proof. Let u be the state right before applying f_i in $\mathcal{F}$, $v = f_i(u)$ be the state after applying f_i but before applying f_{i+1}, and $w = f_{i+1}(v)$ be the one after applying f_{i+1}. Thus it is clear that $u^{(x,y)} = 10, v^{(x,y)} = 00$, and $w^{(x,y)} = 01$. Also, notice that since no other variables are flipped between u, v, and w, the values of all variables other than x and y remain the same in the states u, v and w. Let t be the Boolean string whose value is the same as u, v, and w on all variables except x and y and $t^{(x,y)} = 11$. If $t \notin R$, then the substitution described above gives us the relation $\{10, 00, 01\}$ on x and y, which is precisely the NAND relation. Since R is NAND-free, $t \in R$ (Figure 3.1 (a)) and thus we can replace the path $u \to v \to w$ with the path $u \to t \to w$. This is equivalent to swapping the flips f_{i+1} and f_i. □

Lemma 3.2 now follows immediately. It shows (first proved by Gopalan et al. [47]) that any valid flip sequence can be made canonical.

Lemma 3.2. *For R a NAND-free relation, if $\mathcal{F}$ is a valid sequence at $s \in R$, then there exists a valid canonical sequence $\mathcal{F}'$ equivalent to $\mathcal{F}$ such that $\mathscr{C}(\mathcal{F}') \subseteq \mathscr{C}(\mathcal{F})$ and, for any two flips $f_1, f_2 \in \mathcal{F}'$ of the same sign, if $f_1 <_{\mathcal{F}'} f_2$ then $f_1 <_{\mathcal{F}} f_2$, i.e., the relative order among flips of the same sign is preserved.*

Proof. If $\mathcal{F}$ is not canonical, it must have a negative flip followed by a positive flip somewhere. If both flips act on the same variable, we cancel them out; otherwise, we swap them using the proof of Lemma 3.1. Doing this repeatedly gives us the required canonical sequence $\mathcal{F}'$. The order among the flips of the same sign is preserved since we never swap two flips of the same sign. □

Lemma 3.3. *For R a NAND-free relation, if $\mathscr{C}_1$ and $\mathscr{C}_2$ are two positive flip sets that are valid at $s \in R$, then $\mathscr{C}_1 \cup \mathscr{C}_2$ is also a valid flip set at s.*

Proof. Let $u = \mathcal{F}_1(s)$ and $v = \mathcal{F}_2(s)$, where $\mathcal{F}_1$ and $\mathcal{F}_2$ are valid flip sequences such that $\mathscr{C}(F_1) = \mathscr{C}_1$ and $\mathscr{C}(F_2) = \mathscr{C}_2$. Clearly, $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ is a valid flip sequence from u to v. Thus, we can apply Lemma 3.2 to the sequence $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ to transform it into the canonical sequence $\mathcal{F}$. Let $\mathcal{F}^+$ denote the prefix of $\mathcal{F}$ that contains all the positive flips. It is clear that $\mathcal{F}_1 \cdot \mathcal{F}^+$ is a valid flip sequence at s. We can also see, from the proof of Lemma 3.2, that $\mathscr{C}(\mathcal{F}_1 \cdot \mathcal{F}^+) = \mathscr{C}_1 \cup \mathscr{C}_2$. Each element of $\mathcal{F}^+$ must be an element of $\mathcal{F}_2$. If an element of $\mathcal{F}_2$ is not an element of $\mathcal{F}^+$, then it must be because it got cancelled out with an element of $\mathcal{F}_1$, in which case, the element must be in $\mathcal{F}_1$. □

Later, we prove a similar lemma for the intersection of two flip sets, but for dual-Horn-free relations.

Lemma 3.4. *For R a NAND-free relation and $\mathcal{F}_1$ and $\mathcal{F}_2$ two positive flip sequences that are valid at $s \in R$, if $\mathscr{C}(\mathcal{F}_1) \cap \mathscr{C}(\mathcal{F}_2) = \emptyset$, then $\mathcal{F}_1$ is valid at $\mathcal{F}_2(s)$ and $\mathcal{F}_2$ is valid at $\mathcal{F}_1(s)$.*

Proof. Consider the sequence $\mathcal{F}_2^{-1} \cdot \mathcal{F}_1$ that transforms $\mathcal{F}_2(s)$ to $\mathcal{F}_1(s)$. Applying Lemma 3.2 to it, we obtain the canonical flip sequence $\mathcal{F}_1 \cdot \mathcal{F}_2^{-1}$. Thus $\mathcal{F}_1$ is valid at $\mathcal{F}_2(s)$. Using the same argument on the sequence $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ proves the other claim. □

The token system of dual-Horn-free relations

In this section, we establish stronger properties with the assumption that R is not only NAND-free, but is also dual-Horn-free. We begin by establishing a simple property of relations that are NAND-free and dual-Horn-free in the following lemma.

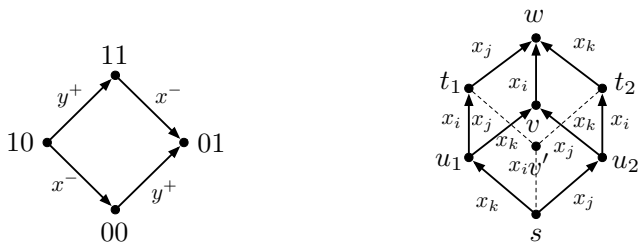


Figure 3.1: (a) Example for Lemma 3.1 (b) Example for Lemma 3.5

Lemma 3.5. *Let R be a NAND-free and dual-Horn-free relation and $s, t_1, t_2 \in R$ be three distinct states such that the flip sequence $\mathcal{F}_1 = x_k^+ x_i^+$ transforms s to t_1, the flip sequence $\mathcal{F}_2 = x_j^+ x_i^+$ transforms s to t_2, and $x_k \neq x_j$. Then the sequence $\mathcal{F}_1' = x_i^+ x_k^+$ also transforms s to t_1 and the sequence $\mathcal{F}_2' = x_i^+ x_j^+$ also transforms s to t_2, i.e., we can swap the flips in both $\mathcal{F}_1$ and $\mathcal{F}_2$.*

Proof. For $u_1 = x_k^+(s)$ and $u_2 = x_j^+(s)$, the sequence $x_j^- x_k^+$ transforms u_2 to u_1. We can reorder the sequence to obtain $x_k^+ x_j^-$, using Lemma 3.1. For $v = x_k^+(u_2)$, we can use a similar argument to show that x_i^+ is a valid flip at v; we let $w = x_i^+(v)$. The values of variables x_i, x_j, and x_k at states s, u_1, u_2, t_1, t_2, v, and w form exactly the seven satisfying assignments $\{000, 001, 010, 101, 110, 011, 111\}$ of the dual-Horn clause $(\overline{x_i} \vee x_j \vee x_k)$ (Figure 3.1 (b)). But since R is dual-Horn-free, there must also exist the state v' for which $x_i = 1, x_j = 0, x_k = 0$. The path $s \to v' \to t_1$ gives the sequence $x_i^+ x_k^+$ and the path $s \to v' \to t_2$ gives the sequence $x_i^+ x_j^+$. □

The seemingly innocuous lemma above turns out to be very powerful. In the following sequence of lemmas, we build on top of it to eventually prove that the set of all positive valid flip sets starting from an assignment s forms a distributive lattice. The lattice structure then helps us formulate a polynomial time algorithm for computing the shortest reconfiguration path.

Lemma 3.6. *Let R be a NAND-free and dual-Horn-free relation and $s, t \in R$ be two satisfying assignments such that $x^+ y^+$ is a valid flip sequence at s and y^+ is a valid flip at t. Furthermore, let $\mathcal{F}$ be a positive flip sequence such that $\mathcal{F}(s) = t$ and $x^+ \notin \mathscr{C}(\mathcal{F})$. Then, the sequence $y^+ x^+$ must also be valid at s.*

Proof. Let v be the vertex with smallest Hamming weight on the path corresponding to $\mathcal{F}$ from s to t (including s and t) at which y^+ is a valid flip. Let $\mathcal{F}_1 = x^+ y^+$ and let $\mathcal{F}_2$ be the

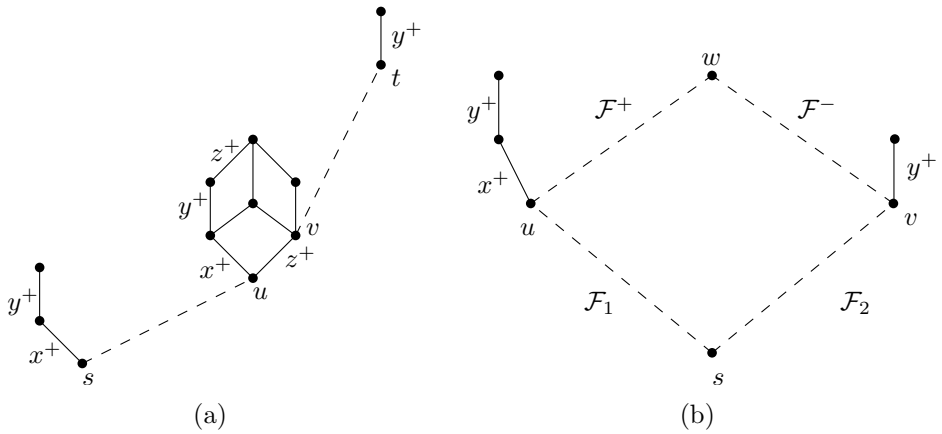


Figure 3.2: Dotted lines denote paths and solid lines denote edges. Hamming weight increases in the upward direction. (a) Proof of Lemma 3.6 (b) Proof of Lemma 3.7.

positive flip sequence that transforms s to v, i.e. $v = \mathcal{F}_2(s)$. Note that $\mathscr{C}(\mathcal{F}_1) \cap \mathscr{C}(\mathcal{F}_2) = \emptyset$, as neither x^+ nor y^+ can appear in $\mathscr{C}(\mathcal{F}_2)$ (See Figure 3.2(a)). If $v = s$, we are done (using Observation 3.1); then let us assume this not to be the case. Let u be the vertex immediately before v on the path from s to t and let $z^+(u) = v$. Since $\mathscr{C}(\mathcal{F}_1) \cap \mathscr{C}(\mathcal{F}_2) = \emptyset$ and $\mathscr{C}(\mathcal{F}_1) \cap \{\mathscr{C}(\mathcal{F}_2) \setminus \{z^+\}\} = \emptyset$, we can apply Lemma 3.4 at s, which implies that x^+y^+ must be valid at both u and v. Now we use Lemma 3.5 at u. Since both x^+y^+ and z^+y^+ are valid sequences at u, y^+x^+ must also be a valid sequence at u. This contradicts the assumption that v was the vertex with smallest Hamming weight on the path where y^+ was a valid flip. □

Lemma 3.7. *Let R be a NAND-free and dual-Horn-free relation. If $\mathcal{F}_1 \cdot x^+ \cdot y^+$ and $\mathcal{F}_2 \cdot y^+$ are both valid positive flip sequences at $s \in R$ such that $x^+ \notin \mathscr{C}(\mathcal{F}_2)$ then $\mathcal{F}_1 \cdot y^+ \cdot x^+$ is also valid at s.*

Proof. Let $u = \mathcal{F}_1(s)$ and $v = \mathcal{F}_2(s)$. We apply Lemma 3.2 to the sequence $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ that transforms u to v to obtain the canonical sequence $\mathcal{F} = \mathcal{F}^+ \cdot \mathcal{F}^-$. Let w be the vertex with maximum Hamming weight on this canonical path (Figure 3.2(b)). Hence, we have $w = \mathcal{F}^+(u)$ and $v = \mathcal{F}^-(w)$. Note that $\mathcal{F}$ does not involve flips of the variables x or y.

Since y^+ is a valid flip at v, $y^+ \notin \mathscr{C}(\mathcal{F}^-)$, and the path from v to w is monotonically increasing, from Lemma 3.4, y^+ is also valid at w. Now using Lemma 3.6, since x^+y^+ is valid at u, $x^+ \notin \mathcal{F}^+$, and y^+ is valid at w, we have that y^+x^+ is also valid at u. □

Lemma 3.3 already shows that the set of valid flip sets is closed under union. To prove

that the set of valid flip sets forms a distributive lattice, we need to show that it is also closed under intersection, which we do in the next lemma.

Lemma 3.8. *Let R be a NAND-free and dual-Horn-free relation. If $\mathscr{C}_1$ and $\mathscr{C}_2$ are two positive flip sets that are valid at $s \in R$, then $\mathscr{C}_1 \cap \mathscr{C}_2$ is also a valid flip set at s.*

Proof. If $\mathscr{C}_1 \subseteq \mathscr{C}_2$ or $\mathscr{C}_2 \subseteq \mathscr{C}_1$, then the statement is trivial. Otherwise, consider any valid ordering $\mathcal{F}_1$ of $\mathscr{C}_1$. We show that if x^+ and y^+ are two consecutive elements of $\mathcal{F}_1$ such that $x^+ \in \mathscr{C}_1 \backslash \mathscr{C}_2$, $y^+ \in \mathscr{C}_1 \cap \mathscr{C}_2$ and $x^+ <_{\mathcal{F}_1} y^+$, then swapping x^+ and y^+ also gives a valid ordering of $\mathscr{C}_1$. Applying such swaps repeatedly, we get an ordering where all elements of $\mathscr{C}_1 \cap \mathscr{C}_2$ appear before all elements of $\mathscr{C}_1 \backslash \mathscr{C}_2$ thus proving that $\mathscr{C}_1 \cap \mathscr{C}_2$ is a valid set at s.

To see how to swap x^+ and y^+ in $\mathcal{F}_1$, suppose u is the vertex on the path corresponding to $\mathcal{F}_1$ on which the sequence $x^+ \cdot y^+$ is performed, and consider an arbitrary valid ordering $\mathcal{F}_2$ of $\mathscr{C}_2$. Let v be the vertex on the path corresponding to $\mathcal{F}_2$ on which y^+ is performed. Such a vertex exists since $y^+ \in \mathscr{C}_1 \cap \mathscr{C}_2$. Now, since $x^+ \cdot y^+$ is valid at u, y^+ is valid at v and the monotonically increasing path from s to v does not contain the flip x^+ (since $x^+ \in \mathscr{C}_1 \backslash \mathscr{C}_2$), applying Lemma 3.7, we can swap y^+ and x^+ in $\mathcal{F}_1$. □

The above lemma, combined with Lemma 3.3, shows that the set of valid flip sets starting at s forms a distributive lattice [9]. Using Birkhoff's representation theorem [9] on it directly implies the next lemma. However, for clarity, we also provide an independent proof.

Let $\prec$ be a partial order defined on a set $\mathscr{C}$ of flips. We say a set $\mathscr{C}' \subseteq \mathscr{C}$ is *downward closed* if for every $x, y \in \mathscr{C}$, $y \in \mathscr{C}' \wedge x \prec y \implies x \in \mathscr{C}'$. We say that an ordering $\mathcal{F}$ of a subset of elements in $\mathscr{C}$ *obeys* the partial order $\prec$ if (i) $\mathscr{C}(\mathcal{F})$ is downward closed and (ii) for every $x, y \in \mathcal{F}$, $x \prec y \implies x <_{\mathcal{F}} y$.

Lemma 3.9. *Let R be a NAND-free and dual-Horn-free relation and s be an element of R. Let $\mathscr{P} = \{x^+ \mid \exists$ a positive flip set $\mathscr{C}$ at s for which $x^+ \in \mathscr{C}\}$. Then there exists a partial order $\prec$ on $\mathscr{P}$ such that any positive flip sequence $\mathcal{F}$ consisting of a subset of $\mathscr{P}$ is a valid flip sequence at s if and only if it obeys the partial order $\prec$.*

Proof. Our proof proceeds by providing an explicit partial order $\prec$ on the flips in $\mathscr{P}$. For $x^+, y^+ \in \mathscr{P}$, let $x^+ \prec y^+$ if all valid positive flip sequences $\mathcal{F}$ starting at s that contain y^+ also contain x^+ and $x^+ <_{\mathcal{F}} y^+$. This is clearly a partial order since if $x^+ \prec y^+$ and $y^+ \prec z^+$ then $x^+ \prec z^+$.

From the definition of the partial order, it is clear that every valid positive flip set must obey the partial order. For the other direction, consider a positive flip sequence $\mathcal{F}^*$ that

obeys the partial order. We will show that $\mathcal{F}^*$ is valid by induction on the length of the flip sequence.

For the base case, $\mathcal{F}^*$ is trivially valid when $|\mathcal{F}^*| = 0$. As the induction hypothesis, suppose that any positive flip sequence of length $i-1$ that satisfies the partial order is valid. Consider the positive flip sequence $\mathcal{F}^* = (f_1, \ldots, f_i)$ that satisfies the partial order, and let $\mathcal{F}_{i-1} = (f_1, \ldots, f_{i-1})$. Let $\mathcal{X}$ be the set of all positive flip sequences valid at s whose last element is f_i. Consider the set $\mathscr{C} = \bigcap_{\mathcal{F} \in \mathcal{X}} \mathscr{C}(\mathcal{F})$. Since $\mathcal{F}^*$ satisfies the partial order, $\mathscr{C} \subseteq \mathscr{C}(\mathcal{F}^*)$. To see why, suppose that $\mathscr{C}$ has an element x^+ that is not there in $\mathscr{C}(\mathcal{F}^*)$. That would mean that x^+ appears before f_i in all valid positive sequences starting at s. But then $x^+ \prec f_i$ and the sequence $\mathcal{F}^*$ does not obey the partial order. Thus using Lemma 3.8, we know that $\mathscr{C}$ is a valid flip set. Since $\mathscr{C}(\mathcal{F}_{i-1})$ is also a valid flip set (from the induction hypothesis), from Lemma 3.3 we know that $\mathscr{C} \cup \mathscr{C}(\mathcal{F}_{i-1}) = \mathscr{C}(\mathcal{F}_{i-1}) \cup \{f_i\} = \mathscr{C}(\mathcal{F}^*)$ (since $\mathscr{C} \subseteq \mathscr{C}(\mathcal{F}^*)$) is a valid flip set. Since $\mathscr{C}(\mathcal{F}_{i-1})$ and $\mathscr{C}(\mathcal{F}^*)$ are both valid flip sets and $\mathscr{C}(\mathcal{F}^*) \setminus \mathscr{C}(\mathcal{F}_{i-1}) = f_i$, $\mathcal{F}^*$ must be a valid flip sequence (from Observation 3.1). □

Efficiently computing the shortest reconfiguration path

We are now ready to provide a polynomial-time algorithm for finding shortest reconfiguration paths in CNF($\mathcal{S}$) formulas where $\mathcal{S}$ is navigable. If every relation in $\mathcal{S}$ is componentwise bijunctive, we use Gopalan et al.'s algorithm. Otherwise, we assume that every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free since the dual case of every relation being OR-free and Horn-free can be handled similarly.

