FLIPS AND SPANNERS

ALEXANDER JOZEF HUBERTUS VERDONSCHOT

A thesis submitted to the Faculty of Graduate and Post Doctoral Affairs
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

Carleton University
Ottawa, Ontario, Canada

ABSTRACT

In this thesis, we study two different graph problems.

The first problem revolves around *geometric spanners*. Here, we have a set of points in the plane and we want to connect them with straight line segments, such that there is a path between each pair of points and these paths do not require large detours. If we achieve this, the resulting graph is called a spanner. We focus our attention on two graphs (the Θ-graph and Yao-graph) that are constructed by connecting each point with its nearest neighbour in a number of cones. Although this construction is very straight-forward, it has proven challenging to fully determine the properties of the resulting graphs. We show that if the construction uses 5 cones, the resulting graphs are still spanners. This was the only number of cones for which this question remained unanswered. We also present a routing strategy (a way to decide where to go next, based only on our current location, its direct neighbourhood, and our destination) on the half-Θ_6-graph, a variant of the graph with 6 cones. We show that our routing strategy avoids large detours: it finds a path whose length is at most a constant factor from the straight-line distance between the endpoints. Moreover, we show that this routing strategy is optimal.

In the second part, we turn our attention to flips in triangulations. A flip is a simple operation that transforms one triangulation into another. It turns out that with enough flips, we can transform any triangulation into any other. But how many flips is enough? We present an improved upper bound of $5.2n - 33.6$ on the maximum flip distance between any pair of triangulations with n vertices. Along the way, we prove matching lower bounds on each step in the current algorithm, including a tight bound of $\lfloor (3n-9)/5 \rfloor$ flips needed to make a triangulation 4-connected. In addition, we prove tight $\Theta(n \log n)$ bounds on the number of flips required in several settings where the edges have unique labels.

ACKNOWLEDGMENTS

I would not have been able to write this thesis without the help and support of many people.

First of all, I would like to thank my supervisors – Prosenjit Bose, Pat Morin, and Vida Dujmović. They are all I could have wished for in my supervisors and more. Combining a wealth of knowledge with a burning curiosity and a penchant for finding fascinating, yet approachable, open problems, they made these past five years into a journey of exploration and excitement.

I also want to thank the other members and students of the Computational Geometry lab for making it such a nice place to work (and occasionally not work). I am especially grateful to fellow PhD students André, Carsten, Dana, and Luis, for being great friends and collaborators. In fact, I am very grateful to all the researchers and students who I got to work with during my PhD studies. Working together was always a pleasure, and they taught me more than classes ever could.

Finally, I would like to thank my family and friends for their support during these long, and at times stressful, years. I am especially grateful to my mother for encouraging me to take the leap of faith that is an international PhD, and to my partner, Gehana, for her unfailing love and support.

Thank you!

CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS v

CONTENTS vii

1 SUMMARY OF THE THESIS 1
- 1.1 Geometric spanners 1
- 1.2 Flips in triangulations 2

I GEOMETRIC SPANNERS 5

2 AN INTRODUCTION TO YAO- AND Θ-GRAPHS 7
- 2.1 Geometric spanners 7
- 2.2 Preliminaries 9
- 2.3 Yao-graphs 10
- 2.4 Θ-graphs 14
- 2.5 History 16

3 THE Θ_5-GRAPH IS A SPANNER 21
- 3.1 Introduction 21
- 3.2 Connectivity 23
- 3.3 Spanning ratio 25
- 3.4 Lower bound 35
- 3.5 Conclusions 38

4 COMPETITIVE ROUTING IN THE HALF-Θ_6-GRAPH 41
- 4.1 Introduction 41
- 4.2 Preliminaries 43
- 4.3 Spanning ratio of the half-Θ_6-graph 45
- 4.4 Remarks on the spanning ratio 48
- 4.5 Routing in the half-Θ_6-graph 49
  - 4.5.1 Positive routing 51
  - 4.5.2 Negative routing 57
- 4.6 A stateful algorithm 60
- 4.7 Bounding the maximum degree 61
  - 4.7.1 Routing in G_{12} 63
  - 4.7.2 Routing in G_9 67
- 4.8 Conclusions 68

II FLIPS IN TRIANGULATIONS 73

5 A HISTORY OF FLIPS IN COMBINATORIAL TRIANGULATIONS 75
- 5.1 Introduction 75
- 5.2 Wagner's bound 77
- 5.3 Komuro's bound 79
- 5.4 Mori et al.'s bound 81
- 5.5 Lower bounds 84

6 MAKING TRIANGULATIONS 4-CONNECTED USING FLIPS 87
6.1 Introduction 87
6.2 Upper bound 88
6.3 Lower bound 102
6.4 Conclusions and open problems 104
6.5 Lemmas and proofs 105
7 EDGE-LABELLED FLIPS 109
7.1 Introduction 109
7.2 Convex polygons 111
7.2.1 Upper bound 111
7.2.2 Lower bound 114
7.2.3 Simultaneous flips 117
7.3 Combinatorial triangulations 120
7.3.1 Upper bound 120
7.3.2 Lower bound 123
7.3.3 Simultaneous flips 124
7.4 Pseudo-triangulations 126
7.4.1 Pointed pseudo-triangulations 128
7.4.2 General pseudo-triangulations 134
7.5 Conclusions and open problems 139

1

SUMMARY OF THE THESIS

This thesis is comprised of two main parts. The first part, found in Chapters 2 through 4, deals with geometric spanners. Chapters 5 through 7 contain the second part, which focuses on flips in triangulations. A brief introduction and summary of each part is given below. The first chapter of each part provides a more detailed introduction.

The common theme in the two parts is that both deal with *graphs*. A graph consists of a set of *vertices*, some of which are connected by *edges*. In this thesis, all graphs will be simple, which means that there is at most one edge connecting each pair of vertices, and edges cannot connect a vertex to itself.

1.1 GEOMETRIC SPANNERS

Spanners can be informally described as graphs in which one never needs to make a large detour. That is, the shortest path between two vertices is proportional to their actual distance. Road networks are a good example; nearby cities are typically connected by a direct road, so that the total distance travelled is not much more than the distance 'as the crow flies'. Spanners have been studied in many different contexts, but we will focus on *geometric spanners*, where the vertices are points in the plane, and the length of an edge is the Euclidean distance between its endpoints. The *spanning ratio* is the maximum ratio between the shortest path in the graph and the straight-line distance between any pair of vertices.

Chapter 2 gives an in-depth introduction to geometric spanners in general, and simple cone-based spanners in particular. The Θ-graph is one such cone-based spanner. To construct it, we partition the plane around each vertex into a number of equiangular cones and add an edge to the 'closest' vertex in each cone, where the closest vertex is defined as the vertex whose projection on the bisector of the cone is closest. It has been shown that for any desired spanning ratio t, there is a number of cones k such that the Θ-graph with k cones (typically written as Θ_k) is guaranteed to have spanning ratio t.

However, it was not known exactly for which values of k the spanning ratio of Θ_k is bounded by a constant. It was known that Θ_3 and below are not constant spanners, while Θ_6 and up are. Recently, Θ_4 was shown to be a constant spanner as well, leaving the question unanswered only for Θ_5. In Chapter 3, we prove that Θ_5 is, indeed, a constant spanner. With the earlier results, this implies that Θ_k is a spanner for all $k \geqslant 4$. This result was first published in the proceed-

ings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2013) [8] and later appeared in Computational Geometry: Theory and Applications [9].

Of course, knowing that there exists a short path to where you want to go is not the end of the story: you also have to know how to find it. This is called *routing*, or *competitive routing* if the spanning ratio of the resulting path is bounded by a constant. If you know the entire graph, routing is nothing more than computing a path, but most settings consider the more restricted scenario where you know your destination, but you can only see your current location and its neighbours. This is referred to as *local* routing. In Chapter 4, we present a local, competitive routing strategy for the half-Θ_6-graph, which is closely related to Θ_6. Our strategy achieves a routing ratio of $5/\sqrt{3} = 2.886\ldots$, which seems slightly disappointing compared to the spanning ratio of 2. This makes it all the more surprising that we managed to show that our algorithm is, in fact, optimal: no other routing strategy can achieve a better routing ratio, under the same restrictions. This is the first such separation between the spanning and routing ratios on a graph. These results were first published in the proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA 2012) [4], and the proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012) [5], and have recently been accepted for publication in the SIAM Journal on Computing [3].

1.2 FLIPS IN TRIANGULATIONS

A triangulation is a planar graph where each face is a triangle (a cycle of three edges). A *flip* is a simple, local operation that transforms one triangulation into another. Specifically, we can flip an edge e by removing it, leaving an empty quadrilateral, and inserting the other diagonal of this quadrilateral. Flips were introduced by Wagner in 1936 in an attempt to make progress on the famous four-colour-theorem, and have been actively studied ever since. Applications of flips range from enumeration [1] and optimization of triangulations [2] to correcting errors in 3-dimensional terrains generated from height measurements [13]. Similar local operations that transform one graph into another in the same class have been used to build robust peer-to-peer network toplogies [12] and to find heuristic solutions to the Traveling Salesman Problem [14].

Wagner showed that, using flips, it is possible to transform any triangulation into any other. One question that has received a great deal of attention since then is: how many flips does this take, in the worst case? Chapter 5 presents a detailed history of various attempts to answer this question. This survey was published as an invited chap-

ter in the proceedings of the XIV Spanish Meeting on Computational Geometry (EGC 2011) [10].

Chapter 6 details our own contribution to answering this question. In particular, we prove a tight bound of $\lfloor (3n-9)/5 \rfloor$ on the number of flips required to make an n-vertex triangulation 4-connected. And since the best known algorithm to transform any triangulation into any other first makes the triangulations in question 4-connected, this improves the upper bound on the total number of flips required from $6n-30$ to $5.2n-33.6$. These results were first published in the proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2011) [6], and subsequently appeared in a special issue of Computational Geometry: Theory and Applications [7].

All of the research on flips thus far has assumed that edges are indistinguishable. But what happens when we give each edge a unique label, that is carried over to the new edge when an edge is flipped? This is the question studied in Chapter 7. We prove the first upper and lower bounds on the number of flips required in this setting. In particular, we show that $\Theta(n \log n)$ flips are required for edge-labelled triangulations of a convex polygon, edge-labelled combinatorial triangulations, and edge-labelled pseudo-triangulations. The results on pseudo-triangulations have been accepted to the 27th Canadian Conference on Computational Geometry (CCCG 2015) [11].

BIBLIOGRAPHY

[1] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1–3):21–46, 1996.

[2] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. 1992.

[3] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SIAM Journal on Computing*. Forthcoming.

[4] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive routing in the half-θ_6-graph. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1319–1328, 2012.

[5] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive routing on a bounded-degree plane spanner. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012)*, pages 299–304, 2012.

[6] Prosenjit Bose, Dana Jansens, André van Renssen, Maria Saumell, and Sander Verdonschot. Making triangulations 4-

connected using flips. In *Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2011)*, pages 241–247, 2011.

[7] Prosenjit Bose, Dana Jansens, André van Renssen, Maria Saumell, and Sander Verdonschot. Making triangulations 4-connected using flips. *Computational Geometry: Theory and Applications*, 47(2A):187–197, 2014. Special issue for CCCG 2011.

[8] Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The θ_5-graph is a spanner. In *Proceedings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2013)*, pages 100–114, 2013.

[9] Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The θ_5-graph is a spanner. *Computational Geometry: Theory and Applications*, 48(2):108–119, 2015.

[10] Prosenjit Bose and Sander Verdonschot. A history of flips in combinatorial triangulations. In *Proceedings of the XIV Spanish Meeting on Computational Geometry (EGC 2011)*, volume 7579 of *Lecture Notes in Computer Science*, pages 29–44. 2012.

[11] Prosenjit Bose and Sander Verdonschot. Flips in edge-labelled pseudo-triangulations. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG 2015)*, pages 63–69, 2015.

[12] Colin Cooper, Martin Dyer, and Andrew J Handley. The flip Markov chain and a randomising P2P protocol. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, pages 141–150, 2009.

[13] Thierry de Kok, Marc van Kreveld, and Maarten Löffler. Generating realistic terrains with higher-order Delaunay triangulations. *Computational Geometry: Theory and Applications*, 36(1):52–65, 2007.

[14] Shen Lin. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269, 1965.

Part I

GEOMETRIC SPANNERS

2

AN INTRODUCTION TO YAO- AND Θ-GRAPHS

In the past thirty years, geometric spanners have become an important field of study in computational geometry. This chapter serves as an introduction to the field, with a focus on two closely related families of geometric spanners: Yao-graphs and Θ-graphs.

Most of the material in this chapter was already known, but the improvement for Yao-graphs with an odd number of cones (Theorem 2.4) is new, although it was discovered independently by Keng and Xia [16]. The proof of the spanning ratio of Θ-graphs (Theorem 2.5) is also new.

2.1 GEOMETRIC SPANNERS

Many practical geometric problems can be modelled as connecting a set of points in the plane. Examples include building roads to connect cities, or creating a communications network among wireless sensors. For these problems, we typically want to achieve good connectivity between the points, while using only a small number of connections. In the case of road networks in particular, we would like to avoid large detours: if cities A and B are fairly close, people should not have to drive to a distant city C to travel from A to B. This is what geometric spanners try to achieve: the shortest path between any two points in the network should be proportional to the distance between the points.

More formally, given a set P of points in the plane, a geometric t-*spanner* of P is a graph G with vertex set P, such that for each pair of points, the length of the shortest path between the corresponding vertices in G is at most t times the Euclidean distance between them. The *spanning ratio* of G is the smallest t for which it is a t-spanner (in other texts, the spanning ratio is also called the *dilation* or *stretch factor*).

This definition is often applied to families of graphs. A family of graphs is called a t-spanner if every graph in the family is a t-spanner, and the spanning ratio of the family is the smallest t such that every graph in the family is a t-spanner. A family of graphs is called a *spanner* if there exists some finite t for which it is a t-spanner.

As a first example, consider the complete graph on P. As it contains an edge between every pair of points in P, this family of graphs is a 1-spanner. And if P does not contain three co-linear points, it is also the only 1-spanner, since the removal of any edge would increase the distance between its endpoints. Of course, the large drawback of

the complete graph is that the number of edges is quadratic in the number of vertices. We would like to find sparser graphs (typically with a linear number of edges) that still have a small spanning ratio.

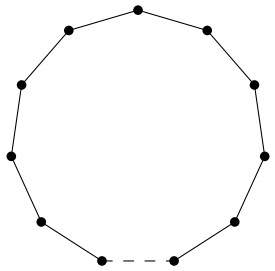


Figure 2.1: The minimum spanning tree of the vertices of a regular n-gon has spanning ratio $n-1$.

The minimum spanning tree is at the other end of the spectrum. In order to be a spanner, a graph clearly needs to be connected (otherwise the spanning ratio is infinite). The minimum spanning tree is the connected graph on P with lowest total edge length. Unfortunately, this family of graphs is not a spanner. To see this, imagine n points spread equally on a circle. The minimum spanning tree of these points will include every edge between two consecutive points, except for one (see Figure 2.1). The endpoints of this non-edge are at distance x, but the only path between them in the graph follows the entire path around the circle, which has length $(n-1)\cdot x$. Thus for every constant t, we can construct a point set with $\lceil t\rceil + 2$ vertices whose minimum spanning tree has spanning ratio $\lceil t\rceil + 1 > t$, meaning that there does not exist a constant t such that every minimum spanning tree is a t-spanner. In fact, for this particular point set, *every* tree has spanning ratio $\Omega(n)$.

THEOREM 2.1 (Eppstein [11], Lemma 15). *Any spanning tree T on $n \geqslant 6$ points spread evenly on a circle has spanning ratio at least $\frac{n}{2\pi}$.*

Proof. Every tree has a vertex separator: a vertex v such that removing v splits T into connected components with at most $n/2$ vertices each. Consider the $n/2+1$ vertices that lie opposite v on the circle. Since each connected component has size at most $n/2$, there must be a pair of vertices x and y from different components that are adjacent on the circle. Since they are in different connected components, the shortest path in T from x to y passes through v. Thus, the spanning ratio of T is at least:

$$\begin{aligned}
\frac{|xv|+|yv|}{|xy|} &\geqslant \frac{2\cdot 2\sin\left(\left(\frac{n}{4}-1\right)\frac{\pi}{n}\right)}{2\sin\left(\frac{\pi}{n}\right)} \\
&\geqslant \frac{2\cdot\frac{1}{2}}{2\frac{\pi}{n}} && \text{(for } n \geqslant 6\text{)} \\
&= \frac{n}{2\pi}
\end{aligned}$$

□

Note that spanners have also been studied for general weighted graphs (where the shortest path in the spanner is compared to the shortest path in the original graph), or for point sets in higher dimensions. In this thesis, we deal almost exclusively with spanners of two-dimensional point sets; any exceptions will be mentioned explicitly. For a broader overview of geometric spanners, we recommend the book by Narasimhan and Smid [18].

2.2 PRELIMINARIES

The proofs in this part of the thesis make extensive use of trigonometry. This section contains a short review of the basic properties used throughout the next chapters. Here, and in the rest of this thesis, we use $|ab|$ to denote the Euclidean distance between two points (or vertices) a and b.

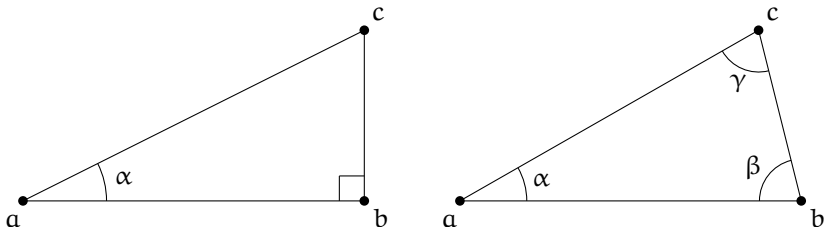


Figure 2.2: (a) A right triangle. (b) A general triangle.