Let ϕ be a CNF($\mathcal{S}$) formula where every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free, $\{x_1, \ldots, x_n\}$ be the set of variables, and $\{C_1, \ldots, C_m\}$ be the set of clauses in ϕ. The formula ϕ defines a relation R_ϕ as described in Section 3.1.1. Let G_ϕ denote the reconfiguration graph corresponding to R_ϕ. We wish to compute the shortest reconfiguration path between s and t in G_ϕ for $s, t \in R_\phi$. Let $\mathscr{P}_s$ and $\mathscr{P}_t$ be the sets of positive flips that occur in any positive flip set valid at s and t, respectively. That is, $\mathscr{P}_s = \{f \mid f$ is a positive flip and $\exists$ a valid flip sequence in R_ϕ containing f and starting at $s\}$. We define $\mathscr{P}_t$ similarly.

The following lemma helps us combine the partial orders for each clause into one partial order for the formula.

Lemma 3.10. *Let ϕ be a CNF($\mathcal{S}$) formula where every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free. For any $s \in R_\phi$, there exists a partial order $\prec_s$ on $\mathscr{P}_s$ such that any positive flip sequence $\mathcal{F}_s$ consisting of a subset of $\mathscr{P}_s$ is a valid flip sequence at s if and only if it obeys the partial order $\prec_s$. Moreover, $\mathscr{P}_s$ and $\prec_s$ can be computed in time polynomial in the size of the formula.*

Proof. We compute $\mathscr{P}_s$ and $\prec_s$ using a directed graph G_s which we construct in two phases. In the first phase, we add vertices and edges, and in the second phase, we delete some vertices and edges.

We define $\mathscr{P} = \{x^+ \mid \exists$ a positive valid flip set containing x^+ at s for some relation in $\mathcal{S}\}$ and let G_s contain a node for each flip in $\mathscr{P}$. The assignment s induces an assignment $f_{X_j}(s)$ on clause $C_j = (R_j, X_j)$ and Lemma 3.9 defines a partial order $\prec_s^j$ that characterizes the valid positive sequences in R_j starting at $f_{X_j}(s)$. For all $p, q \in \{1, \ldots, k_j\}$ such that $p^+ \prec_s^j q^+$, if $X_j(p) \notin \{c_0, c_1\}, X_j(q) \notin \{c_0, c_1\}$ and $X_j(p) \neq X_j(q)$, we add the directed edge $(x^+_{X_j(p)}, x^+_{X_j(q)})$ to G_s. We do this for each clause C_j for $j \in \{1, \ldots, m\}$. This gives us G_s and marks the end of the first phase.

Now, for the second phase, if the vertex in G_s corresponding to a flip f lies on a cycle of G_s or is reachable by a directed path from a vertex that lies on a cycle of G_s, then f cannot lie in a valid positive flip sequence from s. Hence we remove these vertices from G_s as follows. First, any vertex that appears on a directed cycle is marked to be removed. Then, we iteratively mark every vertex that has an incoming edge from a marked vertex. Once the set of marked vertices stops changing, we remove all marked vertices. Let G'_s be the graph thus obtained. Note that G'_s is acyclic.

We claim that $\mathscr{P}_s = V(G'_s)$, and the partial order $\prec_s$ is such that $f_1 \prec_s f_2$ if and only if there is a directed path from f_1 to f_2 in G'_s. It is clear that any vertex that was removed in the second phase cannot be a part of any valid flip sequence at s. To see that $\prec_s$ is the required partial order, it is enough to see that any flip sequence is valid for ϕ if and only if it is valid for each clause.

Computing the partial orders defined by Lemma 3.9 can be accomplished in constant time for each relation in $\mathcal{S}$. Then, the construction and deletion phases for G_s can be accomplished in polynomial time as described above. □

For a set $\mathscr{P}$, a partial order $\prec$ on $\mathscr{P}$, and a subset $A \subseteq \mathscr{P}$, the *smallest lower set* of A is the smallest superset of A that is downward closed. Such a lower set can be constructed in time polynomial in the number of elements of $\prec$ by starting with A and including any element f' not in A such that $f' \prec f$ for some $f \in A$. It is clear that any valid flip set that contains A must also contain the smallest lower set of A.

Now the algorithm for finding the shortest reconfiguration path is clear. We start from s and let S be the set of positive flips on the variables that are set to 1 in t and to 0 in s. Then we compute the smallest lower set S' containing S and perform the flips in S' as prescribed by the partial order $\prec_s$ (on $\mathscr{P}_s$) to reach $s' \in R_\phi$. We perform a similar set of

flips starting from t to reach $t' \in R_\phi$. If $s' = t'$, we are done. Otherwise, we recursively find the shortest path between s' and t'. The complete algorithm is described in Algorithm 1.

Algorithm 1 ShortestPath(s,t)

Require: A CNF($\mathcal{S}$) formula ϕ where all relations in $\mathcal{S}$ are NAND-free and dual-Horn-free; two satisfying assignments s and t.
Ensure: Shortest reconfiguration path between s and t.

```
1: if (s = t) then
2:    return F_0 {the empty flip sequence}
3: end if
4: Let S be the set of positive flips that flip variables assigned 0 in s and 1 in t.
5: Let T be the set of positive flips that flip variables assigned 0 in t and 1 in s.
6: if S contains an element not in P_s or if T contains an element not in P_t then
7:    return Not connected.
8: end if
9: Compute the smallest lower set S' of S in P_s with respect to ≺_s.
10: Compute the smallest lower set T' of T in P_t with respect to ≺_t.
11: Let F_s and F_t be orderings of S' and T' that obey ≺_s and ≺_t, respectively.
12: Let s' = F_s(s) and t' = F_t(t).
13: Let F = ShortestPath(s',t').
14: return F_s · F · F_t^{-1}.
```

We are now ready to prove the following theorem.

Theorem 3.3. *Let $\mathcal{S}$ be a navigable set of relations, ϕ be a CNF($\mathcal{S}$) formula, and s and t two of its satisfying assignments. We can compute the shortest reconfiguration path between s and t in polynomial time.*

Proof. We show that Algorithm 1 finds the shortest path between s and t, and runs in polynomial time. We first address the running time.

For any Boolean vector x, let $\eta(x)$ denote the number of 0's in x and let $\eta = \eta(s)+\eta(t)$. It is clear that Steps 1 to 10 take time polynomial in the input size N, where $N = |\phi| + |\mathcal{S}| + |s| + |t|$. Here $|x|$ denotes the number of bits needed to represent x. Since $\mathcal{F}_s$ and $\mathcal{F}_t$ are both positive flip sequences, $\eta(s') + \eta(t') \leq \eta(s) + \eta(t) - 2$. Thus the running time $T(\eta)$ of the algorithm satisfies the recursive inequality $T(\eta) \leq T(\eta-2)+P(N)$ where $P(N)$ is some polynomial in N. Since $\eta < N$ the recursion solves to a polynomial in N.

Finally, we prove the correctness of the algorithm. We use induction on η. If $\eta = 0$, then $s = t$ and the algorithm is trivially correct.

If the algorithm returns "Not connected", then it is either because of Step 6 or Step 11. If it is because of Step 11, then by the induction hypothesis s' and t' are not connected, and thus s and t are also not connected. Any flip sequence that transforms s to t must perform each flip in S. Thus it is also clear that if Step 6 returns "Not connected", then s and t are not connected.

If the algorithm returns a flip sequence, then we claim that it is a shortest sequence. From induction, we know that $\mathcal{F}$ is a shortest flip sequence from s' to t'. The claim follows from the observation that if s and t are connected, then there must exist a shortest path from s to t that passes through both s' and t'. Let $\mathcal{F}_1 \cdot \mathcal{F}_2^{-1}$ be a shortest flip sequence from s to t such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are both positive. It is clear that $S' \subseteq \mathscr{C}(\mathcal{F}_1)$. Since S' itself is valid, from Lemma 3.10, there must exist a valid ordering of $\mathscr{C}(\mathcal{F}_1)$ that first performs all flips of S'. In this ordering, the vertex reached after performing all flips of S' is exactly s'. Using a similar argument on $\mathcal{F}_2$, we get a shortest path that goes through both s' and t'. □

3.3 Final remarks

Many problems can be modelled as finding shortest paths in large graphs. Our result provides new insights into the kinds of structures a graph will need to possess to be amenable to an efficient shortest path algorithm. The fact that the shortest path in navigable formulas flips variables that are not in the symmetric difference is evidence that our algorithm exploits a property of the reconfiguration graph that is fundamentally new. Any previously known properties that were used to find shortest paths efficiently also rendered the graph too simple, in that any shortest path only flipped the symmetric difference. It will be interesting to see if our results help us understand other large graphs, in particular, the flip graph of triangulations of a convex polygon where the complexity of finding the shortest path is still open.

Chapter 4

Flipping Edge-Labelled Triangulations: Part I[1]

4.1 Introduction

Triangulations of point sets and graphs are rich in mathematical structure [31] and have many practical applications, for example in meshing. This chapter is about *reconfiguration* of triangulations. We begin by discussing some necessary background, including some things we have already discussed in previous chapters in order to make this chapter self-contained.

In general, reconfiguration means transforming one solution of a problem to another via elementary steps [55]. Reconfiguration versions of several traditional problems have gained recent attention, e.g., maximum independent set [55, 59], and satisfiability of boolean formulas [47].

The basic operation for reconfiguring triangulations, both in the combinatorial and the geometric setting, is the *flip* operation that removes one edge of the triangulation and adds the other diagonal of the resulting quadrilateral. In order to obtain a new triangulation, there is a constraint on the removed edge. In the geometric setting a triangulation of a point set is a maximal set of non-crossing edges joining pairs of points. The constraint on a flip is that the two faces incident to the removed edge must form a convex quadrilateral. In the combinatorial setting a triangulation is a maximal planar graph with the clockwise

[1]This chapter represents joint work with Prosenjit Bose and Sander Verdonschot. [18]

order of edges around each vertex specified. The constraint on a flip is that the other diagonal of the quadrilateral should not already be an edge of the triangulation.

The fundamental property of flips is that they can be used to reconfigure any triangulation to any other triangulation that has the same size and—in the geometric case—the same point set. This was proved by Wagner in 1936 for the combinatorial setting (see [15]). For the geometric setting, it is a consequence of Lawson's result [63] that any triangulation of a point set can be flipped to the Delaunay triangulation, which then acts as a "canonical" triangulation from which every triangulation can be reached.

There is a substantial literature on flipping, see the survey [15]. Flips are useful in practice in mesh generation and optimizing triangulations [7, 37]. They are also used for counting and generating triangulations [20, 31, 73].

A main subject of investigation is the minimum length of a flip sequence, the so-called *flip distance*, between two triangulations. Most of the work is on worst case bounds. For combinatorial triangulations on n vertices, the current worst case bound is below $6n$ [17]. For triangulations of general point sets of size n there is a tight worst case bound of $\Theta(n^2)$ [52]. For triangulations of points in convex position a famous result of Sleator et al. [95] shows that the flip distance is at most $2n-10$ and that $2n-10$ flips are necessary in some cases. There is a new proof of this result that does not use hyperbolic geometry [86].

Another interesting problem is to find the exact flip distance between two triangulations. The complexity of this problem is open in the combinatorial setting. The problem was only recently shown to be NP-hard [66, 84] (see Chapter 2), and even APX hard [84], for triangulations of point sets. The case of triangulations of a simple polygon is also NP-hard [1]. However, the case of points in convex position (equivalently, the case of a convex polygon) remains stubbornly unresolved, and the hardness reductions used in the aforementioned papers do not seem to apply.

Flips in triangulations of a convex polygon are especially interesting because they correspond exactly to rotations in a binary tree [95] and flip distance corresponds exactly to *rotation distance*, which was first explored in 1982 [29]. There is an easy factor-2 approximation algorithm for rotation distance which can be improved to 1.98 for some special cases [30]. There are efficient algorithms to compute lower and upper bounds on the rotation distance, but no known guarantees on the gap between these bounds [5, 68, 69, 81].

The idea of performing flips in parallel was introduced by Hurtado et al. [53], see also [42]. In the geometric setting, a set of edges may be *simultaneously flipped* if each edge may be flipped and no two of the edges are incident to the same face. Hurtado et al. showed that $O(\log n)$ simultaneous flips are sufficient and sometimes necessary to reconfigure one triangulation of a convex polygon to another. Bose et al. [16] were the first

to explore simultaneous flips in the combinatorial setting—in this case a simultaneous flip may be performed even if some edges in the set cannot be individually flipped.

In this chapter we initiate the study of *edge-labelled flips*. If the edges of a triangulation are labelled and we perform a flip, the newly added edge is assigned the label of the removed edge. In particular, this means that the set of edge labels is preserved throughout any flip sequence. In general (for point sets) it is not possible to flip between any two edge-labelled triangulations, but this is true for combinatorial triangulations and for triangulations of convex polygons, the settings that we consider in this chapter.

Our initial motivation was to understand the complexity of computing the flip distance between two triangulations of a convex polygon. Is the problem difficult because we do not know which edge flips to which edge? Having this information is the same as having a labelling of the edges.

One result that gives some hope that flip distance might become easier to compute when edge labels are specified is a polynomial time algorithm by Eppstein [38] to compute the flip distance between two triangulations of a point set that contains no empty pentagon. In this case Eppstein shows that each edge of the initial triangulation can only flip to a unique edge of the final triangulation. In other words, there is a unique labelling of edges, which is one ingredient in Eppstein's efficient algorithm.

On the other hand, there is a situation where finding a minimum length reconfiguration sequence is NP-hard even with the mapping between initial and final elements, namely for independent set reconfiguration in perfect graphs. For further details, see Section 4.2.

Our Results. We prove tight $\Theta(n \log n)$ bounds on the worst-case flip distance between two edge-labelled triangulations of a point set in convex position, and between two edge-labelled combinatorial triangulations. This contrasts with the $\Theta(n)$ bounds for unlabelled flips [15, 95]. The extra $\log n$ factor arises from sorting-related issues. We prove the upper bound by reducing to the problem of sorting a list using an operation that reverses a (possibly non-contiguous) subsequence at a cost proportional to the minimum length of a contiguous sublist containing the subsequence. The special case where the subsequence itself must be contiguous is the "length-weighted" reversals model introduced by Pinter and Skiena [85] for applications in comparative genomics. Our model seems more powerful since the best-known bound for sorting in their model is $O(n \log^2 n)$. Our $\Omega(n \log n)$ lower bound generalizes theirs. Our result provides an efficient $O(\log n)$-factor approximation algorithm to compute the flip distance between edge-labelled triangulations of a point set in convex position.

Finally, we consider *simultaneous* flips. We prove that the simultaneous flip distance between two edge-labelled triangulations of a point set in convex position is $O(\log^2 n)$, in

contrast with the $\Theta(\log n)$ bound for the unlabelled case [16]. We also prove that $\Omega(\log n)$ simultaneous flips may be required in some cases.

4.2 Related work

Planar point sets with no empty convex pentagons. In SoCG '07, Eppstein [38] showed that flip distance can be computed in polynomial time between two triangulations of a planar point set that does not contain an empty convex pentagon (the vertices of the pentagon are picked from the point set itself). One such point set is the integer lattice, where flip distance has been explored [20].

Given a planar point set $P = \{p_1, \ldots, p_n\}$, define the *quadrilateral graph* QG to have a node v_{ij} for every pair (p_i, p_j) and an edge (v_{ij}, v_{kl}) if and only if p_ip_j and p_kp_l cross and the points p_i, p_j, p_k, p_l form an *empty convex quadrilateral*, i.e., a quadrilateral with no other points of P inside or on its boundary. Thus QG has an edge (v_{ij}, v_{kl}) if and only if there exists a triangulation of P where p_ip_j can be flipped to p_kp_l. Eppstein showed that if P has no empty (strictly) convex pentagons then every triangulation of P contains exactly one node from each connected component of QG. Thus, given two triangulations T_1 and T_2 of P, each edge e of T_1 can be flipped to a unique edge e' of T_2, namely, the one that lies in the same connected component of QG as e. This induces a unique edge labelling, where two edges have the same label if and only if they lie in the same connected component of QG_P. Although not sufficient, this fact is crucial for Eppstein's polynomial time algorithm.

Vertex-labelled triangulations. Although flipping with edge labels is new (to the best of our knowledge) the effect of vertex labels on flips has already been explored. In the combinatorial setting, vertices of the triangulation are not labelled, so a target triangulation is only reached "up to isomorphism". By contrast, in the geometric setting, the position of a vertex in the plane essentially provides a label for the vertex. Sleator et al. [96] studied the effect of vertex labels in the combinatorial setting. They showed that the worst case flip distance is $\Theta(n \log n)$, in contrast with the linear bound for the unlabelled case. The extra $\log n$ factor arises from sorting-related issues, which their paper explores in a broader context.

Flip distance as a reconfiguration problem. Given a set $P = \{p_1, \ldots, p_n\}$ of points in the plane, we can define a graph G, which has a node v_{ij} for every segment p_ip_j $(1 \leq i < j \leq n)$ and an edge (v_{ij}, v_{kl}) whenever p_ip_j crosses p_kp_l at an interior point. Using this definition, a triangulation of P is a maximum independent set of G. Now a flip can

be seen as a *reconfiguration* of the independent set of G, i.e., given an independent set I of $G = (V, E)$, replace $u \in I$ with $v \in V \backslash I$ so that for all $u' \in I \backslash \{u\}, (u', v) \notin E$. This problem has been studied for general graphs under the title of "reconfigurations" [55, 59]. For a general graph, a reconfiguration sequence does not always exist and deciding whether it exists is PSPACE-complete. However, just as for Eppstein's point sets, it is easy to decide which node goes where for *perfect* graphs: since the size of a maximum independent set of a perfect graph is the same as the size of the minimum cover by cliques, each maximum independent set must contain exactly one vertex from each clique of a minimum clique cover. Interestingly, reconfiguration of independent sets is PSPACE-complete even on perfect graphs [58]. Thus at least in this setting, knowing a labelling function is not enough.