TRIGONOMETRIC FUNCTIONS. The basic trigonometric functions are the *sine*, *cosine*, and *tangent*. They are defined as the ratio of the sides in a right triangle. Consider a triangle abc such that $\angle abc$ is a right angle (see Figure 2.2a). If we let $\alpha = \angle cab$, then

$$\sin\alpha = \frac{|bc|}{|ac|}, \qquad \cos\alpha = \frac{|ab|}{|ac|}, \text{ and} \qquad \tan\alpha = \frac{|bc|}{|ab|}.$$

TRIGONOMETRIC IDENTITIES. There are several more complex equalities that can be derived from these basic functions. The two we use most often are called the *law of sines* and the *law of cosines*. They have the advantage that they apply to all triangles, not only right triangles. In a triangle abc with $\alpha = \angle cab$, $\beta = \angle abc$, and $\gamma = \angle bca$ (see Figure 2.2b), these identities are expressed as follows.

$$\frac{|ab|}{\sin\gamma} = \frac{|ac|}{\sin\beta} = \frac{|bc|}{\sin\alpha} \qquad \text{(law of sines)}$$

$$\begin{aligned} |ab|^2 &= |ac|^2 + |bc|^2 - 2|ac||bc|\cos\gamma \\ |ac|^2 &= |ab|^2 + |bc|^2 - 2|ab||bc|\cos\beta \\ |bc|^2 &= |ab|^2 + |ac|^2 - 2|ab||ac|\cos\alpha \end{aligned} \qquad \text{(law of cosines)}$$

TRIANGLE INEQUALITY. The last equation we cover here is not an equality, but rather an inequality known as the *triangle inequality*. It states that one side of a triangle is never longer than the other two sides combined. Using the notation of the triangle depicted in Figure 2.2b, it can be expressed as follows.

$$\begin{aligned} |ab| &\leqslant |ac| + |bc| \\ |ac| &\leqslant |ab| + |bc| \\ |bc| &\leqslant |ab| + |ac| \end{aligned} \qquad \text{(triangle inequality)}$$

2.3 YAO-GRAPHS

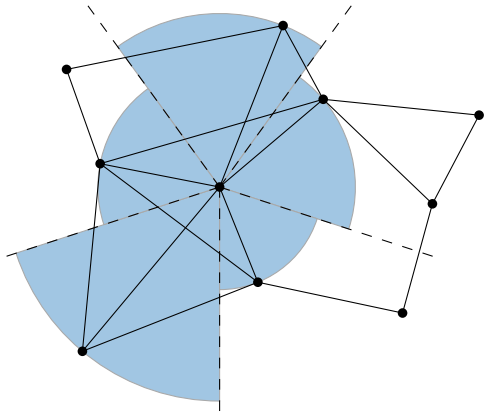


Figure 2.3: An example Y_5-graph. Each vertex adds an edge to the closest vertex in each of five equiangular cones.

One simple way to build a geometric spanner is to take each vertex, partition the plane around it into a fixed number of cones with equal angles, and add an edge between the vertex and the closest vertex in each cone (see Figure 2.3). The resulting graph is called a Yao-graph, and is typically denoted by Y_k, where k is the number of cones around each vertex. This construction guarantees that a Yao-graph with k cones has at most kn edges, where n is the number of vertices. Furthermore, if the cones are narrow enough, we can find a path between any two vertices by starting at one and walking to the closest vertex in the cone that contains the other, repeating this until we end up at our destination. Intuitively, this results in a short path because we are always walking approximately in the right direction, and, since our neighbour is the closest vertex in that direction, never too far.

A Yao-graph whose cones are narrower will typically give a better approximation of the shortest path. If we want to prove this formally,

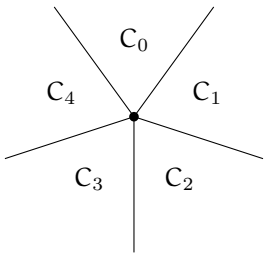


Figure 2.4: The cones used to construct the Y_5-graph. The orientation is the same for all vertices.

we need to iron out a few details in the definition of a Yao-graph. First, we assume that points are in general position; in particular, we assume that for every vertex a, there are no two points at the exact same distance from a. This means that each vertex has a unique closest vertex in each cone. (This is not strictly necessary, but it makes our proofs simpler. If we don't assume general position, the same properties hold by breaking ties arbitrarily.) Second, we label the cones C_0 through C_{k-1} in clockwise order, and we orient them such that the bisector of C_0 aligns with the positive y-axis (see Figure 2.4). This orientation is the same for each vertex. If the apex is not clear from the context, we use C_i^a to denote cone C_i with apex a. The boundary between two cones belongs to the counter-clockwise one (so the boundary between C_0 and C_1 is part of C_0). The crux of the proof lies in the following small geometric lemma that captures our earlier intuition that taking a small step in approximately the right direction makes meaningful progress towards our destination.

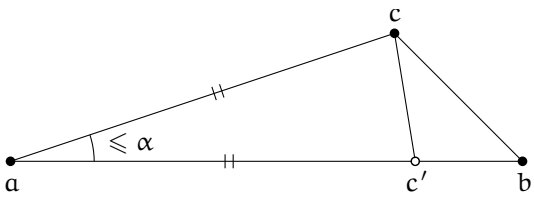


Figure 2.5: We are at a vertex a and want to go to b, but c is the closest vertex in the cone that contains b.

LEMMA 2.2. *Given three points* a, b, *and* c, *such that* $|ac| \leqslant |ab|$ *and* $\angle bac \leqslant \alpha < \pi/3$, *then*

$$|bc| \leqslant |ab| - (1 - 2\sin(\alpha/2)) \cdot |ac|.$$

Proof. Let c' be the point on ab such that $|ac| = |ac'|$ (see Figure 2.5). Since acc' forms an isosceles triangle, we can express $|cc'|$ in terms of $|ac|$:

$$|cc'| = 2\sin(\angle bac/2) \cdot |ac| \leqslant 2\sin(\alpha/2) \cdot |ac|.$$

The inequality holds since $\sin x$ is increasing in this range. Now we just need the triangle inequality:

$$\begin{aligned} |bc| &\leqslant |bc'| + |c'c| \\ &\leqslant |ab| - |ac'| + 2\sin(\alpha/2) \cdot |ac| \\ &= |ab| - (1 - 2\sin(\alpha/2)) \cdot |ac|. \end{aligned}$$ □

Imagine that we are at a vertex a and we want to go to b, but c is the closest vertex in the cone of a that contains b. Then this lemma essentially tells us that if the angle between our destination and the edge we follow (α) is small, the amount of progress we make ($|ab| - |bc|$) is directly proportional to the distance we travel ($|ac|$):

$$\begin{aligned} |ab| - |bc| &\geqslant |ab| - (|ab| - (1 - 2\sin(\alpha/2)) \cdot |ac|) \\ &\geqslant (1 - 2\sin(\alpha/2)) \cdot |ac|. \end{aligned}$$

Now that we have this lemma, we can use an inductive argument to show that Yao-graphs are spanners.

THEOREM 2.3. *For any integer $k \geqslant 7$, the graph Y_k has spanning ratio at most $1/(1 - 2\sin(\theta/2))$, where $\theta = 2\pi/k$.*

Proof. Let a and b be two arbitrary vertices in our point set. We show that the obvious way to get from a to b – keep following the edge in the cone that contains b – not only works, it even gives us a short path. To start off, consider all pairs of vertices (u, v) and sort them by their distance $|uv|$. Our proof proceeds by induction on the index of (a, b) in this sorted order.

In the base case, (a, b) is the closest pair. This means that b must be the closest vertex in the cone of a that contains b, so the edge (a, b) is in the graph. Thus, the shortest path between a and b has length exactly $|ab|$, giving a spanning ratio of 1. Since $1/(1 - 2\sin(\theta/2)) > 1$ for $0 < \theta < 2\pi$, this proves the base case.

For the inductive step, assume that for any pair of vertices (u, v) such that $|uv| < |ab|$, there exists a path from u to v with length at most $1/(1 - 2\sin(\theta/2)) \cdot |uv|$. Now consider the cone of a that contains b. If b is the closest vertex, the edge (a, b) is in the graph and we can use the same argument as in the base case. Otherwise, let c be the closest vertex to a. Note that, because $\angle bac \leqslant \theta \leqslant 2\pi/7 < \pi/3$, we know that $\angle bac$ is not the largest angle in triangle abc. Since the largest angle lies opposite the longest edge, bc is not the longest edge, so $|bc| < |ab|$. Using our inductive hypothesis, this means that there is a path between b and c with length at most $1/(1 - 2\sin(\theta/2)) \cdot |bc|$. So to go from a to b, we can first take the direct edge to c and then

follow the path to b. Since a, b, and c satisfy all the conditions for Lemma 2.2, we can use it to bound the length of the resulting path:

$$\begin{aligned} & |ac| + \frac{1}{1-2\sin(\theta/2)} \cdot |bc| \\ \leqslant\ & |ac| + \frac{1}{1-2\sin(\theta/2)} \cdot (|ab| - (1-2\sin(\theta/2)) \cdot |ac|) \\ =\ & |ac| + \frac{1}{1-2\sin(\theta/2)} \cdot |ab| - |ac| \\ =\ & \frac{1}{1-2\sin(\theta/2)} \cdot |ab|. \end{aligned}$$

Which is exactly what we needed to show. □

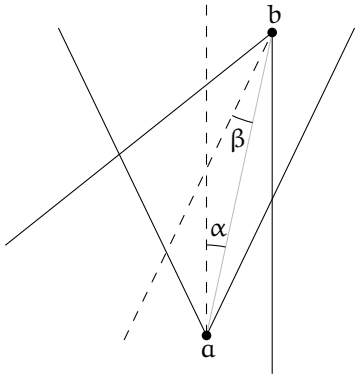


Figure 2.6: If the number of cones is odd, opposite cones are not symmetric and either α or β is small.

Interestingly, we can do a little better when the number of cones is odd. This is caused by the asymmetry in the cones. To see why, consider the situation where we have two vertices, a and b, and the number of cones is odd. Let C^a be the cone of a that contains b and let C^b be the analogous cone for b. Let α and β be the angles between ab and the bisectors of C^a and C^b, respectively (see Figure 2.6). Since the bisector of C^a is parallel to one of the sides of C^b, the transversal ab creates equal angles at a and b, showing that $\alpha + \beta = \theta/2$. Therefore, the smaller of α and β can be at most $\theta/4$. If we assume that α is the smaller of the two, and let c be the closest vertex in C^a, then $\angle bac \leqslant \alpha + \theta/2 \leqslant 3\theta/4$. Plugging this into the proof of Theorem 2.3 gives the following result.

THEOREM 2.4. *For any odd integer $k \geqslant 5$, the graph Y_k has spanning ratio at most $1/(1-2\sin(3/8 \cdot \theta))$, where $\theta = 2\pi/k$.*

Note that this theorem extends to Y_5, as $3\theta/4 = 3/4 \cdot 2\pi/5 = 3\pi/10 < \pi/3$, whereas Theorem 2.3 does not.

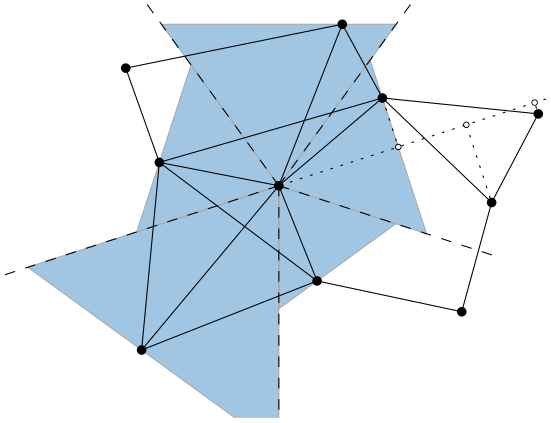


Figure 2.7: An example Θ_5-graph.

2.4 Θ-GRAPHS

If we modify the definition of Yao-graphs slightly we obtain another type of geometric spanner, called a Θ-graph. The only difference lies in the way the closest vertex is determined: for each vertex u, the closest vertex in a cone C is the vertex v whose orthogonal projection on the bisector of C is closest to u (see Figure 2.7). We again assume general position to simplify our proofs; in particular, we assume that no two vertices lie on a line parallel or perpendicular to a cone boundary, guaranteeing that each vertex connects to at most one vertex in each cone, and thus that the graph has at most kn edges. Another way to look at the construction is that we sweep C with a line perpendicular to the bisector, and add an edge to the first vertex we hit. Note that this creates an empty triangle (the shaded regions in Figure 2.7). Given two vertices a and b, we can define their *canonical triangle* $\triangle_{ab}$ as the triangle formed by the boundaries of the cone C of a that contains b and the line through b perpendicular to the bisector of C. Note that the canonical triangle $\triangle_{ba}$ also exists: it is the same size as $\triangle_{ab}$, but is has apex b and is oriented towards a instead. This gives a third way to describe the construction of the Θ-graph, by adding an edge between two vertices if one of their canonical triangles is empty. These canonical triangles play an important role in Chapters 3 and 4. As one might expect from the similarity in construction, Θ-graphs share many of the properties that make Yao-graphs interesting. Their key advantage, however, is that they can be constructed by an easy sweep-line algorithm, whereas all known algorithms for constructing Yao-graphs are more complex. Here we prove that Θ-graphs are spanners as well.

THEOREM 2.5. *For any integer $k \geqslant 7$, the graph Θ_k has spanning ratio at most $t = 1/(1-2\sin(\theta/2))$, where $\theta = 2\pi/k$.*

Proof. This proof is similar to the proof of Theorem 2.3; we have two vertices a and b and we show that there is a path between them of

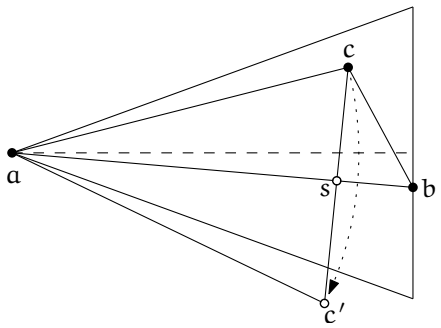


Figure 2.8: The canonical triangle $\triangle_{ab}$, with the closest vertex to a rotated over an angle θ.

length at most $t|ab|$. The proof is again by induction on the relative position of (a, b) among all pairs of points when ordered by distance. For convenience, we translate and rotate the point set such that a is in the origin, and the bisector of the cone of a that contains b coincides with the positive x-axis. We start by considering the inductive step, and prove the base case at the end.

We assume that there is a path from u to v of length at most $t|uv|$ for all pairs (u, v) with $|uv| < |ab|$. If the edge (a, b) is in the graph, we have a path from a to b of length $|ab| \leqslant t|ab|$ and are done, since $t > 1$, so assume that this is not the case. Then there is another vertex c, whose projection on the bisector is closest to a in the cone containing b. Without loss of generality, we assume that c lies above ab (if it does not, we can mirror everything in the x-axis).

Now imagine rotating c clockwise around a by an angle of θ, and let c' be the resulting position (see Figure 2.8). Note that c' lies below ab, as the angle between ac and ab is at most θ. Furthermore, rotating c by the angle between ac and the positive x-axis would move it to a point with the same x-coordinate, but since we rotated it further, c' lies to the left of c and therefore to the left of b. Since two line segments intersect if and only if for both segments, the endpoints lie on opposite sides of the other segment, ab and cc' intersect, and we call their intersection point s. Now we can use the triangle inequality to obtain the following inequalities:

$$\begin{aligned} |ac| &= |ac'| \leqslant |as| + |sc'|, \text{ and} \\ |bc| &\leqslant |bs| + |sc|. \end{aligned}$$

Thus, we get that

$$\begin{aligned} |ac| + |bc| &\leqslant |as| + |sc'| + |bs| + |sc| \\ &= |ab| + |cc'| \\ &= |ab| + 2\sin\frac{\theta}{2}|ac| \\ |bc| &\leqslant |ab| + 2\sin\frac{\theta}{2}|ac| - |ac| \\ &= |ab| - \frac{1}{t}|ac| \end{aligned}$$

For $k \geqslant 7$, this implies that $|bc| < |ab|$, which means that we can apply our inductive hypothesis to bc. By first following ac, this gives us a path from a to b of length at most:

$$\begin{aligned} |ac| + t|bc| &\leqslant |ac| + t\left(|ab| - \frac{1}{t}|ac|\right) \\ &= |ac| + t|ab| - |ac| \\ &= t|ab| \end{aligned}$$

This settles the inductive step. For the base case, if the edge (a, b) is in the graph we are again done. The only way this edge could be absent is if another vertex c had a projection on the bisector closer to a. But we just derived that in that case $|bc| < |ab|$, which contradicts the fact that $|ab|$ is minimal. Therefore this cannot happen, and b must be the closest vertex to a, proving the theorem. □

2.5 HISTORY

Research on geometric spanners was sparked by a paper by Paul Chew in 1986 [7], titled "There is a planar graph almost as good as the complete graph". In that paper and the subsequent journal version [8], he showed that certain Delaunay triangulations are geometric spanners with few edges. In particular, he showed this for Delaunay triangulations whose empty regions are the square and the equilateral triangle. The traditional Delaunay triangulation, which uses a circle, was quickly shown to be a spanner as well [10]. Although its true spanning ratio remains a mystery, the upper bound has been improved multiple times; from the initial bound of 5.08 in 1987, to 2.42 in 1989 [14, 15], and recently to just below 2 [21].

Yao-graphs were introduced independently by Flinchbaugh and Jones [12] and Yao [22] around 1981, before the concept of spanners was even introduced by Chew. Yao showed that Y_8 is a supergraph of the minimum spanning tree and that this still holds in higher dimensions. This gave an efficient algorithm to compute the minimum spanning tree in higher dimensions.