Sorting permutations. As mentioned above, our bound on flip distance relies on bounds for sorting permutations. Sorting permutations using operations from a restricted set of operations has been widely studied. A main operation is the reversal of a subsequence. One of the earliest results of this kind was on "pancake" sorting [44] where a prefix of the sequence can be reversed. Bubble sort operates by swapping two adjacent elements at a time, which is a reversal of size two. The number of size-two reversals it makes is equal to the number of inversions of the permutation and is $\Theta(n^2)$ in the worst case. More generally, any permutation can be sorted with $\Theta(n)$ reversals [62] if arbitrarily large reversals are allowed. Computing the minimum number of operations (of a specific kind) required to sort a permutation is of interest in bioinformatics. In the case of arbitrarily large reversals, this problem is NP-complete [21] for sorting permutations, but is in P for sorting *signed* permutations [3]. Some other kinds of operations that have been considered include transpositions and block interchanges [31]. Most of the results until now have been about the *number* of operations required. Recently, Pinter and Skiena [85] formulated a model that assigns a cost to each operation based on the length of the interval in which it is performed. Improved bounds for this model were given by Bender et al. [6]. We use this cost model but generalize to reversals of non-contiguous sequences.

4.3 Upper bound on the worst-case flip distance

4.3.1 Convex polygons

Note that every triangulation of a convex polygon contains two kinds of edges—a set of *boundary edges* that lie on the convex hull and cannot be flipped, and a set E of non-boundary edges or *diagonals* that can be flipped. Combinatorial triangulations, by

contrast, have no boundary edges. We only label the non-boundary edges. More precisely, an *edge-labelled* triangulation $\mathcal{T}$ is a pair (T, l) where T is a triangulation that has a set E of non-boundary edges and $l : E \to \{1, \ldots, |E|\}$ is a function that assigns a unique label $l(e)$ to each $e \in E$.

We will denote by T^* the unlabelled *canonical* triangulation as shown in Figure 4.1 where there is one vertex p_1 that is shared by all the diagonals. If we draw T^* with p_1 at the top, the drawing induces a left-to-right order on the n diagonals. Reading their labels in this order defines a permutation of $[1..n]$. Given a permutation ρ of $[1..n]$, we will denote by (T^*, ρ) the canonical triangulation where the edge-labels, when read out from left to right, form the permutation ρ. The next lemma shows that in this canonical triangulation, ordered subsequences can be easily reversed, which plays an important role for the upper bound.

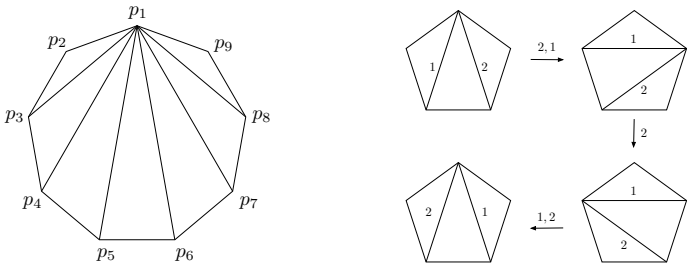


Figure 4.1: (Left) A canonical triangulation (T^*, l) with 6 diagonals. (Right) Flip-sequence to reverse the order of the two diagonals of the canonical triangulation of a pentagon.

Lemma 4.1. *Given (T^*, ρ), let S be a contiguous subsequence of edges, excluding the polygon boundary edges, adjacent to vertex p_1 ordered in counter-clockwise order. Any ordered subsequence $R = i_1, \ldots, i_r$ of S can be reversed with $O(|S|)$ flips. Note that the edges of R need not be consecutive.*

Proof. We use induction on $|S|$. If there is any edge in S that is not in R, simply flip that edge, apply induction since $|S|$ has gotten smaller, and flip the edge back. Therefore, we only need to consider the case where $S = R$. If r is odd, first flip $i_{(1+r)/2}$. Apply induction to reverse the remaining subsequence of $r - 1$ edges. Next, flip $i_{(1+r)/2}$ back. If r is even, then flip both $i_{r/2}$ and $i_{r/2+1}$. Apply induction to reverse the remaining subsequence of $r - 2$ edges. Now, flip $i_{r/2}$ and $i_{r/2+1}$ back. These two edges are out of order but they can be reversed with five flips as shown in Figure 4.1. Thus, five flips reduce the problem size by two. Altogether, at most $2.5r$ flips are used to reverse the subsequence. □

Theorem 4.1. *Any edge-labelled triangulation (T_1, l_1) of a convex polygon P with n edges can be transformed into any other edge-labelled triangulation (T_2, l_2) using $O(n \log n)$ flips.*

Proof. We first ignore the labels and use $O(n)$ flips to transform (T_1, l_1) and (T_2, l_2) to (T^*, l_1') and (T^*, l_2') respectively [95]. Next, we show that a canonical triangulation (T^*, ρ) corresponding to any permutation ρ can be transformed using $O(n \log n)$ flips into the triangulation (T^*, ρ^*), where ρ^* is the sorted permutation. Combining these steps proves the theorem. By using Lemma 4.1, we reduce the problem of flipping edge-labelled triangulations to the problem of sorting permutations in the following model.

Non-contiguous reversals Given a permutation ρ of $[1..n]$, pick any interval $[i..j]$ with $1 < i < j < n$ and $j - i + 1 = k$. Then pick a subsequence $i_1, \ldots, i_r$ of $[i..j]$ and reverse it (without changing the position of any element outside the subsequence). The cost of this reversal is $O(k)$.

In particular, if we can show that any permutation of $[1..n]$ can be sorted with a worst case cost of $O(n \log n)$ in this model, that would give us an $O(n \log n)$ cost for flipping edge-labelled triangulations as well.

We can sort in this model by imitating quicksort. Consider all values of i for which either: (1) $i \leq n/2$ and $\rho_i > n/2$; or (2) $i > n/2$ and $\rho_i \leq n/2$. Reverse the subsequence formed by these values of i. This costs $O(n)$ and ensures that elements smaller than $n/2$ lie in the first half of the sequence and the larger elements lie in the second half. Now recurse in the two halves. □

As a corollary to the $O(n \log n)$ upper bound, we can get an $O(\log n)$-factor approximation algorithm for computing the flip-distance.

Corollary 4.1. *Given two edge-labelled triangulations $\mathcal{T}_1 = (T_1, l_1)$ and $\mathcal{T}_2 = (T_2, l_2)$, we can find an $O(\log n)$-approximation to the flip-distance between them in polynomial time.*

Proof. Call an edge $p_i p_j$ *fixed* if it satisfies the following properties: *a*) it occurs in both T_1 and T_2; *b*) it has the same label in both T_1 and T_2, i.e., $l_1(p_i p_j) = l_2(p_i p_j)$; and *c*) the set of distinct labels that occurs on the left of $p_i p_j$ in $\mathcal{T}_1$ is exactly the same as the set that occurs on the left of $p_i p_j$ in $\mathcal{T}_2$. Our approximation algorithm will not flip any fixed edges. Note that every non-fixed edge must flip at least once. The set of fixed edges divides the polygon into several smaller polygons, each of which is triangulated differently in $\mathcal{T}_1$ and $\mathcal{T}_2$. Suppose the i^{th} polygon has n_i diagonals. These are non-fixed so we have a lower bound of $\Omega(n_i)$ flips. We use the $O(n_i \log n_i)$ flip-sequence from Theorem 4.1 on the i^{th} polygon, thus giving an approximation factor of $O(\log n)$. □

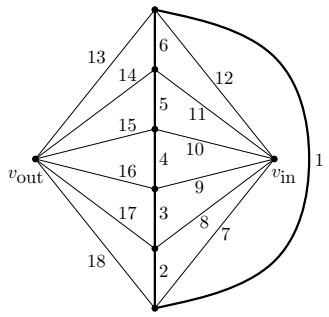


Figure 4.2: The labelled canonical triangulation on 8 vertices. The spine is indicated in bold.

4.3.2 Combinatorial triangulations

With the convex polygon case settled, we turn our attention to combinatorial triangulations. We will assume that the vertices are unlabelled.

Theorem 4.2. *Any edge-labelled combinatorial triangulation with n vertices can be transformed into any other by $O(n \log n)$ flips.*

Proof. We will show that we can transform any edge-labelled combinatorial triangulation into a canonical one using $O(n \log n)$ flips. Since flips are reversible, we can also go from the canonical triangulation to any other, which proves the theorem.

We use a canonical triangulation like the one used by Sleator et al. [96] for the vertex-labelled variant. It is a "double wheel": a cycle of length $n-2$ (called the "spine"), plus a vertex v_{in} inside the cycle and a vertex v_{out} outside the cycle, each connected to every vertex on the cycle (see Figure 4.2). For our canonical labelling, we place labels $1, \ldots, n-2$ (called "group $\mathcal{S}$") consecutively along the spine edges. We place the next $n-2$ labels (group $\mathcal{C}_{\text{in}}$) consecutively on the edges incident to v_{in} and the last $n-2$ labels (group $\mathcal{C}_{\text{out}}$) consecutively on the edges incident to v_{out}.

As for convex polygons, our algorithm first ignores the labels and transforms the given triangulation into the unlabelled canonical triangulation. This requires $O(n)$ flips [96] and puts the edges in place, though possibly with the wrong labels. To fix the labels, we first get the groups right, so all labels in group $\mathcal{S}$ are on the spine, etc., and then rearrange labels within each group.

We use two main tools. The first is a "swap" that interchanges one spine edge with an incident non-spine edge using the constant size sequence of flips shown in Figure 4.3 (in Figure 4.3).

Our second tool is a "scramble" algorithm that reorders all labels incident to v_{in} using $O(n \log n)$ flips. To do this we first flip the spine edge that is part of the exterior face (labelled 1 in Figure 4.2) and then apply the algorithm from Theorem 4.1 to the outerplanar graph induced by the spine plus v_{in}, observing that no flip will create a duplicate edge since the omitted edges are all incident to v_{out}. This method cannot alter the labels on the two non-spine edges that lie on the exterior face of the outerplanar graph (labelled 12 and 7 in Figure 4.2), but since there are only two of these, we can move them to their correct places by swapping them along the spine, using $O(n)$ flips total. Similarly, we can scramble the labels incident to v_{out} with $O(n \log n)$ flips.

To get the labels of group $\mathcal{S}$ on the spine, we partner every edge incident to v_{in} that has a label in $\mathcal{S}$ with an edge on the spine that has a label in $\mathcal{C}_{\text{in}}$ or $\mathcal{C}_{\text{out}}$. A scramble at v_{in} makes each such edge incident to its partner, and then swaps exchange partners. We do the same at v_{out}, thus placing all labels of $\mathcal{S}$ on the spine. Next we partner every edge incident to v_{in} that has a label in $\mathcal{C}_{\text{out}}$ with an edge incident to v_{out} that has a label in $\mathcal{C}_{\text{in}}$. A scramble at v_{in} makes partners incident, and three swaps per pair can then exchange partners.

Now each edge's label is in the correct group, but labels within each group may be wrong. We can scramble at v_{in} and at v_{out} to rearrange the labels in $\mathcal{C}_{\text{in}}$ and $\mathcal{C}_{\text{out}}$. The last thing we need to do is reorder the edges on the spine. We use swaps to exchange the labels on the spine with those incident to v_{in} in $O(n)$ flips and scramble at v_{in} to order them correctly. Since this does not affect the labels on the spine, we can simply exchange the edges once more to obtain the canonical triangulation. □

4.4 Lower bounds

Theorem 4.3. *There exist pairs of edge-labelled triangulations of a convex polygon with n diagonals, such that any flip sequence that transforms one into the other contains at least $\Omega(n \log n)$ flips.*

Proof. Sleator et al. [96] showed that, starting from any binary tree, any sequence of k rotations can generate at most c^k distinct trees, for some constant c. The original proof assumes a vertex-labelled setting, but translates to any setting where the labelling is uniquely determined by the labelling before the flip and the edge that was flipped. Since each permutation of the edge labels in the canonical triangulation results in a distinct labelled triangulation, there are at least $n!$ distinct labelled triangulations. If we let d be

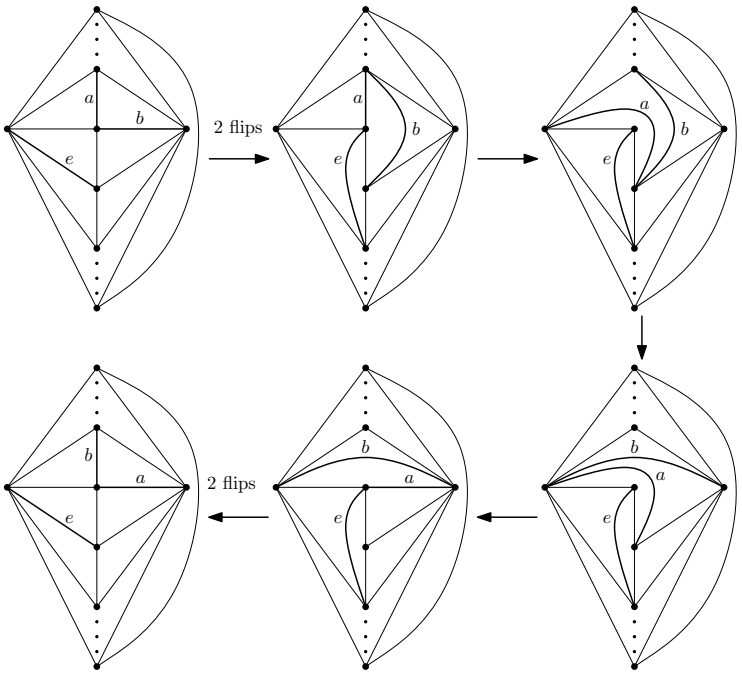


Figure 4.3: Swapping two edges a and b that are consecutive around a vertex on the spine. Although edge e ends up at the same place as at the start of the sequence, it essentially acts as a catalyst here. If we did not flip it, we would not be able to flip edge a after edge b, as that would create a duplicate edge.

the minimum number of flips that is sufficient to transform a given labelled triangulation into any other, then $c^d \geq n!$, which implies $d \geq \Omega(n \log n)$. □

The result of Sleator et al. [96] used in the above proof also applies to flips in combinatorial triangulations. Since there are at least $\Omega(n!)$ distinct edge-labelled combinatorial triangulations, we immediately obtain the following result.

Theorem 4.4. *There exist pairs of edge-labelled combinatorial triangulations on n vertices, such that any flip sequence that transforms one into the other has at least $\Omega(n \log n)$ flips.*

4.4.1 Sorting in length-weighted models

Consider the following general model of sorting. Given a permutation ρ, pick an interval $[i..j]$ with $1 \leq i \leq j \leq n$ and $j - i + 1 = k$ and perform a set of swaps in it. The cost

of this operation is $O(k)$. Adding constraints on the kinds of swaps leads to more specific models. For example, in the non-contiguous reversals model from Section 4.3.1, we are only allowed to swap a set $\{(i_1, j_1), \ldots, (i_m, j_m)\}$ of pairs such that for any $x, y \in [1..m]$, if $x < y$, then $i \leq i_x < i_y < j_y < j_x \leq j$, i.e., the interval $[i_y, j_y]$ is nested inside the interval $[i_x, j_x]$. Lemma 4.1 and Theorem 4.3 imply that the worst-case cost of sorting in the non-contiguous reversals model is $\Omega(n \log n)$.

A less general model of *contiguous reversals* was considered by Pinter and Skiena [85] and later by Bender et al. [6]. In this model, at cost $O(k)$ we pick an interval $[i..j]$ where $j - i = k$ and reverse it. Our lower bound implies theirs.

More generally, proving that a model can be "simulated" by flips will imply an $\Omega(n \log n)$ lower bound on the worst-case cost of sorting in that model. We apply this idea to the following very general model of sorting:

Non-crossing swaps Pick an interval $[i..j]$ (with $k = j-1+1$) and swap a set $\{(i_1, j_1), \ldots, (i_m, j_m)\}$ of pairs inside it such that there do not exist $x, y \in [1..m]$ with $x < y$ for which $i_x \leq i_y \leq j_x \leq j_y$, i.e, either the two intervals $[i_x, j_x]$ and $[i_y, j_y]$ nest or are disjoint. The cost of this operation is $O(k)$.

Recall that in the non-contiguous reversals model, we wanted *all* pairs to nest. Therefore this model is more general. The next lemma shows that this model can be simulated using flips, which implies that the lower bound also applies to this model.

Lemma 4.2. *Let ρ and ρ' be two permutations of $[1..n]$ such that ρ' can be obtained from ρ using one set of non-crossing swaps of cost $O(k)$. Then the triangulation (T^*, ρ) can be flipped to (T^*, ρ') with $O(k)$ flips.*

Proof. The flip-sequence is constructed almost exactly like in Lemma 4.1. We first make the set of edges that participate in a swap a connected set. Next, we find $x \in [1..m]$ such that the interval (i_x, j_x) does not contain any interval (i_y, j_y) for $y \neq x$ and flip ρ_{i_x} followed by ρ_{j_x}. Then we recurse on the smallest triangulation containing the rest of the edges and then perform the required flips on the pentagon containing ρ_{i_x} and ρ_{j_x}. □

Theorem 4.5. *The worst-case cost of sorting in the non-crossing swaps model is $\Omega(n \log n)$.*

This lower bound is quite powerful because the non-crossing swaps model is more general than (length-weighted versions of) a large number of sorting models [31, Chapter 3].

4.5 Simultaneous flips

Theorem 4.6. *Given two edge-labelled triangulations of a convex polygon, $O(\log^2 n)$ simultaneous flips are always sufficient to transform one to the other.*

Proof. We first ignore the labels and, by the result of Galtier et al. [42], use $O(\log n)$ simultaneous flips to transform both (T_1, l_1) and (T_2, l_1) to the canonical triangulations (T^*, l_1') and (T^*, l_2') respectively. Next, we show how to transform a general canonical triangulation (T^*, ρ) to (T^*, ρ^*), where ρ^* is the sorted permutation. We do this by imitating quicksort. We show that the "separation" step of quicksort can be carried out in $O(\log n)$ simultaneous flips. We can then recurse in both halves simultaneously. Thus we get the recursion $T(n) = T(n/2) + O(\log n)$, which solves to $T(n) = O(\log^2 n)$.

We now address the separation step of quicksort. Let $E_l = \{\rho_i \mid i < n/2 \text{ and } \rho_i \geq n/2\}$ be the set of edges that are in the left half in ρ but in the right half in ρ^*. Similarly, let E_r be the set of edges that are in the right half in ρ but in the left half in ρ^*. Let E be the set of all non-boundary edges. We will first flip all the edges of $E \backslash (E_l \cup E_r)$. One simultaneous flip can flip every other edge of the set, and therefore $O(\log n)$ simultaneous flips can flip the whole set. The edges of $E_l \cup E_r$ now form a canonical triangulation of a smaller convex polygon. For the remainder of the proof we work only with this smaller convex polygon. In other words, we only consider canonical triangulations (T^*, ρ) where *every* edge on the left half wants to go to the right half and vice versa.

Let $\mathcal{T}$ be a canonical triangulation where every edge on the left half is coloured red and every edge on the right half is coloured blue. Let $\mathcal{T}'$ be the triangulation with the colours interchanged. We show how to transform $\mathcal{T}$ to $\mathcal{T}'$ with $O(\log n)$ simultaneous flips. We do this by converting both $\mathcal{T}$ and $\mathcal{T}'$ into a common coloured triangulation in which the diagonals form a path that alternates between red and blue edges, as shown in Figure 4.4. We call this triangulation an *alternating zig-zag*. Note that p_1 is the high degree vertex of the canonical triangulation.

Imagine a line L passing through p_1 that separates the red edges from the blue edges in the canonical triangulation and crosses all the edges in the alternating zig-zag. We show how to transform the alternating zig-zag to $\mathcal{T}'$. We claim that in $O(1)$ simultaneous flips, we can ensure that half the blue edges are to the left of this line and half the red edges are to the right. Moreover, the edges that still cross L form an alternating zig-zag of a subpolygon at most half the size of $\mathcal{T}'$.

To do this, divide the zig-zag into hexagons, as shown in Figure 4.4. Each hexagon will have three diagonals, two red and one blue. Any transformation of a hexagon takes

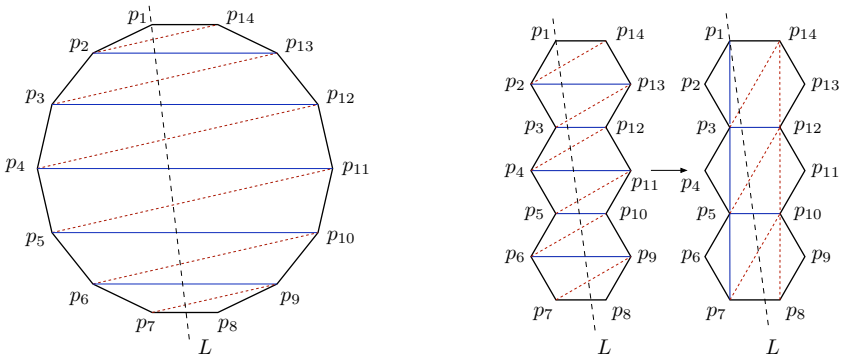


Figure 4.4: (Left) An 11-edge zig-zag alternating between red (dotted) and blue (solid) edges. (Right) An alternating zig-zag can be thought of as a collection of hexagons. In $O(1)$ simultaneous flips, we can move half of the edges to the correct side of L.

a constant number of flips. In particular, transform each hexagon to have one blue edge on the left and one red edge on the right, as shown in Figure 4.4. Perform the same transformation on all hexagons simultaneously.

Thus with a constant number of simultaneous flips, half the blue edges are completely to the left of L and half the red edges are completely to the right. Repeating the process $O(\log n)$ times ensures that every blue edge is to the left of L and every red edge is to the right. The subpolygon to the left of L now has only blue diagonals. With $O(\log n)$ simultaneous flips we can make it canonical with p_1 as the top-most vertex [42]. The same can be done with the red edges. Thus the alternating zig-zag can be transformed into $\mathcal{T}'$ using $O(\log n)$ simultaneous flips. Similarly, we can transform the alternating zig-zag into $\mathcal{T}$ using $O(\log n)$ simultaneous flips. □

The $\Omega(\log n)$ lower bound on the worst-case number of simultaneous flips in the unlabelled case trivially provides an $\Omega(\log n)$ lower bound in the labelled setting as well. We prove a stronger lower bound. In particular, we prove that even the "separation" step of quick sort requires at least $\Omega(\log n)$ simultaneous flips.

Theorem 4.7. *Let $\mathcal{T}$ and $\mathcal{T}'$ be the triangulations defined in Section 4.5, i.e., both are canonical, in $\mathcal{T}$ all edges on the left half are labelled "red" and all edges on the right half labelled "blue" and in $\mathcal{T}'$, the labels are interchanged. Transforming $\mathcal{T}$ to $\mathcal{T}'$ requires at least $\Omega(\log n)$ simultaneous flips.*

Proof. Consider the line L that passes through p_1 and separates the red edges from blue ones in $\mathcal{T}$. For each edge, the side of L it inhabits in $\mathcal{T}$ is different from the side it inhabits

in $\mathcal{T}'$. A single flip cannot replace an edge that lies completely to the left of L with an edge that lies completely to the right. Thus in any simultaneous flip-sequence that transforms $\mathcal{T}$ to $\mathcal{T}'$, for any edge e, there must be a triangulation where e intersects L.

Consider any simultaneous flip-sequence $\mathcal{F}$ that transforms $\mathcal{T}$ to $\mathcal{T}'$, i.e., $\mathcal{F}$ is the sequence $(\mathcal{T} = \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{k-1}, \mathcal{T}_k = \mathcal{T}')$. For any $j \in [1..k]$, let c_j be the number of edges e for which there exists a $\mathcal{T}_r$ with $r \leq i$ such that e intersects L in T_r. From the argument above, it is clear that $c_1 = 0$ and $c_k = n$. We claim that for all $j \in [1..k-1]$, $c_{j+1} \leq 2c_j + 1$. This shows that $k \geq \Omega(\log n)$.

To see why the claim is true, consider the j^{th} simultaneous flip in the sequence. It makes $\Delta_j = c_{j+1} - c_j$ new edges cross L. Each flip in the simultaneous flip happens in its own quadrilateral and exactly two boundary-edges of each quadrilateral intersect L. Since two adjacent quadrilaterals can share an edge and since the quadrilateral at the top and bottom share their boundaries with the polygon, the total number of quadrilateral edges that cross L in $\mathcal{T}_j$ is at least $\Delta_j - 1$. But this number is also at most c_j. Therefore $\Delta_j - 1 \leq c_j$, which means $c_{j+1} \leq 2c_j + 1$. □

4.6 Conclusion

We have initiated the exploration of flips in edge-labelled triangulations. For combinatorial triangulations and triangulations of convex polygons, any edge-labelled triangulation can be flipped to any other, and the worst case number of flips is a $\log n$ factor more than for the unlabelled case. With simultaneous flips, edge labels raise the worst case bound by at most a $\log n$ factor, but we could not prove that this is tight.

Our motivation was to settle the complexity of computing the flip distance between two triangulations of a convex polygon, but this remains an open question, both in the labelled and unlabelled settings. For proving hardness, we suggest tackling the case when labels are not necessarily distinct since this generalizes both settings.

Flipping between edge-labelled triangulations of a general point set in the plane is wide open. In Chapter 5, we generalize our results to spiral polygons and present evidence that edge-labelled triangulations of planar point sets are hard to analyse.

Chapter 5

Flipping Edge-Labelled Triangulations: Part II[1]

5.1 Introduction

In this chapter, we continue the investigation of Chapter 4 by studying the edge-labelled triangulations of geometric objects that are more general than convex polygons. Our first observation is that even a slight generalization changes the nature of the problem drastically. It is easy to construct, for example, a polygon that has a unique triangulation. If we label this triangulation in two different ways, it is clear that there is no way to reconfigure one to the other using flips. This never happens for convex polygons since any two edge-labelled triangulations of a convex polygon can be transformed into one another.

A slightly more involved example is shown in Figure 5.1. In the polygon $ABCDE$, the only diagonal that AC can be flipped to is BE and the only diagonal that BE can be flipped to is AC. Thus consider the following two labelled triangulations created by labelling the triangulation shown in the figure: one has AC labelled red and EC labelled blue, and the other with AC labelled blue and EC labelled red. It is clear that no flip sequence transforms between these two.

The occasional disconnectedness of the flip graph raises new questions.

1. When is the flip graph connected?

[1]This chapter represents joint work with Prosenjit Bose, Anna Lubiw, and Sander Verdonschot [18].

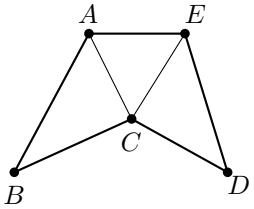


Figure 5.1: A polygon with a disconnected flip graph

2. Given two edge-labelled triangulations, how do we decide whether there exists a flip sequence that transforms one to the other, i.e., they are connected in the flip graph?

3. Given two edge-labelled triangulations that are connected in the flip graph, what is the distance between them in the worst case?

4. Given two edge-labelled triangulations, what is the shortest path between them in the flip graph?

The third question is what we focused on in the previous chapter under the special case where the predicate was always known to be true. In this chapter, we provide answers to the first three questions for the case of spiral polygons—simple polygons that have at most one reflex chain. Next, we provide evidence that polygons with two reflex chains are already hard to analyze. We end the chapter by making some conjectures about more general cases.

5.2 Preliminaries

The generalized configurations that one may want to study are planar point sets, simple polygons, and polygonal domains. All of them fit into the following framework. We are given a two-dimensional geometric object $\mathcal{P}$ that is defined by a set P of points on a plane. For example, if the object is a point set, then P itself is the object; if the object is a polygon or a polygonal domain, then P is the set of its vertices. The object partitions the plane into regions some of which are said to be *inside* the object and others *outside*. For polygons and polygonal domains, the definitions of inside and outside are the natural ones with the boundary considered to be outside. For planar point sets, we let the region inside the convex hull to be inside and the region outside, including the convex hull itself, to be outside. For any pair of points $x, y \in P$, we say that the line segment joining x and y is a *diagonal* of $\mathcal{P}$ if it is completely inside $\mathcal{P}$ and does not contain any other points of P on

it. We say that two diagonals *cross* if they intersect at a point that is not an endpoint. A *triangulation* of $\mathcal{P}$ is any maximal collection of non-crossing diagonals.

We define flips as follows. For a given point set P in the plane and any polygon $\mathcal{Q}$, we say that $\mathcal{Q}$ is *empty* if it does not contain any points of P other than its vertices, i.e., if $\mathcal{Q}$ has boundary B, interior I and vertices V, then $B \setminus V \cup I$ does not contain any points from P. Given a triangulation T of $\mathcal{P}$, pick four line segments each of which is either a diagonal of T or a boundary edge of $\mathcal{P}$ such that the four segments form the boundary of an empty convex quadrilateral Q whose interior is inside $\mathcal{P}$. Then one of the diagonals of Q must also be in T (from maximality). Replacing this diagonal with the other diagonal of Q will be called one flip.

Flips define the flip graph, where there is a node for each triangulation of $\mathcal{P}$ and edge (T, T') whenever there exists a flip that converts T into T'. It is well known that the flip graph is always connected [64, 7]. Let D be the set of all diagonals of $\mathcal{P}$ and for any pairwise non-crossing subset D' of D, let $\mathscr{T}_{D'}$ be the set of all triangulations that contain all elements of D'. It is also known, using constrained Delaunay triangulations [7], that the graph induced on the flip graph by the set $\mathscr{T}_{D'}$ is connected. This amounts to saying that any triangulation can be transformed into any other while keeping a certain desired set of diagonals fixed.

The following graph, first defined by Eppstein [38], is also useful in our investigations.

Definition 5.1 (Quadrilateral graph)**.** *For a geometric object $\mathcal{P}$ as defined above, we define a* quadrilateral graph *denoted $QG_{\mathcal{P}}$, in which there is a vertex for each diagonal of $\mathcal{P}$ and there is an edge (d, d') whenever d and d' cross and their four endpoints form an empty convex quadrilateral. Whenever the context is clear, we will drop the subscript from $QG_{\mathcal{P}}$.*

It is easy to see that there exists an edge between d and d' in QG if and only if there exists a triangulation in which d can be flipped to d'.

An *edge-labelled* triangulation $\mathcal{T}$ is a pair (T, l) where T is a triangulation that has a set D of diagonals and $l : D \to \{1, \ldots, |D|\}$ is a function that assigns a unique label $l(d)$ to each $d \in D$. A flip of an edge e to e' assigns label $l(e)$ to e'. Since the labels are unique, we will frequently use the labels to refer to the diagonals. If a flip sequence F transforms triangulation $\mathcal{T}_1$ to $\mathcal{T}_2$ such that label i is assigned to d_1 in $\mathcal{T}_1$ and to d_2 in $\mathcal{T}_2$, we say that F *moves* label i from d_1 to d_2. For label i, the diagonal with that label is given by $l^{-1}(i)$.

We define the *edge-labelled flip graph*, denoted FG, as the graph that has a node for each edge-labelled triangulation and an edge $(\mathcal{T}, \mathcal{T}')$ if there exists a flip that transforms $\mathcal{T}$ to $\mathcal{T}'$. Since this chapter is about edge-labelled triangulations, we will omit the qualifier

"edge-labelled" from edge-labelled flip graphs and simply call them flip graphs from now on.

Unlike the case of unlabelled triangulations, flip graphs and quadrilateral graphs are closely related for the case of edge-labelled triangulations. For example, the following is easy to show.

Lemma 5.1. *If the flip graph is connected, then the quadrilateral graph is also connected.*

Proof. Consider any two diagonals d and d'. Let T be a triangulation containing d and T' be a triangulation containing d'. Label both T and T' to get $\mathcal{T}$ and $\mathcal{T}'$ such that d and d' get the same label λ in both. Since the flip graph is connected, there exists a flip sequence that transforms $\mathcal{T}$ to $\mathcal{T}'$. This flip sequence provides a sequence $e_1, \ldots, e_k$ of diagonals such that $e_1 = d$, $e_k = d'$, and for any e_i and e_{i+1}, there exists a triangulation where e_i can be flipped to e_{i+1}. This means there is a path from d to d' in QG. □

What about the other direction? Does the connectivity of QG imply connectivity of FG? Although it is unclear whether it is true in general, we prove it for the case of spiral polygons. For general geometric objects, connectivity of QG does imply a much weaker condition for FG. In particular, if we care about only one label at a time, connectivity of QG is all we need to look at.

Theorem 5.1. *Let $\mathcal{T}$ and $\mathcal{T}'$ be two triangulations such that label λ is assigned to d in $\mathcal{T}$ and to d' in $\mathcal{T}'$. We can move λ from d to d' if and only if d and d' lie in the same connected component of QG.*

Proof. We use the fact that any unlabelled triangulation can be flipped to any other while keeping a desired set of diagonals fixed. If d and d' do not lie in the same component of QG, then it is clear that we cannot move λ from d to d'. For the other direction, let $d = e_1, \ldots, e_k = d'$ be a path between d and d' in QG. Consider the empty quadrilateral Q formed by the endpoints of e_1 and e_2. We can transform $\mathcal{T}$ to some triangulation $\mathcal{T}_1$ that has all four boundary edges of the quadrilateral as its diagonals while keeping λ fixed. Then we flip λ inside Q. This moves λ to e_2. In general, suppose we have moved λ to e_i. Let Q be the empty convex quadrilateral formed by e_i and e_{i+1}. We create Q while keeping λ fixed and then flip λ to move it to e_{i+1}. □

5.3 Spiral polygons

For a spiral polygon P with m convex and k reflex vertices, let $C = c_1c_2 \cdots c_m$ be the convex chain and $R = r_1r_2 \cdots r_k$ denote the reflex chain such that c_1 and r_1 are adjacent, and c_m and r_k are adjacent. For any $i < j$, we will denote the subchain $c_i \dots c_j$ by C_{ij} and the subchain $r_i \dots r_j$ by R_{ij}. We will also abuse notation and use C, R, C_{ij}, and R_{ij} to denote the sets of vertices in the corresponding chains. For chains $C_{j,j'}$ and $R_{i,i'}$, if c_j is visible from r_i and c'_j is visible from r'_i, we will use the shorthand "polygon defined by $C_{j,j'}$ and $R_{i,i'}$" to denote the polygon whose boundary consists of the chains $C_{j,j'}$, $R_{i,i'}$, and the edges r_ic_j and $r_{i'}c_{j'}$. For any $i < j$, we will say that c_i occurs to the *left* of c_j, and r_i occurs to the left of r_j even though it may not be true geometrically. We will use the terms *left-most* and *right-most* in a similar fashion. For two subchains C_{ij} and $C_{i'j'}$, we say that C_{ij} occurs to the left of $C_{i'j'}$ if $i \leq i'$ and $j \leq j'$. For what follows, n denotes the number of diagonals in the triangulations.

For any point c on the convex chain, we define $V(c)$ to be the set of points on the reflex chain that are visible from c. Similarly, for any point r on the reflex chain, $V(r)$ denotes the set of points on the convex chain that are visible from r. The following properties are easy to show.

Property 5.1. *For any $c \in C$ and $r \in R$, the sets $V(c)$ and $V(r)$ form subchains of R and C respectively.*

Property 5.2. *For $i < i'$ and $j < j'$, $V(r_i)$ occurs to the left of $V(r_{i'})$ and $V(c_j)$ occurs to the left of $V(c_{j'})$.*

Property 5.3. *For any two consecutive vertices r_i and r_{i+1} on the reflex chain, $|V(r_i) \cap V(r_{i+1})| \geq 1$.*

5.3.1 Connectivity of the flip graph

In this section, we provide a characterization for the spiral polygons for which the flip graph is connected. We say that a spiral polygon is *locally convex* if for every four consecutive vertices a, b, c, and d on the convex chain, the convex quadrilateral $abcd$ is empty.

Theorem 5.2. *For any spiral polygon $\mathcal{P}$, the following statements are equivalent:*

1. *$\mathcal{P}$ is locally convex.*

2. *$QG_{\mathcal{P}}$ is connected.*

3. *$FG_{\mathcal{P}}$ is connected.*

Furthermore, the diameter of each connected component of FG is $O(n^2)$ and there are cases when FG has a diameter of $\Omega(n^2)$.

Proof. We prove that (1) $\implies$ (3), (3) $\implies$ (2), and (2) $\implies$ (1). The implication (3) $\implies$ (2) is given by Lemma 5.1. Thus we focus on the other two.