To the best of our knowledge, the first proof that Yao-graphs are geometric spanners was published in 1993, by Althöfer et al. [1]. In

particular, they showed that for every spanning ratio $t > 1$, there exists a number of cones k such that Y_k is a t-spanner. It appears that some form of this result was known earlier, as Clarkson [9] already remarked in 1987 that Y_{12} is a $1+\sqrt{3}$-spanner, albeit without providing a proof or reference. In 2004, Bose et al. [6] provided a more specific bound on the spanning ratio, by showing that for $k > 8$, Y_k is a geometric spanner with spanning ratio at most $1/(\cos\theta - \sin\theta)$, where $\theta = 2\pi/k$. This bound was later improved to $1/(1-2\sin(\theta/2))$, for $k > 6$ [3]. We presented a simplified version of this proof for Theorem 2.3. The improvement for odd k given in Theorem 2.4 is a recent development by Barba et al. [2].

The Θ-graph was introduced independently by Clarkson [9] and Keil [13, 15], as an alternative to Yao-graphs that was easier to compute. Both papers prove a spanning ratio of $1/(\cos\theta - \sin\theta)$, which was later improved to $1/(1-2\sin(\theta/2))$ by Ruppert and Seidel [19]. The proof of Theorem 2.5 is significantly simpler than their proof, and is based on another proof by Lukovski [17, p. 11].

This bound of $1/(1-2\sin(\theta/2))$ was the best known upper bound on the spanning ratio for over twenty years. Only very recently have researchers been able to prove that the true bound is lower. In 2012, Bose et al. [4] showed that Θ-graphs with $4m+2$ cones ($m \geqslant 1$) have a spanning ratio of $1+2\sin(\theta/2)$. Surprisingly, they were also able to give a matching lower bound, making this the first family of Θ-graphs for which a tight bound on the spanning ratio is known. Later, they used similar techniques to improve the upper bound on the spanning ratio of all other Θ-graphs [5], although these do not yet match the best known lower bounds. A good overview of these results, and more, can be found in the thesis of André van Renssen [20].

BIBLIOGRAPHY

[1] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

[2] Luis Barba, Prosenjit Bose, Mirela Damian, Rolf Fagerberg, Wah Loon Keng, Joseph O'Rourke, André van Renssen, Perouz Taslakian, Sander Verdonschot, and Ge Xia. New and improved spanning ratios for Yao graphs. *Journal of Computational Geometry*, 6(2):19–53, 2015. Special issue for SoCG 2014.

[3] Prosenjit Bose, Mirela Damian, Karim Douïeb, Joseph O'Rourke, Ben Seamone, Michiel Smid, and Stefanie Wuhrer. $\pi/2$-angle Yao graphs are spanners. *ArXiv e-prints*, 2010. arXiv:1001.2913 [cs.CG].

[4] Prosenjit Bose, Jean-Lou De Carufel, Pat Morin, André van Renssen, and Sander Verdonschot. Optimal bounds on Theta-

graphs: More is not always better. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012)*, pages 305–310, 2012.

[5] Prosenjit Bose, Jean-Lou De Carufel, Pat Morin, André van Renssen, and Sander Verdonschot. Towards tight bounds on Theta-graphs. *Theoretical Computer Science*, 2014. Forthcoming.

[6] Prosenjit Bose, Anil Maheshwari, Giri Narasimhan, Michiel Smid, and Norbert Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry: Theory and Applications*, 29(3):233–249, 2004.

[7] L. Paul Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the 2nd Annual ACM symposium on Computational Geometry (SoCG 1986)*, pages 169–177, 1986.

[8] L. Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[9] Kenneth L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the 19th ACM Symposium on the Theory of Computing (STOC 1987)*, pages 56–65, 1987.

[10] David P. Dobkin, Steven J. Friedman, and Kenneth J. Supowit. Delaunay graphs are almost as good as complete graphs. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS 1987)*, pages 20–26, 1987.

[11] David Eppstein. Spanning trees and spanners. In J.R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science, 1999.

[12] B. E. Flinchbaugh and L. K. Jones. Strong connectivity in directional nearest-neighbor graphs. *SIAM Journal on Algebraic and Discrete Methods*, 2(4):461–463, 1981.

[13] J. Mark Keil. Approximating the complete Euclidean graph. In *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory (SWAT 88)*, pages 208–213, 1988.

[14] J. Mark Keil and Carl A. Gutwin. The Delaunay triangulation closely approximates the complete Euclidean graph. In *Proceedings of the 1st Workshop on Algorithms and Data Structures (WADS 1989)*, pages 47–56, 1989.

[15] J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7(1):13–28, 1992.

[16] Wah Loon Keng and Ge Xia. The Yao graph Y_5 is a spanner. *ArXiv e-prints*, 2013. arXiv:1307.5030 [cs.CG].

[17] Tamás Lukovski. *New results on geometric spanners and their applications*. PhD thesis, Universität Paderborn, 1999.

[18] Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

[19] Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

[20] André van Renssen. *Theta-graphs and other constrained spanners*. PhD thesis, Carleton University, 2014.

[21] Ge Xia. Improved upper bound on the stretch factor of Delaunay triangulations. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG 2011)*, pages 264–273, 2011.

[22] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

3

THE Θ_5-GRAPH IS A SPANNER

Given any set of points in the plane, we show that the Θ-graph with 5 cones is a geometric spanner with spanning ratio at most $\sqrt{50 + 22\sqrt{5}} \approx 9.960$. This is the first constant upper bound on the spanning ratio of this graph. The upper bound uses a constructive argument that gives a (possibly self-intersecting) path between any two vertices, of length at most $\sqrt{50 + 22\sqrt{5}}$ times the Euclidean distance between the vertices. We also prove that $\frac{1}{2}(11\sqrt{5} - 17) \approx 3.799$ is a lower bound on the spanning ratio.

The results in this chapter were first published in the proceedings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2013) [5], and have subsequently been published in Computational Geometry: Theory and Applications [6]. This chapter contains joint work with Prosenjit Bose, Pat Morin, and André van Renssen.

3.1 INTRODUCTION

As described in Section 2.5, most early research focused on Yao- and Θ-graphs with a large number of cones. However, using the smallest possible number of cones is important for many practical applications, where the cost of a network is mostly determined by the number of edges. One such example is point-to-point wireless networks. These networks use narrow directional wireless transceivers that can transmit over long distances (up to 50km [14, 12]). The cost of an edge in such a network is therefore equal to the cost of the two transceivers that are used at each endpoint of that edge. In such networks, the cost of building Θ_6 is approximately 29% higher than the cost of building Θ_5 if the transceivers are randomly distributed [11]. Assuming that we still want our network to be a spanner, this leads to the natural question: for which values of k are Y_k and Θ_k spanners? Kanj [10] presented this question as one of the main open problems in the area of geometric spanners.

Surprisingly, this question was not studied until quite recently. In 2009, El Molla [9] showed that both Y_2 and Y_3 are not spanners, and these proofs translate to Θ_2 and Θ_3 as well. Since the general proofs (presented in Theorems 2.3 and 2.5) work for $k \geqslant 7$, this left the question open for graphs with 4, 5 and 6 cones. A surprising connection between Θ-graphs and Delaunay triangulations led to the first positive result on this question, when Bonichon et al. [3] showed that Θ_6 is the union of two rotated copies of the empty equilateral

triangle Delaunay triangulation. This graph had been shown to be a 2-spanner by Chew [7] over 20 years earlier. This result was then used by Damian and Raudonis [8] to show that Y_6 is a spanner as well. The next graphs to fall were Y_4 [4] and Θ_4 [2], both of which were shown to be spanners, albeit with very loose upper bounds on the spanning ratio of 663 (Y_4) and 237 (Θ_4). The improvement on the spanning ratio of Yao-graphs with an odd number of cones presented in Theorem 2.4, discovered by Barba et al. [1], settled the matter for Y_5, leaving only Θ_5.

Note that this problem was already claimed to be solved in 1991, by Ruppert and Seidel [13]. Specifically, they wrote:

> In the planar case, some improvement can be made on the constants. In particular, when k is odd, there is an asymmetry between the cones [...] that we can take advantage of by growing paths from both ends. Interestingly, this asymmetry allows us to prove a bound near 10 on the path lengths even for the case $k = 5$. [...] The details are omitted here due to lack of space.

However, to the best of our knowledge they never published a proof of this claim.

In this chapter we present the final piece of this puzzle, by giving the first constant upper bound on the spanning ratio of Θ_5, thereby proving that it is a geometric spanner. We show that the spanning ratio is at most $\sqrt{50 + 22\sqrt{5}} \approx 9.960$. Note that this bound is slightly better than the bound for Y_5 given by Theorem 2.4, although Barba et al. [1] improved the bound for Y_5 to $2 + \sqrt{3} \approx 3.74$ using a different technique. Since the proof for Θ_5 is constructive, it gives us a path between any two vertices, u and w, of length at most $9.960 \cdot |uw|$. Surprisingly, this path can cross itself, a property we observed for the shortest path as well (see Figure 3.1). We also prove that $\frac{1}{2}(11\sqrt{5} - 17) \approx 3.799$ is a lower bound on the spanning ratio.

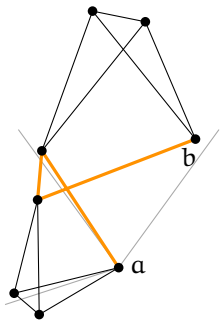


Figure 3.1: An example where the shortest path between two vertices (in bold) in Θ_5 crosses itself.

3.2 CONNECTIVITY

Recall that the *canonical triangle* $\triangle_{uv}$ of two vertices u and v is the triangle bounded by the cone of u that contains v and the line through v perpendicular to the bisector of that cone. We define the size $|\triangle_{uv}|$ of a canonical triangle as the length of one of the sides incident to the apex u. This gives us the useful property that any line segment between u and a point inside the triangle has length at most $|\triangle_{uv}|$.

To introduce the structure of the proof that the spanning ratio of Θ_5 is bounded, we first show that the Θ_5-graph is connected.

THEOREM 3.1. *The Θ_5-graph is connected.*

Proof. We prove that there is a path between any (ordered) pair of vertices in Θ_5, using induction on the size of their canonical triangle. Formally, given two vertices u and w, we perform induction on the rank (relative position) of $\triangle_{uw}$ among the canonical triangles of all pairs of vertices, when ordered by size. For ease of description, we assume that w lies in the right half of C_0^u. The other cases are analogous.

If $\triangle_{uw}$ has rank 1, it is the smallest canonical triangle. Therefore there can be no point closer to u in C_0^u, so the edge (u, w) must be in the graph. This proves the base case.

If $\triangle_{uw}$ has a larger rank, our inductive hypothesis is that there exists a path between any pair of vertices with a smaller canonical triangle. Let a and b be the left and right corners of $\triangle_{uw}$. Let m be the midpoint of ab and let x be the intersection of ab and the bisector of $\angle mub$ (see Figure 3.2a).

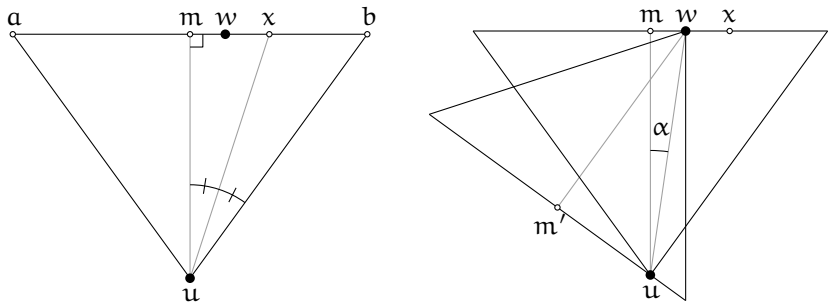


Figure 3.2: (a) The canonical triangle $\triangle_{uw}$. (b) If w lies between m and x, then $\triangle_{wu}$ is smaller than $\triangle_{uw}$.

If w lies to the left of x, consider the canonical triangle $\triangle_{wu}$. Let m' be the midpoint of the side of $\triangle_{wu}$ opposite w and let $\alpha = \angle muw$ (see Figure 3.2b). Note that $\angle uwm' = \frac{\pi}{5} - \alpha$, since um and the vertical

border of $\triangle_{wu}$ are parallel and both are intersected by uw. Using basic trigonometry, we can express the size of $\triangle_{wu}$ as follows.

$$|\triangle_{wu}| = \frac{|wm'|}{\cos\frac{\pi}{5}} = \frac{\cos\angle uwm' \cdot |uw|}{\cos\frac{\pi}{5}}$$
$$= \frac{\cos\left(\frac{\pi}{5}-\alpha\right)\cdot\frac{|um|}{\cos\alpha}}{\cos\frac{\pi}{5}} = \frac{\cos\left(\frac{\pi}{5}-\alpha\right)}{\cos\alpha}\cdot|\triangle_{uw}|$$

Since w lies to the left of x, the angle α is less than $\pi/10$, which means that $\cos(\frac{\pi}{5}-\alpha)/\cos\alpha$ is less than 1. Hence $\triangle_{wu}$ is smaller than $\triangle_{uw}$ and by induction, there is a path between w and u. Since the graph is undirected, we are done in this case. The rest of the proof deals with the case where w lies on or to the right of x.

If $\triangle_{wu}$ is empty, there is an edge between u and w and we are done, so assume that this is not the case. Then there is a vertex v_w that is closest to w in C_3^w (the cone of w that contains u). This gives rise to four cases, depending on the location of v_w (see Figure 3.3a). In each case, we will show that $\triangle_{uv_w}$ is smaller than $\triangle_{uw}$ and hence we can apply induction to obtain a path between u and v_w. Since v_w is the closest vertex to w in C_3, there is an edge between v_w and w, completing the path between u and w.

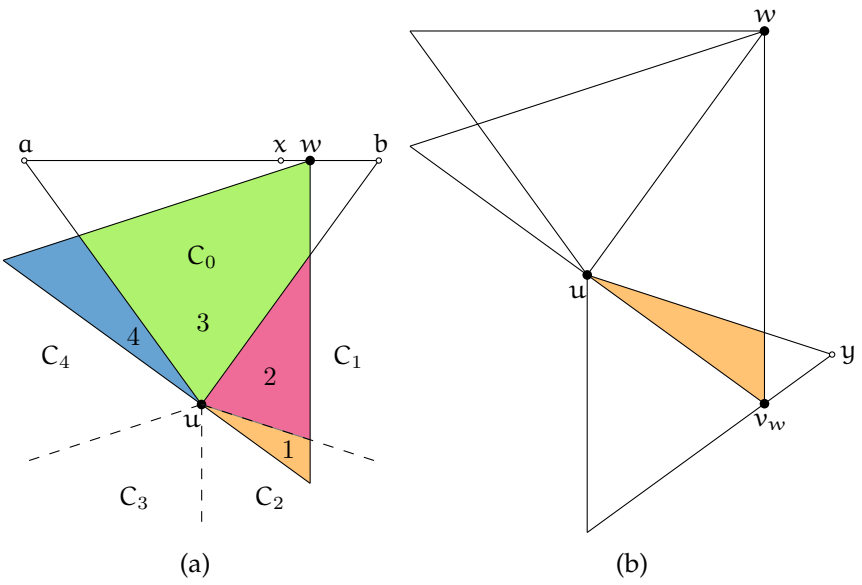


Figure 3.3: (a) The four cases for v_w. (b) Case 1: The situation that maximizes $|\triangle_{uv_w}|$ when v_w lies in C_2^u.

CASE 1. v_w lies in C_2^u. In this case, the size of $\triangle_{uv_w}$ is maximized when v_w lies in the bottom right corner of $\triangle_{wu}$ and w lies on b. Let

y be the rightmost corner of $\triangle_{uv_w}$ (see Figure 3.3b). Using the law of sines, we can express the size of $\triangle_{uv_w}$ as follows.

$$\begin{aligned} |\triangle_{uv_w}| &= |uy| \\ &= \frac{\sin \angle uv_w y}{\sin \angle uyv_w} \cdot |uv_w| \\ &= \frac{\sin \frac{3\pi}{5}}{\sin \frac{3\pi}{10}} \cdot \tan \frac{\pi}{5} \cdot |\triangle_{uw}| \\ &< |\triangle_{uw}| \end{aligned}$$

CASE 2. v_w lies in C_1^u. In this case, the size of $\triangle_{uv_w}$ is maximized when w lies on b and v_w lies almost on w. By symmetry, this gives $|\triangle_{uv_w}| = |\triangle_{uw}|$. However, v_w cannot lie precisely on w and must therefore lie a little closer to u, giving us that $|\triangle_{uv_w}| < |\triangle_{uw}|$.

CASE 3. v_w lies in C_0^u. As in the previous case, the size of $\triangle_{uv_w}$ is maximized when v_w lies almost on w, but since v_w must lie closer to u, we have that $|\triangle_{uv_w}| < |\triangle_{uw}|$.

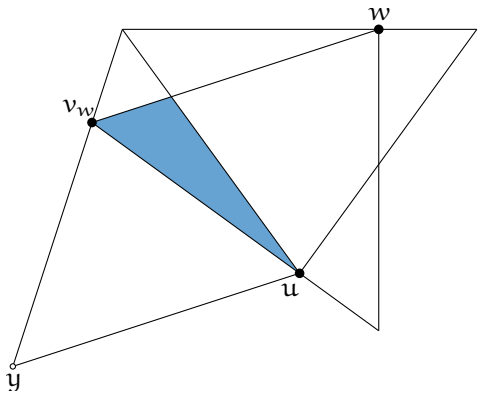


Figure 3.4: Case 4: The situation that maximizes $|\triangle_{uv_w}|$ when v_w lies in C_4^u.

CASE 4. v_w lies in C_4^u. In this case, the size of $\triangle_{uv_w}$ is maximized when v_w lies in the left corner of $\triangle_{wu}$ and w lies on x. Let y be the bottom corner of $\triangle_{uv_w}$ (see Figure 3.4). Since x is the point where $|\triangle_{uw}| = |\triangle_{wu}|$, and $v_w yuw$ forms a parallelogram, $|\triangle_{uv_w}| = |\triangle_{uw}|$. However, by general position, v_w cannot lie on the boundary of $\triangle_{wu}$, so it must lie a little closer to u, giving us that $|\triangle_{uv_w}| < |\triangle_{uw}|$.