For (1) $\implies$ (3), we provide an explicit flip sequence. Given any two edge-labelled triangulations $\mathcal{T}$ and $\mathcal{T}'$, we ignore labels and transform both of them into a canonical triangulation that we describe later. Using Hanke's upper bound [50] on the diameter of the flip graph for unlabelled triangulations of a spiral polygon, this can be done with $O(n)$ flips. Next, we rearrange the labels on the canonical triangulation.

Our canonical triangulation has the property that each diagonal connects a convex vertex with a reflex vertex. Any such triangulation induces an ordering on its diagonals: we say that $r_i c_j$ occurs to the *left* of $r_{i'} c_{j'}$ if $i \leq i'$. Thus reading the labels of the diagonals from left to right gives us a permutation of $[1..n]$, which means rearranging the labels amounts to sorting this permutation.

Our flip sequence will imitate insertion sort. In particular, for a canonical triangulation, let $d_1, \ldots, d_n$ be its diagonals listed left-to-right. We define an *insertion* step to be a sequence of flips that rearranges the labels in the following way: the label of d_j moves to d_i for some $i < j$, and for every $i' \in [i..j-1]$, the label of $d_{i'}$ moves to $d_{i'+1}$. We will say that any such sequence *inserts* d_j at d_i. We will show that in our canonical triangulation, any such insertion step can be performed with a flip sequence of length at most $O(n)$. Since sorting requires at most $O(n)$ insertions, this gives an upper bound of $O(n^2)$.

Before describing the canonical triangulation, we show the following useful property of locally convex spiral polygons.

Lemma 5.2. *For any two consecutive vertices $r_i, r_{i+1} \in R$ on the reflex chain, $|V(r_i) \cap V(r_{i+1})| \geq 3$.*

Proof. Suppose, for contradiction, that $|V(r_i) \cap V(r_{i+1})| \leq 2$. Let c_j be the right-most vertex of $V(r_i)$ and $c_{j'}$ be the left-most vertex of $V(r_{i+1})$ (Figure 5.2). From Property 5.3, we know that either $j = j'$ or $j' = j - 1$. Consider the vertices $c_{j'-1}$ and c_{j+1}. Note that they exist since c_1 and c_m are visibly only from r_1 and r_k respectively. Consider the infinite ray starting at r_{i+1} in the direction of $r_{i+1}c_{j'}$ and rotate it towards $c_{j'-1}$ until it hits a point

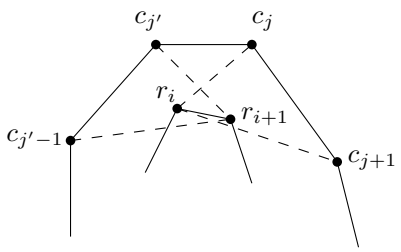


Figure 5.2: Proof of Lemma 5.2

inside the triangle $r_{i+1}c_{j'}c_{j'-1}$. Since r_{i+1} does not see $c_{j'-1}$, this ray must hit r_i before it hits $c_{j'-1}$. That means r_i lies inside the triangle $r_{i+1}c_{j'}c_{j'-1}$. Similarly, r_{i+1} lies inside the triangle $r_ic_jc_{j+1}$. This is possible only if both r_i and r_{i+1} lie inside the quadrilateral $c_{j'-1}c_{j'}c_jc_{j+1}$ (which is a triangle in case $j = j'$). But that would mean that the polygon was not locally convex. □

Next, we describe how to construct the canonical triangulation T. We describe T by specifying, for each boundary edge, the apex of the unique triangle of T that contains it. For each edge r_ir_{i+1} of the reflex chain, we pick the apex to be a convex vertex c and connect both r_i and r_{i+1} with c. Let $V(r_i) \cap V(r_{i+1}) = \{c_j, \ldots, c_{j'}\}$. We pick c to be $c_{\lfloor \frac{j+j'}{2} \rfloor}$. After doing this, each edge c_jc_{j+1} of the convex chain has a unique reflex vertex r that can be its apex; thus we connect both c_j and c_{j+1} with r. It is clear that this forms a triangulation such that each of its diagonals joins a convex vertex with a reflex vertex and thus can be ordered from left to right.

Next, we describe how to perform insertions in T using $O(n)$ flips. We do that by partitioning the set of diagonals of T into smaller subsets and describing how to perform insertions inside each subset. We say that a set of diagonals of T is a *reflex fan* if it shares a reflex vertex and a *convex fan* if it shares a convex vertex.

We construct the partitioning using the following iterative procedure. Let $\mathcal{D}$ be the set of subsets $\{D_1, \ldots, D_t\}$ of the partition. We start with $\mathcal{D}$ being empty and add sets to it one by one. First, for each convex vertex c of the polygon that has at least two diagonals incident on it, we add the set of diagonals of T incident on c to $\mathcal{D}$. Next, for each reflex vertex r that has at least two diagonals incident on it, we add to $\mathcal{D}$ the set of all diagonals of T that are incident on r and have not been already added to $\mathcal{D}$.

It is clear that this is a partition since no diagonal is added in two different subsets. We claim that the partition satisfies the following two properties: (a) each D_i is either a reflex fan or a convex fan, and (b) for any $i < j$, all diagonals of D_i occur to the left of all

diagonals of D_j. It is clear that each subset is a fan. To see why (b) holds, consider two subsets D and D' of $\mathcal{D}$. If both are convex fans or both are reflex fans, then clearly one of them lies completely to the left of the other. Thus suppose D is a reflex fan and D' is a convex fan that shares the convex vertex c'. Let $d' \in D'$ occur between $d_i, d_j \in D$. Then it is clear that T contains at most one diagonal incident on c' and thus at no point would the procedure above have added D' to $\mathcal{D}$.

We show that insertions can be performed using $O(n)$ flips by showing that (a) for any D_i, an insertion between two diagonals of D_i can be performed with $O(|D_i|)$ flips, and (b) for any D_i and D_{i+1}, an insertion from D_{i+1} to D_i can be performed with $O(|D_i| + |D_{i+1}|)$ flips. These two together give us the required bound.

Lemma 5.3. *For each D_i, insertion between two edges of D_i can be performed using $O(|D_i|)$ flips.*

Proof. We consider the two cases of D_i being a convex fan and a reflex fan one by one.

If D_i is a reflex fan (Figure 5.3(a)), let r be its common reflex vertex and $C_{jj'}$ be the set of convex vertices. For any two consecutive diagonals $rc_{j''}$ and $rc_{j''+1}$, if the diagonals $rc_{j''-1}$ and $rc_{j''+2}$ also lie in D_i, then we claim that we can swap the labels of $rc_{j''}$ and $rc_{j''+1}$ with $O(1)$ flips. Consider the pentagon formed by r and the chain $c_{j''-1} \dots c_{j''+2}$. Since $\mathcal{P}$ is locally convex, this pentagon must be empty and convex. Thus we can swap $rc_{j''}$ and $rc_{j''+1}$ with five flips (Figure 4.1(b)).

If $rc_{j''-1}$ and $rc_{j''+2}$ do not lie in D_i, we claim that they must either lie in T or be a boundary edge of $\mathcal{P}$. If D_i is D_1, then the left-most diagonal of D_i is rc_2 and since rc_1 is a boundary edge of $\mathcal{P}$, we are done. Similarly, if $i = t$, we are also done. Otherwise, let d be the right-most diagonal of D_i and d' be the left-most diagonal of D_{i+1}. Since no diagonal of T lies between d and d', it is clear that they share a common endpoint. If the shared endpoint is a convex vertex c, then c has at least the two diagonals d and d' incident on it and thus the procedure for constructing $\mathcal{D}$ would have put both d and d' into the convex fan corresponding to c. Thus d and d' must share the reflex vertex r and therefore it can be used as $rc_{j''+2}$. Similarly, we can also show the existence of $rc_{j''-1}$. Thus the pentagon formed by r and the chain $c_{j''-1} \cdots c_{j''+2}$ is convex and empty and can be used to swap $rc_{j''}$ and $rc_{j''+1}$.

This shows that any two adjacent diagonals of a reflex fan can be swapped using five flips. By performing a sequence of adjacent swaps, we can perform any insertion with $O(|D_i|)$ flips.

Next, for the case when D_i is a convex fan, let c_j be the common convex vertex and $R_{ii'}$ be the set of reflex vertices. This means c_j was chosen to be the apex vertex for each

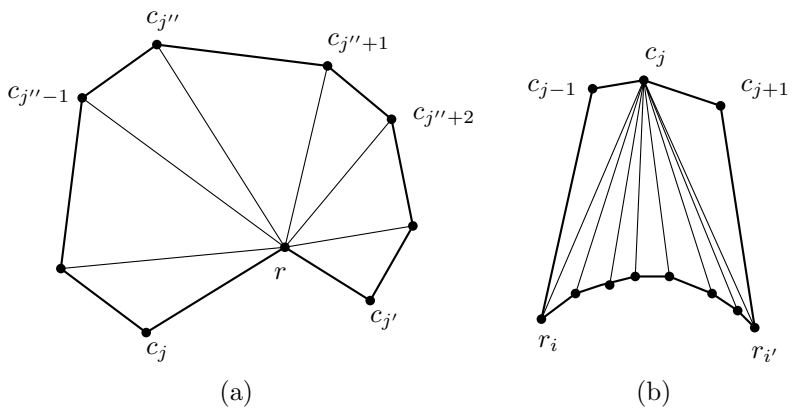


Figure 5.3: (a) A reflex fan, (b) A convex fan.

edge of $R_{i,i'}$ during the construction of T. Using the fact that the apex is always chosen to be in the "middle" of the set of visible vertices and Lemma 5.2, we get that c_{j-1} and c_{j+1} must exist and be visible from all vertices of $R_{i,i'}$. Since D_i consists of all the diagonals incident on c_j, the diagonals $r_i c_{j-1}$ and $r_i c_{j+1}$ must exist. These two diagonals together with the chains $C_{j-1,j+1}$ and $R_{i,i'}$ form a polygon that looks like the one in Figure 5.3(b). Below we describe how to perform insertions in this polygon.

Let the diagonals of this polygon be labelled $\lambda_1, \ldots, \lambda_s$ from left to right. Figure 5.4(a) shows how to insert λ_s (dotted) at λ_1 (dashed) in three steps each involving at most $|D_i|$ flips. For inserting an arbitrary λ_{i_2} at λ_{i_1}, we first perform a flip sequence to get the triangulation in Figure 5.4(b) and then use the strategy above inside the smaller polygon. □

Next, we show how to do insertions from D_{i+1} to D_i. Let d be the right-most diagonal of D_i and d' the left-most diagonal of D_{i+1}. Note that it is sufficient to describe how to swap the labels of d and d' because that combined with insertions within the D_i's is enough to perform arbitrary insertions.

Lemma 5.4. *For any D_i and D_{i+1}, let d be the right-most diagonal of D_i and d' be the left-most diagonal of D_{i+1}. Then the labels of d and d' can be swapped with at most $O(|D_i| + |D_{i+1}|)$ flips.*

Proof. We need to consider four cases. The case when D_i and D_{i+1} are both convex fans is shown in Figure 5.5(a) and the case when D_i is convex and D_{i+1} is reflex is shown in Figure 5.5(b). In both cases, let d and d' share vertex r. Our strategy is to perform a

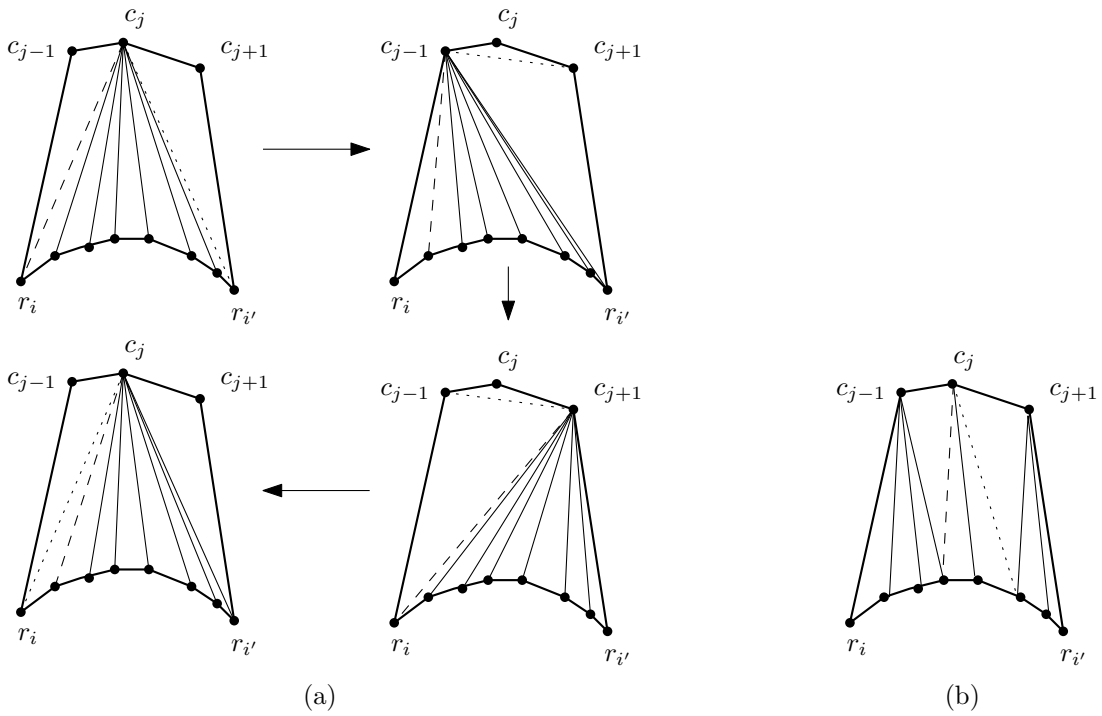


Figure 5.4: Insertions in convex fans. Let $\lambda_1, \ldots, \lambda_s$ be the labels from left to right. (a) Inserting λ_s (dotted) at λ_1 (dashed): the three steps represent flip sequences $\lambda_1 \ldots \lambda_s, \lambda_{s-1} \ldots \lambda_1$, and λ_s followed by $\lambda_1 \ldots \lambda_{s-1}$ respectively. (b) Inserting arbitrary λ_{i_2} (dotted) at λ_{i_1} (dashed): flip sequence $\lambda_1 \ldots \lambda_{i_1-1}$ followed by $\lambda_s \ldots \lambda_{i_2+1}$ results in the triangulation shown. Then use part (a).

linear number of flips in the convex fans to get an empty pentagon $efghr$ where d and d' can be swapped.

The case when D_{i+1} is convex and D_i is reflex is symmetric to the case when D_i is convex and D_{i+1} reflex. Finally, it is not possible to have both D_i and D_{i+1} reflex. Suppose, for contradiction, that they are both reflex and let d be the right-most diagonal of D_i and d' be the left-most diagonal of D_{i+1}. Clearly, d and d' share an endpoint, which cannot be a reflex vertex, otherwise all diagonals of $D_i \cup D_{i+1}$ would share the same endpoint and they would not be in different sets. If d and d' share a convex vertex c, then that convex vertex is incident to at least two diagonals and hence our procedure for constructing T would have put them both into the convex fan corresponding to c. □

This completes the proof that (1) $\implies$ (3). The last step in the proof of Theorem 5.2 is to show (2) $\implies$ (1). We show that in the contrapositive. That is, we show that if $\mathcal{P}$ is not locally convex, then QG is not connected. The following lemma is useful.

Lemma 5.5. *Let $c_j, c_{j+1}, c_{j+2}, c_{j+3}$ be four consecutive vertices on C such that the quadrilateral $c_jc_{j+1}c_{j+2}c_{j+3}$ is not empty. Let r_i be the right-most vertex of R that is visible from c_j. Then r_i must lie inside the quadrilateral $c_jc_{j+1}c_{j+2}c_{j+3}$.*

Proof. Consider the ray starting at c_j and extending in the direction of c_jc_{j+1} and rotate it clockwise around c_j (Figure 5.6). Let r_i be the first vertex inside the quadrilateral $c_jc_{j+1}c_{j+2}c_{j+3}$ that it hits. If it hits more than one vertices of R simultaneously, then pick the left-most one to be r_i. It is clear that r_i can see c_j but r_{i+1} cannot. Thus r_i must be the right-most vertex of R visible to c_j. □

Lemma 5.6. *If there exist four consecutive vertices $c_j, c_{j+1}, c_{j+2}, c_{j+3}$ on C such that the quadrilateral $c_jc_{j+1}c_{j+2}c_{j+3}$ is not empty, then there exist three spiral polygons $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ such that:*

1. *Each boundary edge of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ is either a boundary edge or a diagonal of $\mathcal{P}$.*
2. *Each diagonal of $\mathcal{P}$ is a diagonal of exactly one of $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$.*
3. *No diagonal of $\mathcal{P}_s$ is connected by an edge in QG to a diagonal of $\mathcal{P}_t$ for $s \neq t$.*

Proof. Let $\mathcal{P}_1$ be the spiral polygon defined by the chains $C_{1,j+2}$ and $R_{1,i}$ (Figure 5.6). The definition of $\mathcal{P}_2$ and $\mathcal{P}_3$ depends on whether c_{j+1} is visible from r_{i+1}. If it is, let $\mathcal{P}_3$ be the polygon defined by $C_{j+1,m}$ and $R_{i+1,k}$ and let $\mathcal{P}_2$ be the quadrilateral $c_{j+1}c_{j+2}r_{i+1}r_i$; otherwise let $\mathcal{P}_3$ be the polygon defined by $C_{j+1,m}$ and $R_{i,k}$ and let $\mathcal{P}_2$ be the empty polygon.