Since any vertex in C_3^u would be further from w than u itself, these four cases are exhaustive. □

3.3 SPANNING RATIO

In this section, we prove an upper bound on the spanning ratio of Θ_5.

LEMMA 3.2. *Between any pair of vertices u and w of Θ_5, there is a path of length at most $c \cdot |\triangle_{uw}|$, where $c = 2\left(2+\sqrt{5}\right) \approx 8.473$.*

Proof. We begin in a way similar to the proof of Theorem 3.1. Given an ordered pair of vertices u and w, we perform induction on the size of their canonical triangle. If $|\triangle_{uw}|$ is minimal, there must be a direct edge between them. Since $c > 1$ and any edge inside $\triangle_{uw}$ with endpoint u has length at most $|\triangle_{uw}|$, this proves the base case. The rest of the proof deals with the inductive step, where we assume that there exists a path of length at most $c \cdot |\triangle|$ between every pair of vertices whose canonical triangle $\triangle$ is smaller than $\triangle_{uw}$. As in the proof of Theorem 3.1, we assume that w lies in the right half of C_0^u. If w lies to the left of x, we have seen that $\triangle_{wu}$ is smaller than $\triangle_{uw}$. Therefore we can apply induction to obtain a path of length at most $c \cdot |\triangle_{wu}| < c \cdot |\triangle_{uw}|$ between u and w. Hence we need to concern ourselves only with the case where w lies on or to the right of x.

If u is the vertex closest to w in C_3^w or w is the closest vertex to u in C_0^u, there is a direct edge between them and we are done by the same reasoning as in the base case. Therefore assume that this is not the case and let v_w be the vertex closest to w in C_3^w. We distinguish the same four cases for the location of v_w (see Figure 3.3a). We already showed that we can apply induction on $\triangle_{uv_w}$ in each case. This is a crucial part of the proof for the first three cases.

The basic strategy for the rest of the proof is as follows. If we can find a path of length $g \cdot |\triangle_{uw}|$ that leaves us with a strictly smaller canonical triangle of size $h \cdot |\triangle_{uw}|$, where $h < 1$, we can then apply induction to obtain a path of length $g \cdot |\triangle_{uw}| + c \cdot h \cdot |\triangle_{uw}|$. Since we aim to show that there is a path of length at most $c \cdot |\triangle_{uw}|$, we can derive:

$$\begin{aligned} g \cdot |\triangle_{uw}| + c \cdot h \cdot |\triangle_{uw}| &\leqslant c \cdot |\triangle_{uw}| \\ g + c \cdot h &\leqslant c \\ g &\leqslant (1-h) \cdot c \\ \frac{g}{1-h} &\leqslant c. \end{aligned}$$

Therefore we are done if $g/(1-h) \leqslant 2\left(2+\sqrt{5}\right) \approx 8.473$.

CASE 1. v_w lies in C_2^u. By induction, there exists a path between u and v_w of length at most $c \cdot |\triangle_{uv_w}|$. Since v_w is the closest vertex to w in C_3^w, there is a direct edge between them, giving a path between u and w of length at most $|wv_w| + c \cdot |\triangle_{uv_w}|$.

Given any initial position of v_w in C_2^u, we can increase $|wv_w|$ by moving w to the right. Since this does not change $|\triangle_{uv_w}|$, the worst case occurs when w lies on b. Then we can increase both $|wv_w|$ and $|\triangle_{uv_w}|$ by moving v_w into the bottom corner of $\triangle_{wu}$. This gives rise to the same worst-case configuration as in the proof of Theorem 3.1,

depicted in Figure 3.3b. Building on the analysis there, we can bound the worst-case length of the path as follows.

$$|wv_w| + c \cdot |\triangle uv_w| \quad = \quad \frac{|\triangle uw|}{\cos\frac{\pi}{5}} + c \cdot \frac{\sin\frac{3\pi}{5}}{\sin\frac{3\pi}{10}} \cdot \tan\frac{\pi}{5} \cdot |\triangle uw|$$

This is at most $c \cdot |\triangle uw|$ for $c \geqslant 2(2+\sqrt{5})$. Since we picked $c = 2(2+\sqrt{5})$, the theorem holds in this case. Note that this is one of the cases that determines the value of c.

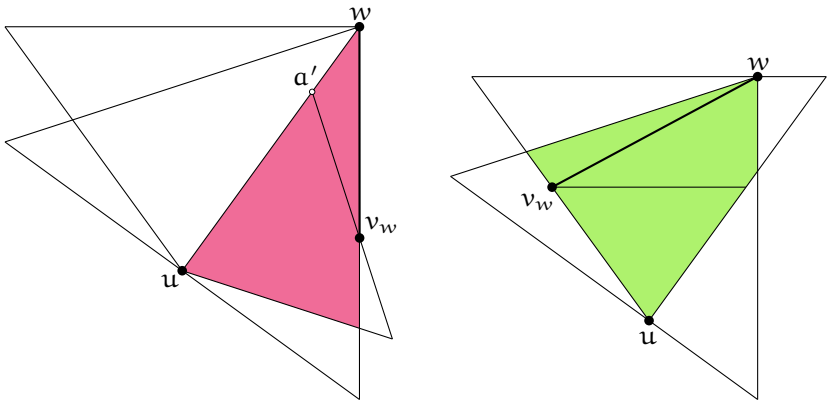


Figure 3.5: (a) Case 2: Vertex v_w lies on the boundary of C_3^w after moving it down along the side of $\triangle uv_w$. (b) Case 3: Vertex v_w lies on the boundary of C_0^u after moving it left along the side of $\triangle uv_w$.

CASE 2. v_w lies in C_1^u. By the same reasoning as in the previous case, we have a path of length at most $|wv_w| + c \cdot |\triangle uv_w|$ between u and w and we need to bound this length by $c \cdot |\triangle uw|$.

Given any initial position of v_w in C_1^u, we can increase $|wv_w|$ by moving w to the right. Since this does not change $|\triangle uv_w|$, the worst case occurs when w lies on b. We can further increase $|wv_w|$ by moving v_w down along the side of $\triangle uv_w$ opposite u until it hits the boundary of C_1^u or C_3^w, whichever comes first (see Figure 3.5a).

Now consider what happens when we move v_w along these boundaries. If v_w lies on the boundary of C_1^u and we move it away from u by ε, $|\triangle uv_w|$ increases by ε. At the same time, $|wv_w|$ might decrease, but not by more than ε. Since $c > 1$, the total path length is maximized by moving v_w as far from u as possible, until it hits the boundary of C_3^w. Once v_w lies on the boundary of C_3^w, we can express the size of $\triangle uv_w$ as follows, where a' is the top corner of $\triangle uv_w$.

$$\begin{aligned} |\triangle uv_w| &= |\triangle uw| - |wa'| \\ &= |\triangle uw| - |wv_w| \cdot \frac{\sin \angle wv_w a'}{\sin \angle wa'v_w} \\ &= |\triangle uw| - |wv_w| \cdot \frac{\sin\frac{\pi}{10}}{\sin\frac{7\pi}{10}} \end{aligned}$$

Now we can express the length of the complete path as follows.

$$\begin{aligned} |wv_w| + c \cdot |\triangle uv_w| &= |wv_w| + c \cdot \left(|\triangle uw| - |wv_w| \cdot \frac{\sin \frac{\pi}{10}}{\sin \frac{7\pi}{10}} \right) \\ &= c \cdot |\triangle uw| - \left(c \cdot \frac{\sin \frac{\pi}{10}}{\sin \frac{7\pi}{10}} - 1 \right) \cdot |wv_w| \end{aligned}$$

Since $c > \sin \frac{7\pi}{10} / \sin \frac{\pi}{10} \approx 2.618$, we have that $c \cdot (\sin \frac{\pi}{10} / \sin \frac{7\pi}{10}) - 1 > 0$. Therefore $|wv_w| + c \cdot |\triangle uv_w| < c \cdot |\triangle uw|$.

CASE 3. v_w lies in C_0^u. Again, we have a path of length at most $|wv_w| + c \cdot |\triangle uv_w|$ between u and w and we need to bound this length by $c \cdot |\triangle uw|$.

Given any initial position of v_w in C_0^u, moving v_w to the left increases $|wv_w|$ while leaving $|\triangle uv_w|$ unchanged. Therefore the path length is maximized when v_w lies on the boundary of either C_0^u or C_3^w, whichever it hits first (see Figure 3.5b).

Again, consider what happens when we move v_w along these boundaries. Similar to the previous case, if v_w lies on the boundary of C_0^u and we move it away from u by ε, $|\triangle uv_w|$ increases by ε, while $|wv_w|$ might decrease by at most ε. Since $c > 1$, the total path length is maximized by moving v_w as far from u as possible, until it hits the boundary of C_3^w. Once there, the situation is symmetric to the previous case, with $|\triangle uv_w| = |\triangle uw| - |wv_w| \cdot (\sin \frac{\pi}{10} / \sin \frac{7\pi}{10})$. Therefore the theorem holds in this case as well.