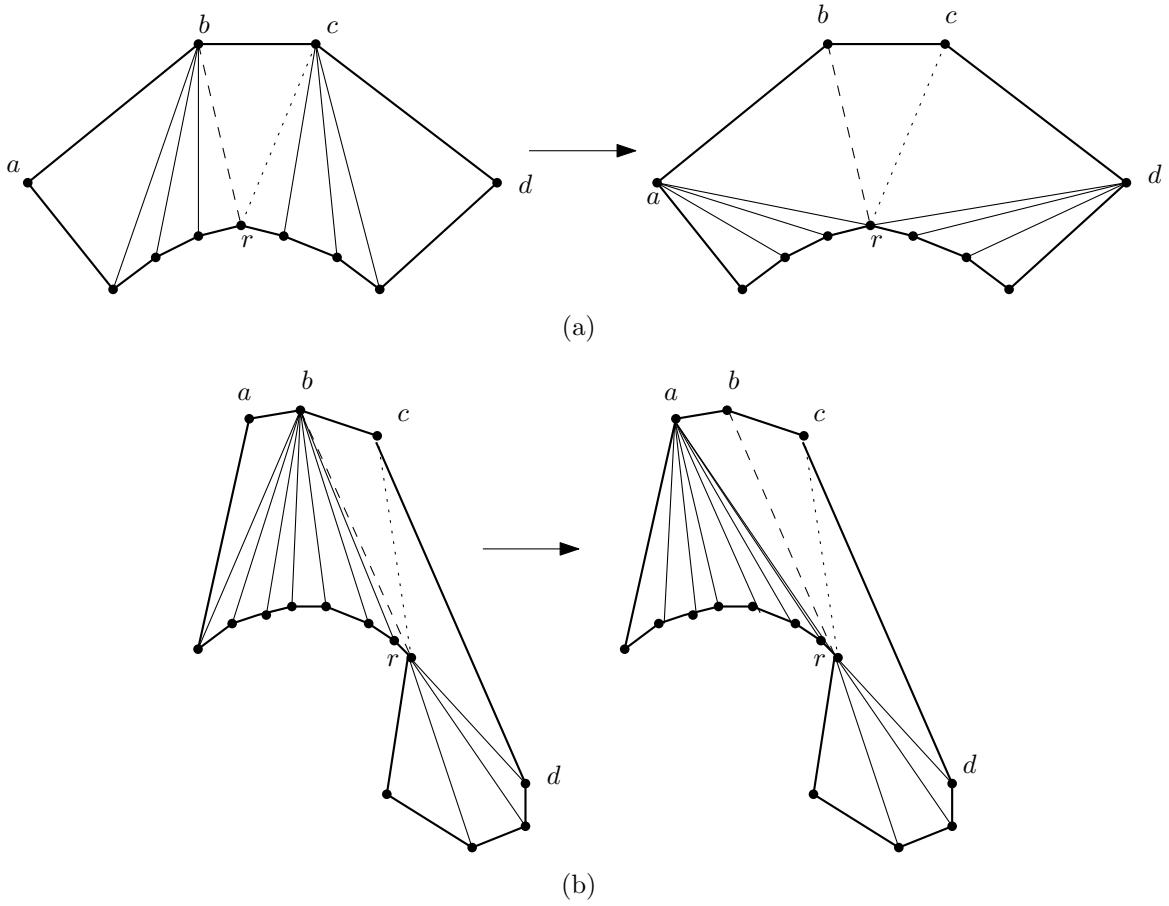


Figure 5.5: Swapping labels between adjacent fans D_i and D_{i+1} using the pentagon $abcdr$. (a) D_i and D_{i+1} are both convex fans. (b) D_i is a convex fan but D_{i+1} is a reflex fan.

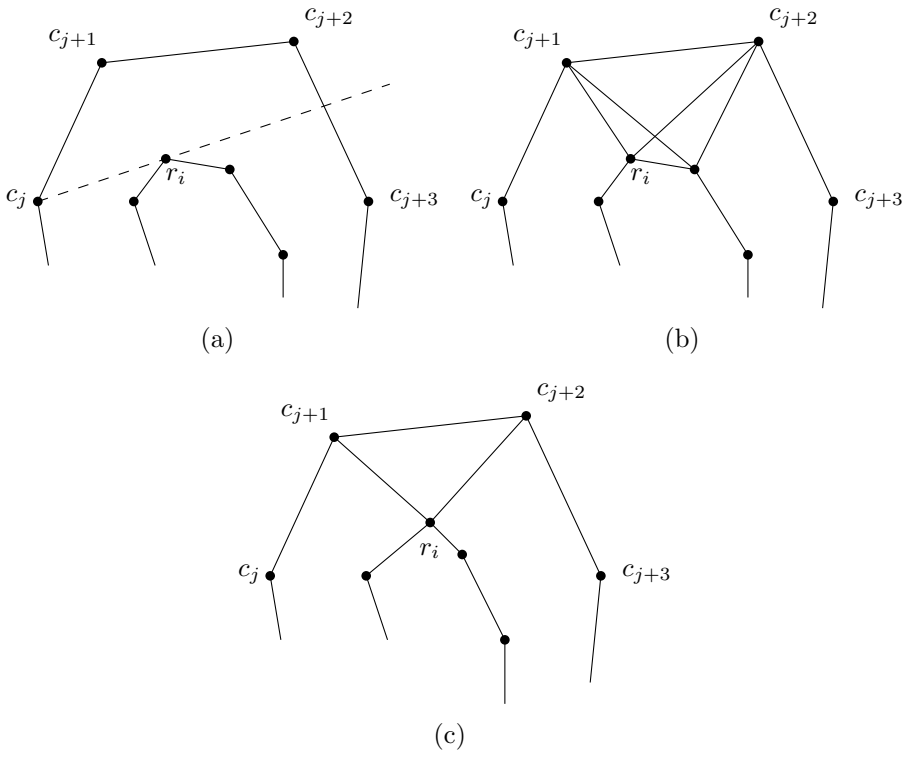


Figure 5.6: (a) Proof of Lemma 5.5. (b) Decomposition into polygons (Lemma 5.6) if r_{i+1} is visible from c_{j+1}. (c) Decomposition if r_{i+1} is not visible from c_{j+1}.

With these definitions, it is easy to check that conditions (1) and (2) above are satisfied. For (3), suppose, for contradiction, that there are diagonals d_s of $\mathcal{P}_s$ and d_t of $\mathcal{P}_t$ for $s \neq t$ such that they are connected by an edge in QG. This means d_s and d_t must cross, which is possible only if d_s is incident on c_{j+1} and d_t is incident on c_{j+2}. However, with this constraint, whatever we pick the other endpoints of d_s and d_t to be, the convex quadrilateral formed by their four endpoints will always contain r_i. □

This concludes the proof of Theorem 5.2. □

Theorem 5.2 gives us a way to decide, in polynomial time, whether a given spiral polygon has a connected flip graph: for each quadrilateral formed by four consecutive vertices of the convex chain, check if it is empty. A naive implementation of this procedure will take $O(n^2)$ time: there are $O(n)$ quadrilaterals to check and for each quadrilateral, there are at most n points that could potentially lie inside it. However, we can reduce the running time to $O(n)$ using Lemma 5.6 since it provides, for each quadrilateral $c_j c_{j+1} c_{j+2} c_{j+3}$, a certificate in the form of exactly one point—namely, the right-most point of R that is visible from c_j—such that if the quadrilateral is non-empty, it must contain that point. Moreover, Property 5.3 implies that if quadrilateral Q lies to the left of Q' on the convex chain C, then the certificate of Q lies to the left of the certificate of Q' on the reflex chain R. This observation directly gives us a way to find the certificate for each quadrilateral in linear time. Thus we get the following theorem.

Theorem 5.3. *Given a spiral polygon $\mathcal{P}$, we can decide in linear time whether the flip graph of the edge-labelled triangulations of $\mathcal{P}$ is connected.*

5.3.2 Connectivity between two given triangulations

We now turn out attention to the problem of deciding whether there exists a path in the flip graph between two given edge-labelled triangulations of a spiral polygon.

Lemma 5.7. *Given any spiral polygon $\mathcal{P}$, there exists a set $\{\mathcal{P}_1, \ldots, \mathcal{P}_t\}$ of spiral polygons such that:*

1. *For each $\mathcal{P}_i$, each boundary edge of $\mathcal{P}_i$ is either a boundary edge or a diagonal of $\mathcal{P}$.*
2. *Each diagonal of $\mathcal{P}$ is a diagonal of exactly one of $\mathcal{P}_1, \ldots, \mathcal{P}_t$.*
3. *No diagonal of $\mathcal{P}_i$ is connected by an edge in QG to a diagonal of $\mathcal{P}_j$ for $i \neq j$.*

4. Each $\mathcal{P}_i$ is locally convex.

Proof. If $\mathcal{P}$ is already locally convex, we are done. Otherwise $\mathcal{P}$ has four consecutive vertices on its convex chain that form a non-empty quadrilateral. Using Lemma 5.6, we get three polygons on which we recurse to obtain the desired set $\{\mathcal{P}_1, \ldots, \mathcal{P}_t\}$. □

The lemma combined with Theorem 5.2 directly gives us the following.

Theorem 5.4. *Given a spiral polygon $\mathcal{P}$ and two of its edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$, there exists a flip sequence that transforms $\mathcal{T}_1$ to $\mathcal{T}_2$ if and only if for each label λ, the diagonal of $\mathcal{T}_1$ with label λ is in the same connected component of $QG_{\mathcal{P}}$ as the diagonal of $\mathcal{T}_2$ with label λ. Moreover, if such a sequence exists, it is of length at most $O(n^2)$.*

Proof. The 'only if' direction is easy. For the 'if' direction, we provide a flip sequence.

First ignore labels and transform both $\mathcal{T}_1$ and $\mathcal{T}_2$ into the same unlabelled triangulation T. Then rearrange the labels in T. If the diagonal with label λ in $\mathcal{T}_1$ was in the same component of QG as the diagonal with label λ in $\mathcal{T}_2$, they must be diagonals of the same polygon $\mathcal{P}_i$. But since each $\mathcal{P}_i$ is locally convex, any rearrangement inside it can be carried out with at most $O(|\mathcal{P}_i|^2)$ flips. □

It is easy to compute the decomposition of Lemma 5.7 in $O(n)$ time using a slight modification of the algorithm of Theorem 5.3. Using the decomposition, we can also check, in linear time, whether for each label λ, the diagonals of $\mathcal{T}_1$ and $\mathcal{T}_2$ with that label lie in the same polygon of the decomposition. This gives us the following.

Theorem 5.5. *Given a spiral polygon $\mathcal{P}$ and two of its edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$, one can decide in $O(n)$ time whether there exists a sequence of flips that transforms $\mathcal{T}_1$ to $\mathcal{T}_2$.*

Finally, note that our quadratic bound on the flip distance is tight. Consider, for example, a locally convex spiral polygon for which every reflex vertex r has $|V(r)| \leq 4$. Let $d_1, \ldots, d_n$ be the diagonals of the canonical triangulation of this polygon listed left-to-right. It is clear that the distance in QG between any d_i and d_j is $\Omega(|j - i|)$. Thus there are $\Omega(n^2)$ pairs of diagonals at distance $\Omega(n^2)$ in QG. Since any flip sequence that moves a label from d_i to d_j must contain at least $\Omega(|j - i|)$ flips, this gives a lower bound of $\Omega(n^2)$.

Theorem 5.6. *There exists a spiral polygon that has a connected flip graph whose diameter is $\Omega(n^2)$.*

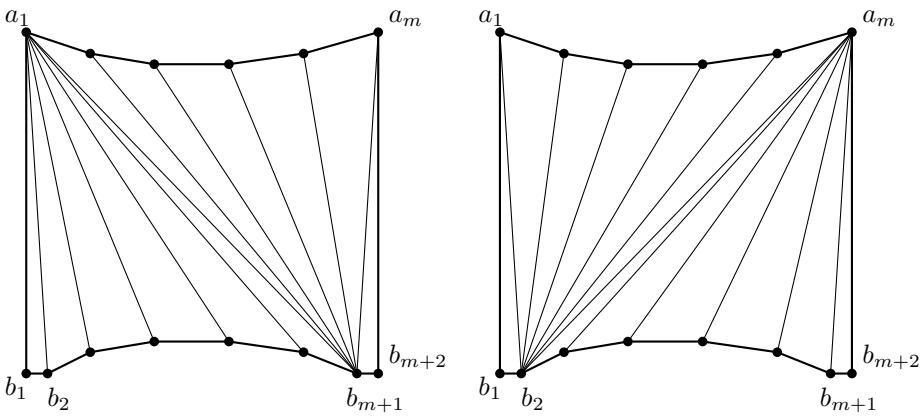


Figure 5.7: (a) A left-inclined augmented channel. (b) A right-inclined augmented channel.

5.4 More general polygons and planar point sets

In this section, we provide an example of a polygon with two reflex chains where the diameter of the flip graph is $\Theta(n^3)$. The example also demonstrates a difficulty in generalizing our results on spiral polygons to more general cases. In particular, for the case of spiral polygons, Lemma 5.7 shows that whether two diagonals are connected or not can be determined by looking at the region "in between" them. The example in this section fails to satisfy this property, i.e., even to be able to decide whether two adjacent diagonals are connected we need to look at the entire polygon.

We call our example an *augmented channel* since it is obtained by adding two new vertices to the channel of Chapter 2. An augmented channel consists of two chains $A = a_1 \dots a_m$ and $B = b_1 \dots b_{m+2}$, as shown in Figure 5.7, such that:

- Every vertex on A is visible from every vertex on B.
- The vertices on A form a convex chain.
- All vertices on B, except for b_1, b_2, b_{m+1}, and b_{m+2} are reflex.

The difference between an augmented channel and a channel is the presence of the two convex vertices b_2 and b_{m+1} on B.

Theorem 5.7. *The diameter of the flip graph of an augmented channel is* $\Theta(n^3)$.

Proof. We first show the upper bound.

Note that any triangulation of the augmented channel induces an ordering on its diagonals. If all diagonals connect a vertex on A with a vertex on B, then the ordering induced is the same as the order in which a ray passing through the polygon from left to right will hit each diagonal. If, however, one or both of the diagonals b_1b_3 and b_mb_{m+2} are present, then we assign to the diagonals the same ordering as we would in the triangulation obtained on flipping the ones that are present. The ordering on the diagonals defines an ordering on the labels.

Note also that given an ordering on the labels, all triangulations that induce that ordering can be transformed into each other with $O(n^2)$ flips (using Theorem 3.8 from Hurtado et al. [52]). Thus for the $O(n^3)$ bound on the diameter of the flip graph, the challenge is in rearranging the labels. For any triangulation, let D_1 denote the set consisting of the first m labels and D_2 denote the set consisting of the last m labels. Since there are $2m-1$ labels in total, D_1 and D_2 will have one label in common, namely, the m^{th} label. Finally, note that in the left-inclined triangulation (Figure 5.7(a)), D_2 is exactly the set of diagonals incident on b_{m+1} and thus forms a convex fan. Similarly, in the right-inclined triangulation (Figure 5.7(a)), D_1 forms a convex fan. This means both D_1 and D_2 can be sorted in $O(n^2)$ flips and any insertion in D_1 or D_2 can be performed in $O(n)$ flips using Lemma 5.3.

Thus our strategy is as follows. Given $\mathcal{T}_1$ and $\mathcal{T}_2$, first ignore labels and transform both into the right-inclined triangulation. Then, if D_1 contains a label bigger than m, insert it at b_2a_m and transform the triangulation into a left-inclined triangulation while preserving the order of labels. Next, if D_2 contains a label smaller than m, insert the label at a_1b_{m+1} and transform back into the right-inclined triangulation and repeat until D_1 only contains labels $1,\ldots,m$ and D_2 only contains labels $m,\ldots,2m-1$. Finally, sort both D_1 and then sort D_2. Since transforming between the left-inclined and right-inclined triangulations takes $O(n^2)$ flips and insertion inside D_1 or D_2 takes $O(n)$ flips, we get a bound of $O(n^3)$ flips.

To show the lower bound, we use the observation that to move a label from $D_2 \setminus D_1$ to $D_1 \setminus D_2$, we must first move that label from $D_2 \setminus D_1$ to the m^{th} diagonal and then to $D_1 \setminus D_2$. The first step can be performed only if the diagonal a_1b_{m+1} is present and the second step can be carried out only if the diagonal b_2a_m is present. Going from a triangulation that has the diagonal a_1b_{m+1} to a triangulation that has the diagonal b_2a_m requires $\Omega(n^2)$ flips using the argument from Hurtado et al. [52] or Property 2.1 from Chapter 2. Thus if we want to transform a triangulation with $D_1 = \{m,\ldots,2m-1\}$ and $D_2 = \{1,\ldots,m\}$ to a triangulation with $D_1 = \{1,\ldots,m\}$ and $D_2 = \{m,\ldots,2m-1\}$, we will need at least $\Omega(n^3)$

flips. □

The theorem above demonstrates that deciding whether, say, a_1b_m and a_1b_{m-1} are connected depends on the exact position of b_2 and b_{m+1} since in the example above, they are connected, but if b_2 and b_{m+1} were reflex vertices then they would not be.

Thus the problem of deciding the connectivity of the flip graph for general polygons and point sets seems to be tricky. However, we conjecture that even in the most general case, the flip graph is connected if and only if the quadrilateral graph is connected.

Chapter 6

Reconfiguring Ordered Bases of a Matroid[1]

In this chapter we study reconfiguring bases of a matroid when the basis elements are assigned labels. Reconfiguring unlabelled bases was studied by Ito et al. [55]; their main message was that unlabelled bases are easy to analyze. Well known properties of matroids show that the reconfiguration graph is always connected and that its diameter is $O(n)$. To the best of our knowledge, reconfiguring labelled bases has not been considered before, but the idea of labelled bases has appeared by the name of *ordered* bases.

As in the case of triangulations of planar point sets and simple polygons, the reconfiguration graph of ordered bases is not always connected. Our main result gives a complete characterization of reconfigurability, i.e., given two ordered bases, we provide a means of testing whether one can be transformed to another. We also provide bounds on the total number of flips needed in the worst case. For general matroids, we show an upper bound of $O(n^{1.5})$ where n is the rank of the matroid. For graphic matroids, we show a better bound of $O(n \log n)$.

6.1 A quick primer on matroids

The material in this section and in Section 6.3.3 is based on the exposition in the textbook by Oxley [80].

[1] This chapter represents joint work with Jim Geelen.

Matroids are set systems that were originally defined to study an abstract theory of dependence. Given an $m \times n$ matrix, define the set system $M = (E, \mathcal{I})$ such that $E = [1..n]$ and $I \in \mathcal{I}$ if and only if the set of columns corresponding to I forms an independent set of vectors. This set system is an example of a matroid.

Definition 6.1. *A matroid is a tuple $M = (E, \mathcal{I})$ where E is a set and $\mathcal{I}$ is a set of its subsets such that:*

(I1) $\emptyset \in \mathcal{I}$.

(I2) If $I \in \mathcal{I}$ *and* $I' \subseteq I$ *then* $I' \in \mathcal{I}$.

(I3) If I_1 *and* I_2 *are in* $\mathcal{I}$ *and* $|I_2| > |I_1|$ *then there exists an element* $e \in I_2 - I_1$ *such that* $I_1 \cup \{e\} \in \mathcal{I}$.

It is easy to see that these three properties are satisfied for the set system formed by independent sets of column vectors of a matrix. The sets of $\mathcal{I}$ are called *independent sets* and property (I3) above is often called the *independence augmentation* property. Many naturally-occurring set systems, often in unrelated areas, happen to be matroids. Since the linearly independent subsets of columns of a matrix form the independent sets of a matroid, it is apparent that sets of affinely independent points should also form a matroid. For example, given a set of points in a plane, the set of subsets of size at most two and subsets of three non-collinear points forms a matroid. A popular matroid that appears in graph theory is the one formed by the set of forests of a graph. Given a graph G, define the matroid $M = (E, \mathcal{I})$ where E conisists of all edges of G and $I \in \mathcal{I}$ if and only if the edges corresponding to I form a forest (i.e., do not contain a cycle) in G. A matroid formed this way is called a *graphic matroid*.

A *maximal independent set* of a matroid $M = (E, \mathcal{I})$ is a set $I \in \mathcal{I}$ such that no superset of I is a member of $\mathcal{I}$. A maximal independent set is also called a *basis* (note the correspondence with bases of a vector space). The set of all bases is often denoted by the letter $\mathcal{B}$. The following two properties of bases are easy to show.

Property 6.1. *If B_1 and B_2 are both bases of a matroid $M = (E, \mathcal{I})$ then $|B_1| = |B_2|$.*

Proof. For contradiction, assume $|B_1| < |B_2|$. Then from property (I3), there exists an element $e \in B_2 - B_1$ such that $B_1 \cup \{e\} \in \mathcal{I}$. Thus B_1 was not maximal. □

Property 6.2. *For any two bases B_1 and B_2 of a matroid $M = (E, \mathcal{I})$ and an element $x \in B_1 - B_2$, there exists $y \in B_2 - B_1$ such that $(B_1 - \{x\}) \cup \{y\}$ is also a basis.*

Proof. Using property (I3) on $B_1 \setminus \{x\}$ and B_2, we directly get the desired $y \in B_2 - B_1$. □

Property 6.1 helps one define a useful quantity—the *rank* of a matroid, denoted $r(M)$—as $r(M) = |B|$ for any basis B. Property 6.2, which we will call the *basis exchange property* from now on, already contains hints of reconfiguration. Indeed, given a matroid $M = (E, \mathcal{I})$ and a basis B, define a reconfiguration step as replacing an element $e \in B$ with an element $e' \in E \setminus B$ such that the new set is also a basis. This defines a reconfiguration graph that has a vertex for each basis and edge for each reconfiguration step. The following is an easy consequence of Property 6.2.

Theorem 6.1. *The reconfiguration graph of bases of a matroid M is connected and has diameter at most $r(M)$.*

Proof. Given any two bases B_1 and B_2 of M, one application of the basis exchange property increases $|B_1 \cap B_2|$ by one. Since the minimum and maximum possible values of $|B_1 \cap B_2|$ are 0 and $r(M)$ respectively, the total number of applications of Property 6.2 required to transform B_1 into B_2 in the worst case is $r(M)$. □

Matroid reconfiguration was first considered by Ito et al. [55] and the theorem above is from that paper, although it is a well known result in the literature on matroids.

A basis of a matroid has properties similar to a spanning tree of a graph. One can also define a *circuit* of a matroid to have properties similar to a cycle of a graph. A circuit of a matroid is defined as a minimal non-independent set, i.e., a set $C \subseteq E$ which is not independent but every subset of C is independent. The set of circuits of a matroid satisfies some nice properties, including the following *circuit exchange* property.

Property 6.3. *Given any two circuits C_1 and C_2, an element $e \in C_1 \cap C_2$, and an element $f \in C_2 - C_1$, there exists another circuit C_3 such that $f \in C_3 \subset C_1 \cup C_2 - \{e\}$.*

Circuits help us state the basis exchange property in a way that gives us more control.

Property 6.4. *Given a basis B, and an element $e \notin B$, adding e to B creates a unique circuit C and removing any element of C then gives us another basis B'.*

Proof. First of all, suppose, for contradiction, that adding e to B creates more than one circuit, C and C' being any two of them. Clearly, e must be a part of both, and therefore, from Property 6.3, there exists another circuit $C'' \subseteq C \cup C' - \{e\}$. This is not possible since $C \cup C' - \{e\} \subseteq B$ and therefore is independent.

Given that C is unique, it is also clear that removing any element of C will give us a basis. □

6.2 Matroids and triangulations

The set of triangulations of a point set (or a simple polygon) has some matroid-like properties. In particular, given a point set P, let E be the set of all line segments d such that the endpoints of d are in P and no other points of P lie on d and let $\mathcal{I}$ be the set of all subsets of non-crossing segments of E. Then it is clear that the set system $(E, \mathcal{I})$ satisfies properties (I1) and (I2) but not property (I3). However, the sets of maximal sets in $\mathcal{I}$ do satisfy the property that if B_1 and B_2 are both maximal, then $|B_1| = |B_2|$, because maximal sets of non-crossing segments correspond to triangulations and all triangulations contain the same number of edges.

The absence of property (I3) is costly, since it plays a crucial role in proving the correctness of certain greedy algorithms on matroids. For example, consider the task of computing the minimum spanning tree of a weighted graph. The age-old Kruskal's algorithm does the trick. However, the greedy algorithm does not find a minimum-weight triangulation of a point set, and, for Euclidean weights, the problem was finally proved NP-hard in 2008 [78].

In this chapter, we study reconfiguration of ordered (labelled) bases of a matroid, which may or may not provide new insight about the question of reconfiguring labelled triangulations from the previous chapter. In light of the discussion above, there is evidence both for and against the hypothesis that matroid reconfiguration may be related to triangulation reconfiguration. In any case, for the purpose of studying labelled reconfiguration, matroids are a good starting point since the unlabelled case of reconfiguration is very easy for matroids.

6.3 Reconfiguring ordered bases of a matroid

6.3.1 Problem statement

An ordered basis of a matroid is defined in the same way as an edge-labelled triangulation, i.e., an ordered basis $\mathcal{T} = (B, l)$ is a tuple where B is a basis and $l : B \to \{1, \ldots, |B|\}$ is a function that assigns a unique label to each element of B. A reconfiguration step is now a basis exchange operation where the new element gets the same label as the replaced element, i.e., if $e \in B$ is replaced with $e' \in E \setminus B$, then e' gets assigned the label $l(e)$. Two bases are said to be the same if they have the same elements and the elements are assigned the same labels. From now on, we will use *flip* to denote one reconfiguration step

and *flip graph* to denote the reconfiguration graph. Since the labels are unique, we will frequently use the labels to refer to the elements they are assigned to. If a flip sequence F transforms the ordered basis $\mathcal{T}_1$ to $\mathcal{T}_2$ such that label i is assigned to e_1 in $\mathcal{T}_1$ and to e_2 in $\mathcal{T}_2$, we say that F *moves* label i from e_1 to e_2. For label i, the element with that label is given by $l^{-1}(i)$. When we are talking specifically about graphic matroids, we will use the terms spanning tree and edge instead of basis and element respectively.

It is easy to show now that the flip graph is not always connected. For example, let G be a path on n vertices. Clearly, G has only one spanning tree, which is G itself. Now this spanning tree gives us several ordered bases depending on how we label its edges. However, none of the elements of this basis can be flipped and thus any ordered basis of G cannot be reconfigured to any other ordered basis.

This observation makes the following questions interesting:

1. What properties guarantee that we can reconfigure one ordered basis to another?
2. What is the number of reconfiguration steps needed in the worst case?

6.3.2 Graphic matroids

When are two ordered bases reconfigurable?

In this section, we provide a polynomial time algorithm that takes a graph and two labelled spanning trees as its input and decides whether there exists a reconfiguration sequence that transforms one to the other.

Theorem 6.2. *Given two labelled spanning trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of a graph G, we can reconfigure one to the other if and only if for each label i, the edge with that label in $\mathcal{T}_1$ and the edge with that label in $\mathcal{T}_2$ lie in the same 2-connected component of G. Moreover, the flip sequence, if it exists is of length at most $O(n^2)$.*

Proof. The 'only if' direction is clear because any flip of edge e to e' can be performed only if both e and e' lie in the same 2-connected component. For the 'if' direction, we will provide an explicit flip sequence to transform $\mathcal{T}_1$ to $\mathcal{T}_2$.

First, pick any (unlabelled) spanning tree B and transform both $\mathcal{T}_1$ and $\mathcal{T}_2$ to B while ignoring the labels. Let F_1 be the sequence that transforms $\mathcal{T}_1$ to B and F_2 be the sequence that transforms $\mathcal{T}_2$ to B. Obviously, the labels of the edges of B obtained from the two sequences will not match in general. We show, below, a flip sequence F of length at most

$O(n^2)$ to rearrange the labels in B. Thus performing F_1 followed by F followed by the reverse of F_2 transforms $\mathcal{T}_1$ to $\mathcal{T}_2$.

Thus the problem is now reduced to the following: given one spanning tree B and two labelled spanning trees $\mathcal{T}_1 = (B, l_1)$ and $\mathcal{T}_2 = (B, l_2)$, we want to transform $\mathcal{T}_1$ to $\mathcal{T}_2$. We do that by repeatedly swapping labels. Thus for any label i for which the edge with that label in $\mathcal{T}_1$ is different from the edge with that label in $\mathcal{T}_2$, we swap the labels in $O(n)$ flips. Let $e = l_1^{-1}(i)$ and $e' = l_2^{-1}(i)$. Since e and e' lie in the same 2-connected component of G, there must exist a cycle C of G that goes through both e and e'. If there exists only one edge of C that does not lie in B, then we are done: let that edge be e''; use Property 6.4 to replace e with e'', e' with e, and, finally, e'' with e'. This sequence swaps the labels of e and e'. Now suppose there exists another edge f of C that does not lie in B. Adding f to B creates a cycle C' that must contain an edge $f' \notin C$. Thus replacing f' with f in B increases the number of edges of C that currently lie in the spanning tree by one and does not move the label i. Repeating this at most $n - 1$ times, we will have a basis that contains every edge of C except one. In this basis we can swap e to e'. □

Tightening the bounds

We show, in this section, that the $O(n^2)$ bound on the worst-case flip distance from the previous section can be improved. Note that the common spanning tree B we chose in the previous section to flip both $\mathcal{T}_1$ and $\mathcal{T}_2$ to was chosen completely arbitrary. We could have perhaps chosen a spanning tree that made the task of swapping labels easier. That is precisely what we do in this section.

For any spanning tree B of G, its *fundamental graph* S_B is defined as the bipartite graph that has a vertex for each edge of G and an edge between u and v if and only if:

1. The edge corresponding to u lies in B.
2. The edge corresponding to v lies in $E \setminus B$.
3. Replacing the edge corresponding to u with the edge corresponding to v in B is a valid flip. In other words, adding v to B creates a circuit that contains u.

Let the diameter of S_B be d. We claim that we can perform any swap in B with a flip sequence of size at most $O(d)$. We can show this by induction. For two edges e and e' of G that lie in the same 2-connected component, let e_1 and e_2 be the two vertices that occur immediately after e on the path from e to e' in S_B. Replacing e with e_1 followed by

replacing e_2 with e and then e_1 with e_2 results in swapping e with e_2. Distance between e_2 and e' is now shorter and thus we can apply induction to swap e_2 with e'.

We obtain improved bounds on the diameter of the flip graph by showing that there exists a spanning tree B such that the diameter of S_B is at most $O(\log n)$. Note that it is sufficient to show this for a 2-connected graph since we can just repeat the argument inside each 2-connected component of the graph.

For a 2-connected graph G and a cycle C of its edges, we denote by G/C the graph obtained on contracting C, and by $E(G)$ the set of edges of G. Note that $E(G/C) = E(G) - E(C)$. Contracting C divides G into 2-connected components, or *blocks*, that are the maximal subgraphs of G/C that are 2-connected. Note that any two edges $e, e' \in E(G/C)$ are in different blocks if and only if all paths in G containing both e and e' pass through at least one vertex of C. We first show a lemma that helps us formulate an inductive argument.

Lemma 6.1. *Any 2-connected graph G with m edges contains a cycle C such that all blocks of G/C have at most $m/2$ edges.*

Proof. Let C be the cycle that minimizes the size of the largest block obtained upon contracting it. If all blocks in G/C are of size at most $m/2$, then we are done. Otherwise there exists a block H of size bigger than $m/2$. Some edges of $E(H)$ are incident on vertices of C in G; let those vertices be $\{v_1, \ldots, v_k\}$ in clockwise order along C. There are two paths between v_1 and v_2 along the cycle C in G, one clockwise and one counterclockwise. Let P be the one that is counterclockwise and thus contains all vertices of $\{v_1, \ldots, v_k\}$. There also exists a path P' between v_1 and v_2 that goes through the vertices of H. We define C'' to be $P \cup P'$. We claim that the size of the largest block of G/C'' is smaller than the size of the largest block of G/C, hence reaching a contradiction.

First, note that if H' is a block of G/C different from H, then for all $e \in H$ and $e' \in H'$, if e and e' lie in G/C'', then they must be in different components of G/C''. Suppose, for contradiction, that they lie in the same component. Then there exists a path that contains both e and e' but none of the vertices from P. But P contains all vertices of C that have an edge of H incident on them. Thus the path containing both e and e' cannot contain any vertex of C, which means e and e' were in the same block of G/C.

Now G/C'' contains two kinds of blocks: those that contain edges of H and those that do not. Blocks of the first kind must have size at most the size of H from the argument above. In fact, they must be strictly smaller than H since C'' contains at least one edge of H. Blocks of the second kind should also be smaller than H since at worst, such a block contains all edges of G/C that are not edges of H, which has at most $m/2$ edges. □

We are now ready to construct our spanning tree B.

Lemma 6.2. *Given a 2-connected graph G, there exists a spanning tree B such that the diameter of S_B is $O(\log n)$.*

Proof. We start with setting B to be an empty set and gradually add edges into it. The algorithm for constructing B proceeds in iterations. In the first iteration we find the cycle C of Lemma 6.1 such that all blocks of G/C are of size at most $m/2$. Next, we add all edges but one of C to B and contract C thus breaking the graph G into several blocks. In general, in any iteration, we start with the collection of blocks the previous iteration resulted in, contract a connected set of edges inside each block, and add some of the contracted edges to B. Each iteration reduces the number of vertices of G by contracting a set of edges. The algorithm terminates once the graph G is left with just one vertex.

Let H^i be a block at the beginning of iteration i, E^i be the edges of H^i that we contract in the i^{th} iteration, and H^{i+1} be one of the blocks that H^i gets decomposed into when we contract E^i. Then in iteration $i+1$, we pick E^{i+1}, i.e., the edges of H^{i+1} to be contracted, as follows. Let C^{i+1} be the cycle of Lemma 6.1 for H^{i+1}. Let c be the vertex of H^{i+1} corresponding to the contraction of E^i. From 2-connectivity of H^{i+1}, there must exist two edge-disjoint paths P_1^{i+1} and P_2^{i+1} (possibly empty) from c to two different vertices of C^{i+1} such that $P_1^{i+1} \cap C^{i+1} = P_2^{i+1} \cap C^{i+1} = \emptyset$—simply pick the minimal superset of C^{i+1} in H^{i+1} that makes c 2-connected with C^{i+1}. The set E^{i+1} then consists of all edges but one of C^{i+1} and all edges but one of $P_1^{i+1} \cup P_2^{i+1}$. This completes the description of the algorithm.

Since each iteration reduces the size of each block by at least half, it is clear that the algorithm terminates in at most $O(\log n)$ iterations. To see the correctness of the algorithm, first note that when the algorithm terminates, B spans all vertices of G: it is clear that the set of all edges contracted through the course of the algorithm spans G; and the set of edges we add to B in each iteration is chosen such that it spans the graph induced by the set of edges contracted in that iteration. Next, we claim that the algorithm maintains the following two invariants: 1) at the end of each iteration, the set B does not contain a cycle, 2) for each i, and the sets E^i and E^{i+1} as defined above, there is a path of length $O(1)$ in S_B between any edge of E^{i+1} and any edge of E^i. To see (1), note that the edges added to B during an iteration do not contain a cycle. Thus the only way they could create a cycle in B is by completing a cycle some of whose edges were already present in B. However, this cannot happen since after adding edges to B in each iteration, the algorithm contracts all those edges that could potentially complete a cycle and thus those edges are not available in any future iterations. To see claim (2), note that every edge of C^{i+1} is

connected in S_B to the omitted edge of C_{i+1}, and every edge of $P_1^{i+1} \cup P_2^{i+1}$ is connected to the omitted edge of $P_1^{i+1} \cup P_2^{i+1}$. Finally, from 2-connectivity, we know that the endpoints of P_1^{i+1} and P_2^{i+1} that lie in the graph induced by E^i are distinct. Consider any edge on the path between those two endpoints in E^i. That edge is connected to the omitted edge of $P_1^{i+1} \cup P_2^{i+1}$. This completes the proof. □

This gives us the following strengthened form of Theorem 6.2.

Theorem 6.3. *Given two labelled spanning trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of a graph G, we can reconfigure one to the other if and only if for each label i, the edge with that label in $\mathcal{T}_1$ and the edge with that label in $\mathcal{T}_2$ lie in the same 2-connected component of G. Moreover, the flip sequence, if it exists, is of length at most $O(n \log n)$.*

6.3.3 General matroids

Adding more tools to our matroidal toolbox

We would like to generalize the results from the previous section to more general matroids. However, we used many concepts about graphs without talking about their analogs in the world of matroids. The theory of matroids has very elegant generalizations for some graphic concepts and some graphic concepts have no matroidal analogs. In this section, we explore what can and cannot be generalized to matroids.

Our proof for graphic matroids relied on induction using edge contraction. Edge deletion and edge contraction are two commonly used operations for carrying out inductive arguments on graphs. Edge deletion has an obvious matroidal analog. Given a matroid $M = (E, \mathcal{I})$, deleting a set $F \subseteq E$ of edges results in the matroid $M' = (E', \mathcal{I}')$, denoted by $M \setminus F$ such that $E' = E \setminus F$ and for each $I \in \mathcal{I}$, we add the set $I' = I \setminus F$ to $\mathcal{I}'$. Edge contractions also have a matroidal analog that can be elegantly defined using matroid duals.

Given a matroid $M = (E, \mathcal{I})$, let $B^* = \{E - B \mid B \in B(M)\}$. Then it is known that B^* is the set of bases of a matroid on E. The matroid whose set of bases is B^* is called the *dual* of M and is denoted by M^*. One can then define edge contraction as follows. Given a matroid $M = (E, \mathcal{I})$ and a set $F \subseteq E$ of edges, the matroid obtained on contracting the edges of F is defined as $M/F = (M^* \setminus F)^*$, i.e., we delete the edges in the dual. This definition is justified because for a graphic matroid, this operation corresponds exactly to the operation of contracting the edges of the graph, i.e., if M is a graphic matroid defined on a graph G, then the set of forests of M/F is exactly the set of forests of the graph

obtained on contracting F in G. We will not prove this statement here, but a proof can be found, for example, in [80].

The concept of graph connectivity can also be generalized to matroids. For any matroid $M = (E, \mathcal{I})$, define the relation ψ by $e \,\psi\, f$ if and only if either $e = f$ or there exists a circuit of M that contains both e and f. It can be shown that the relation ψ is an equivalence relation and we say that each equivalence class defines a *connected component*, or a *block*.

Finally, one can also define the *fundamental graph* for a basis of a matroid by replacing 'cycle' with 'circuit' in the definition of the fundamental graph for spanning trees.

Solution for general matroids

With these definitions, many of the proofs from the previous section can be copied almost verbatim to give analogous proofs for matroids except Lemma 6.3. However, we can prove a matroidal analog of Lemma 6.3 with looser bounds, which finally leads to a worst-case upper bound of $O(n^{1.5})$ on the flip distance.

It is easy to check that every step of the proof of Theorem 6.2 goes through for matroids and thus we get the following theorem, which we state without proof.

Theorem 6.4. *Given two labelled bases $\mathcal{T}_1$ and $\mathcal{T}_2$ of a matroid M, we can reconfigure one to the other if and only if for each label i, the element with that label in $\mathcal{T}_1$ and the element with that label in $\mathcal{T}_2$ lie in the same block of M. Moreover, the flip sequence, if it exists is of length at most $O(n^2)$.*

In order to tighten the bound, we use the following lemma, as proved in [71] (Corollary 1.4).

Lemma 6.3. *Let C be the biggest cycle of a matroid M. Then the biggest cycle of M/C is strictly smaller than C.*

This gives us the following theorem.

Theorem 6.5. *Given two labelled bases $\mathcal{T}_1$ and $\mathcal{T}_2$ of a matroid M, we can reconfigure one to the other if and only if for each label i, the element with that label in $\mathcal{T}_1$ and the element with that label in $\mathcal{T}_2$ lie in the same block of M. Moreover, the flip sequence, if it exists is of length at most $O(n^{1.5})$.*

Proof. We show this by providing a basis B whose fundamental graph S_B has a diameter of $O(\sqrt{n})$. We do this using almost exactly the iterative algorithm of Lemma 6.2 with one difference: instead of picking the cycle of Lemma 6.3 for each block in each iteration, we just pick the biggest circuit of each block. We also need a matroidal analog of the union $P_1^{i+1} \cup P_2^{i+1}$ of paths. We claim that picking the minimal subset P^{i+1} of H^i that connects E^i and C^{i+1} in H^i does the trick. Finally, the elements of H^{i+1} that will get added to B are all but one elements of C^{i+1} and all but one elements of P^{i+1}. The algorithm terminates once no elements of the matroid are left to contract.

We first notice that the algorithm terminates in $O(\sqrt{n})$ iterations. This is because the number of possible iterations for which there exists a block containing a circuit of size $\Omega(\sqrt{n})$ can be at most $O(\sqrt{n})$ and once the size of the biggest circuit in each block has reduced to $O(\sqrt{n})$, there can be at most $O(\sqrt{n})$ more iterations.

Now, to see the correctness of the algorithm, we once again claim that the following two invariants hold: 1) the set B never contains any circuits, and 2) for each element e of E^{i+1}, there exists an element e' of E^i such that there is a path of length $O(1)$ between e and e' in S_B. For the second invariant, first note that since P^{i+1} is minimal, there must exist a circuit C that contains some elements of C^{i+1}, some elements of E^i, and *all* elements of P^{i+1}. It is clear that there must exist a circuit that contains some elements of C^{i+1} and some elements of E^i because of connectivity; that this circuit contains all elements of P^{i+1} is clear from minimality. This shows that the second invariant holds.

To see why the first invariant holds, we show that any circuit of $C^{i+1} \cup P^{i+1} \cup E^i$ either contains all elements of P^{i+1} or none of them. Since we exclude one element of P^{i+1} and one element of C^{i+1} from B, this shows the first invariant. So suppose, for contradiction, that there exists a circuit C' that contains some elements of P^{i+1} but not all and let f be an element of $C' \cap P^{i+1}$. Clearly, C' cannot contain elements of C^{i+1} and E^i both, otherwise P^{i+1} would not be minimal. Thus suppose without loss of generality that C' contains elements from E^i but not from C^{i+1}. Let $g \in C^{i+1} \cap C$ be any element. Then, from the circuit exchange property (Property 6.3), we know that there exists another circuit C'' such that $g \in C'' \subseteq C \cup C' - \{f\}$. It is clear that C'' must contain an element from E^i since otherwise it would be independent. Since C'' does not contain f, it implies that P^{i+1} is not minimal.

Finally, just like in the case of graphic matroids, it is clear that B spans the matroid once the algorithm terminates. □

6.4 Conclusion

In this chapter we studied the reconfiguration of labelled bases of a matroid and provided an upper bound of $O(n \log n)$ on the worst-case reconfiguration distance for graphic matroids, and a bound of $O(n^{1.5})$ for general matroids. The obvious next question is whether this is tight. The only lower bound we have so far is $\Omega(n)$.

Labelled reconfiguration in some other settings might also be interesting to study. For example, since perfect matchings of a graph also form a matroid, it will be interesting to see what kinds of bounds can be obtained on the diameter of the reconfiguration graph of labelled perfect matchings. Unlabelled perfect matchings were studied by Ito et al. [55] and they showed an upper bound of $O(n)$.

Finally, it will be interesting to find examples where computing the reconfiguration distance is in P for the labelled version but NP-hard for the unlabelled version.

References

[1] Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. In *European Symposium on Algorithms (ESA)*, volume 8125 of *LNCS*, pages 13–24. Springer, 2013.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[3] D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[4] Imre Bárány and Günter Rote. Strictly convex drawings of planar graphs. *Documenta Mathematica*, 11:369–391, 2006.

[5] Jean-Luc Baril and Jean-Marcel Pallo. Efficient lower and upper bounds of the diagonal-flip distance between triangulations. *Information Processing Letters*, 100:131–136, 2006.

[6] Michael A. Bender, Dongdong Ge, Simai He, Haodong Hu, Ron Y. Pinter, Steven Skiena, and Firas Swidan. Improved bounds on sorting with length-weighted reversals. In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 919–928, 2004.

[7] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry, 2nd edition*, pages 47–123. World Scientific, Lecture Notes Series on Computing, 1995.

[8] T. Biedl, G. Kant, and M. Kaufman. On triangulating planar graphs under the four-connectivity constraint. *Algorithmica*, 19(4):427–446, 1997.

[9] Garrett Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, 09 1937.

[10] Marthe Bonamy and Nicolas Bousquet. Recoloring bounded treewidth graphs. In *Proceedings of the 7th Latin-American Algorithms, Graphs, and Optimization Symposium (LAGOS)*, 2013.

[11] Marthe Bonamy, Matthew Johnson, Ioannis Lignos, Viresh Patel, and Danil Paulusma. On the diameter of reconfiguration graphs for vertex colourings. *Electronic Notes in Discrete Mathematics*, 38(0):161 – 166, 2011.

[12] Paul Bonsma. The complexity of rerouting shortest paths. In *Math. Foundations of Computer Science (MFCS)*, volume 7464 of *LNCS*, pages 222–233. Springer, 2012.

[13] Paul Bonsma. Rerouting shortest paths in planar graphs. *CoRR*, abs/1204.5613, 2012.

[14] Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009.

[15] P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry: Theory and Applications*, 42:60–80, 2009.

[16] Prosenjit Bose, Jurek Czyzowicz, Zhicheng Gao, Pat Morin, and David R. Wood. Simultaneous diagonal flips in plane triangulations. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '06, pages 212–221, 2006.

[17] Prosenjit Bose, Dana Jansens, André van Renssen, Maria Saumell, and Sander Verdonschot. Making triangulations 4-connected using flips. *Computational Geometry*, 47:187–197, 2014.

[18] Prosenjit Bose, Anna Lubiw, Vinayak Pathak, and Sander Verdonschot. Flipping edge-labelled triangulations. *CoRR*, abs/1310.1166, 2013.

[19] Prosenjit Bose and Sander Verdonschot. A history of flips in combinatorial triangulations. In *Computational Geometry*, volume 7579 of *Lecture Notes in Computer Science*, pages 29–44. 2012.

[20] Pietro Caputo, Fabio Martinelli, Alistair Sinclair, and Alexandre Stauffer. Random lattice triangulations: structure and algorithms. In *Proc. 45th Annual ACM Symposium on Theory of Computing*, STOC '13, pages 615–624, 2013.

[21] A. Carprara. The reversal median problem. *INFORMS Journal on Computing*, 15(1):93–113, 2003.

[22] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(56):913–919, 2008.

[23] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Mixing 3-colourings in bipartite graphs. *European Journal of Combinatorics*, 30(7):1593–1606, 2009.

[24] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.

[25] Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Information Processing Letters*, 109(16):918–922, 2009.

[26] E.J. Cockayne, O. Favaron, C. Payan, and A.G. Thomason. Contributions to the theory of domination, independence and irredundance in graphs. *Discrete Mathematics*, 33(3):249–258, 1981.

[27] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[28] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM, 2001.

[29] Karel Culik II and Derick Wood. A note on some tree similarity measures. *Information Processing Letters*, 15(1):39–42, 1982.

[30] B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On distances between phylogenetic trees. In *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pages 427–436, 1997.

[31] Jesus A. De Loera, Jorg Rambau, and Francisco Santos. *Triangulations: Structures for Algorithms and Applications*. Springer, 1st edition, 2010.

[32] Patrick Dehornoy. On the rotation distance between binary trees. *Advances in Mathematics*, 223(4):1316–1355, 2010.

[33] Satyan L. Devadoss and Joseph O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

[34] D.Z Djokovic. Distance-preserving subgraphs of hypercubes. *Journal of Combinatorial Theory, Series B*, 14(3):263–267, 1973.

[35] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Spring-Verlag, New York, 1997.

[36] N. Dyn, I. Goren, and S. Rippa. Transforming triangulations in polygonal domains. *Computer Aided Geometric Design*, 10:531–536, 1993.

[37] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.

[38] David Eppstein. Happy endings for flip graphs. *J. Comp. Geom.*, 1:3–28, 2010.

[39] David Eppstein, Jean-Claude Falmagne, and Sergei Ovchinnikov. *Media theory - interdisciplinary applied mathematics*. Springer, 2008.

[40] Guillaume Fertin, Anthony Labarre, Irena Rusu, Eric Tannier, and Stephane Vialette. *Combinatorics of Genome Rearrangement*. MIT Press, 1st edition, 2009.

[41] Gerd Fricke, Sandra Mitchell Hedetniemi, Stephen T. Hedetniemi, and Kevin R. Hutson. γ-Graphs of Graphs. *Discussiones Mathematicae Graph Theory*, 31(3):517–531, 2011.

[42] Jérôme Galtier, Ferran Hurtado, Marc Noy, Stephane Perennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *Int. J. Comput. Geometry Appl.*, 13(2):113–133, 2003.

[43] M.R. Garey and D.S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journ. App. Math.*, 32:826–834, 1977.

[44] William H. Gates and C. Papadimitriou. Bounds for sorting by prefix reversal. *Discrete Mathematics*, 27:47–57, 1979.

[45] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 210–219, 2001.

[46] Oded Goldreich, editor. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*. Springer, 2011.

[47] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The connectivity of boolean satisfiability: computational and structural dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, 2009.

[48] Gregory Gutin and Vadim E. Zverovich. Upper domination and upper irredundance perfect graphs. *Discrete Mathematics*, 190(1-3):95–105, 1998.

[49] Ruth Haas and Karen Seyffarth. The k-Dominating Graph. *Graphs and Combinatorics*, March 2013. Online publication.

[50] Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.

[51] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[52] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete and Computational Geometry*, 22:333–346, 1999.

[53] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Parallel edge flipping. In *Canadian Conference on Computational Geometry*, CCCG, pages 26–27, 1998.

[54] Takehiro Ito and Erik D. Demaine. Approximability of the subset sum reconfiguration problem. In *Proceedings of the 8th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 58–69, 2011.

[55] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.

[56] Takehiro Ito, Marcin Kamiński, and Erik D. Demaine. Reconfiguration of list edge-colorings in a graph. *Discrete Appl. Math.*, 160(15):2199–2207, 2012.

[57] Takehiro Ito, Kazuto Kawamura, Hirotaka Ono, and Xiao Zhou. Reconfiguration of list L(2,1)-labelings in a graph. In *Proceedings of the 23rd International Symposium on Algorithms and Computation*, pages 34–43, 2012.

[58] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths. *Theoretical Computer Science*, 412(39):5205–5210, 2011.

[59] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, June 2012.

[60] Iyad A. Kanj and Ge Xia. Flip distance is in FPT time $O(n + k \cdot c^k)$. *CoRR*, abs/1407.1525, 2014.

[61] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 334–343, New York, NY, USA, 1988. ACM.

[62] J. Kececioglu and R. Ravi. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13:180–210, 1995.

[63] C. Lawson. Software for C^1 surface interpolation. In J. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

[64] Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972.

[65] Ming Li and Louxin 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. Springer-Verlag, 1998.

[66] Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. In *Canadian Conference on Computational Geometry*, CCCG, pages 119–124, 2012.

[67] Joan M. Lucas. The rotation graph of binary trees is Hamiltonian. *J. Algorithms*, 8:503–535, 1988.

[68] Joan M. Lucas. Untangling binary trees via rotations. *The Computer Journal*, 47(2):259–269, 2004.

[69] Fabrizio Luccio, Antonio Mesa Enriquez, and Linda Pagli. Lower bounds on the rotation distance of binary trees. *Information Processing Letters*, 110(21):934–938, 2010.

[70] Ernst W. Mayr and C. Greg Plaxton. On the spanning trees of weighted graphs. *Combinatorica*, 12(4):433–447, 1992.

[71] Nolan Mcmurray, Talmage James Reid, Bing Wei, and Haidong Wu. Largest circuits in matroids. *Advances in Applied Mathematics*, 34(1):213–216, 2005.

[72] L. McShine and P. Tetali. On the mixing time of the triangulation walk and other Catalan structures. *DIMACS-AMS Volume on Randomization Methods in Algorithm Design*, 43:147–160, 1998.

[73] Michael Molloy, Bruce Reed, and William Steiger. On the mixing rate of the triangulation walk. In *DIMACS Series in Discrete Math. Theor. Comput. Sci*, pages 143–179, 2000.

[74] R. Mori, A. Nakamoto, and K. Ota. Diagonal flips in Hamiltonian triangulations on the sphere. *Graphs Combin.*, 19(3):413–418, 2003.

[75] Amer E. Mouawad, Naomi Nishimura, Vinayak Pathak, and Venkatesh Raman. Shortest reconfiguration paths in the solution space of boolean formulas. *CoRR*, abs/1404.3801, 2014.

[76] Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman. Vertex cover reconfiguration and beyond, 2014. arXiv:1402.4926.

[77] Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki. On the parameterized complexity of reconfiguration problems. In *Proceedings of the 8th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 281–294, 2013.

[78] Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):11:1–11:29, May 2008.

[79] E. Osherovich and A.M. Bruckstein. All triangulations are reachable via sequences of edge-flips: an elementary proof. *Computer Aided Geometric Design*, 25:157–161, 2008.

[80] J.G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

[81] Jean Pallo. An efficient upper bound of the rotation distance of binary trees. *Information Processing Letters*, 73:87–92, 2000.

[82] David Peleg. Proximity-preserving labeling schemes. *J. Graph Theory*, 33(3):167–176, March 2000.

[83] Alexander Pilz. Flip distance between triangulations of a planar point set is NP-complete. http://arxiv.org/abs/1206.3179.

[84] Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 14:589–604, 2014.

[85] R. Pinter and S. Skiena. Sorting with length-weighted reversals. In *Proc. 13th International Conference on Genome Informatics*, GIW '02, pages 173–182, 2002.

[86] Lionel Pournin. A combinatorial method to find sharp lower bounds on flip distances. *Proc. 25th International Conference on Formal Power Series and Algebraic Combinatorics*, pages 1–12, 2013.

[87] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, 1978.

[88] Pieter Hendrik Schoute. Analytic treatment of the polytopes regularly derived from the regular polytopes. *Verhandelingen der Koninklijke Akademie van Wetenschappen te Amsterdam*, 11(3):87, 1911.

[89] Konrad W. Schwerdtfeger. A computational trichotomy for connectivity of boolean satisfiability. *CoRR*, abs/1312.4524, 2013.

[90] Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *The Electronic Journal of Combinatorics*, 18(1):P70, 2011.

[91] Micha Sharir, Adam Sheffer, and Emo Welzl. On degrees in random triangulations of point sets. *Journal of Combinatorial Theory, Series A*, 118(7):1979 – 1999, 2011.

[92] A. Shrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[93] D. D. Sleator, R. Tarjan, and W. Thurston. Rotation distance. In T.M. Cover and B. Gopinath, editors, *Open problems in communication and computation*. Springer-Verlag, 1987.

[94] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. Assoc. Comput. Mach.*, 32:652–686, 1985.

[95] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Amer. Math. Soc*, 1:647–681, 1988.

[96] D.D. Sleator, R.E. Tarjan, and W.P. Thurston. Short encodings of evolving structures. *SIAM J. Discrete Math.*, 5:428–450, 1992.

[97] S. Trakultraipruk. *Connectivity Properties of Some Transformation Graphs*. PhD thesis, London School of Economics, 2013.

[98] Jan van den Heuvel. The complexity of change. *CoRR*, abs/1312.2816, 2013.

[99] K. Wagner. Bemerkung zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereiningung*, 46:26–32, 1936.

[100] T. Watanabe, T. Ae, and A. Nakamura. On the node cover problem of planar graphs. In *Proceedings of the International Symposium on Circuits and Systems*, pages 78–81, 1979.

[101] Oren Weimann and David Peleg. A note on exact distance labeling. *Inf. Process. Lett.*, 111(14):671–673, July 2011.