CASE 4. v_w lies in C_4^u. This is the hardest case. Similar to the previous two cases, the size of $\triangle uv_w$ can be arbitrarily close to that of $\triangle uw$, but in this case $|wv_w|$ does not approach 0. This means that simply invoking the inductive hypothesis on $\triangle uv_w$ does not work, so another strategy is required. We first look at a sub-case where we *can* apply induction directly, before considering the position of v_u, the closest vertex to u in C_0.

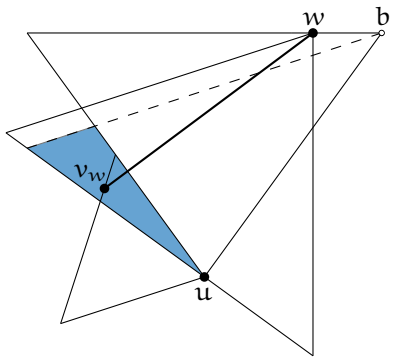


Figure 3.6: Case 4a: Vertex v_w lies in $C_4^u \cap C_3^b$.

CASE 4a. v_w lies in $C_4^u \cap C_3^b$. This situation is illustrated in Figure 3.6. Given any initial position of v_w, moving w to the right onto b increases the total path length by increasing $|wv_w|$ while not affecting $|\triangle_{uv_w}|$. Here we use the fact that v_w already lies in C_3^b, otherwise we would not be able to move w onto b while keeping v_w in C_3^w. Now the total path length is maximized by placing v_w on the left corner of $\triangle_{wu}$. Since this situation is symmetrical to the worst-case situation in Case 1, the theorem holds by the same analysis.

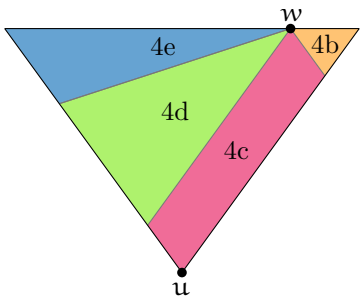


Figure 3.7: The four different cases for the position of v_u.

Next, we distinguish four cases for the position of v_u (the closest vertex to u in C_0), illustrated in Figure 3.7. The cases are: (4b) w lies in $C_4^{v_u}$, (4c) w lies in $C_0^{v_u}$, (4d) w lies in $C_1^{v_u}$ and v_u lies in C_3^w, and (4e) w lies in $C_1^{v_u}$ and v_u lies in C_4^w. These are exhaustive, since the cones C_4, C_0 and C_1 are the only ones that can contain a vertex above the current vertex, and w must lie above v_u, as v_u is closer to u. Further, if w lies in $C_1^{v_u}$, v_u must lie in one of the two opposite cones of w. We can solve the first two cases by applying our inductive hypothesis to $\triangle_{v_u w}$.

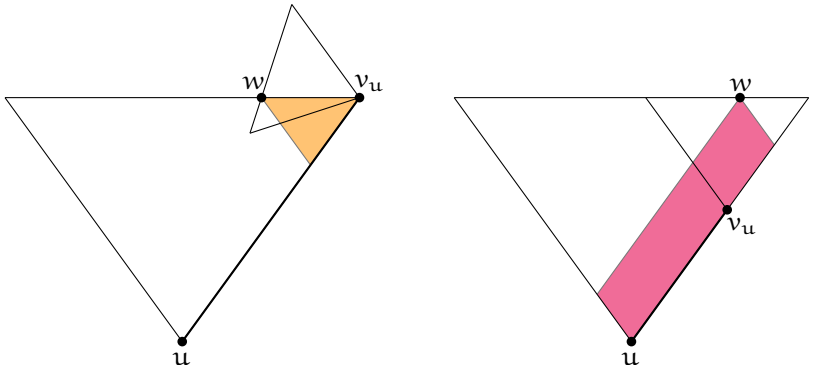


Figure 3.8: (a) The worst-case configuration with w in $C_4^{v_u}$. (b) A configuration with w in $C_0^{v_u}$, after moving v_u onto the right side of C_0^u.

CASE 4b. w lies in $C_4^{v_u}$. To apply our inductive hypothesis, we need to show that $|\triangle_{v_u w}| < |\triangle_{uw}|$. If that is the case, we obtain a

path between v_u and w of length at most $c \cdot |\triangle_{v_u w}|$. Since v_u is the closest vertex to u, there is a direct edge from u to v_u, resulting in a path between u and w of length at most $|uv_u| + c \cdot |\triangle_{v_u w}|$.

Given any initial positions for v_u and w, moving w to the left increases $|\triangle_{v_u w}|$ while leaving $|uv_u|$ unchanged. Moving v_u closer to b increases both. Therefore the path length is maximal when w lies on x and v_u lies on b (see Figure 3.8a). Using the law of sines, we can express $|\triangle_{v_u w}|$ as follows.

$$\begin{aligned} |\triangle_{v_u w}| &= \frac{\sin\frac{3\pi}{5}}{\sin\frac{3\pi}{10}} \cdot |wv_u| = \frac{\sin\frac{3\pi}{5}}{\sin\frac{3\pi}{10}} \cdot \frac{\sin\frac{\pi}{10}}{\sin\frac{3\pi}{5}} \cdot |\triangle_{uw}| \\ &= \frac{\sin\frac{\pi}{10}}{\sin\frac{3\pi}{10}} \cdot |\triangle_{uw}| = \frac{1}{2}\left(3-\sqrt{5}\right) \cdot |\triangle_{uw}| \end{aligned}$$

Since $\frac{1}{2}\left(3-\sqrt{5}\right) < 1$, we have that $|\triangle_{v_u w}| < |\triangle_{uw}|$ and we can apply our inductive hypothesis to $\triangle_{v_u w}$. Since $|uv_u| = |\triangle_{uw}|$, the complete path has length at most $c \cdot |\triangle_{uw}|$ for

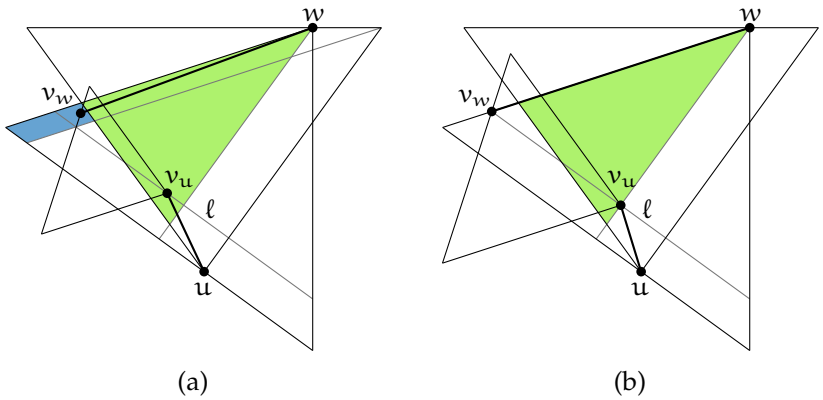
$$c \geqslant \frac{1}{1-\frac{1}{2}\left(3-\sqrt{5}\right)} = \frac{1}{2}\left(1+\sqrt{5}\right) \approx 1.618.$$

CASE 4c. w lies in $C_0^{v_u}$. Since v_u lies in C_0^u, it is clear that $|\triangle_{v_u w}| < |\triangle_{uw}|$, which allows us to apply our inductive hypothesis. This gives us a path between u and w of length at most $|uv_u| + c \cdot |\triangle_{v_u w}|$. For any initial location of v_u, we can increase the total path length by moving v_u to the right until it hits the side of C_0^u (see Figure 3.8b), since $|\triangle_{v_u w}|$ stays the same and $|uv_u|$ increases. Once there, we have that $|uv_u| + |\triangle_{v_u w}| = |\triangle_{uw}|$. Since $c > 1$, this immediately implies that $|uv_u| + c \cdot |\triangle_{v_u w}| \leqslant c \cdot |\triangle_{uw}|$.

To solve the last two cases, we need to consider the positions of both v_u and v_w. Recall that for v_w, there is only a small region left where we have not yet proved the existence of a short path between u and w. In particular, this is the case when v_w lies in cone C_4^u, but not in C_3^b.

CASE 4d. w lies in $C_1^{v_u}$ and v_u lies in C_3^w. We would like to apply our inductive hypothesis to $\triangle_{v_u v_w}$, resulting in a path between v_u and v_w of length at most $c \cdot |\triangle_{v_u v_w}|$. The edges (w, v_w) and (u, v_u) complete this to a path between u and w, giving a total length of at most $|uv_u| + c \cdot |\triangle_{v_u v_w}| + |v_w w|$.

First, note that v_u cannot lie in $\triangle_{w v_w}$, as this region is empty by definition. Since v_w lies in C_4^u, this means that v_w must lie in $C_4^{v_u}$. We first show that $\triangle_{v_u v_w}$ is always smaller than $\triangle_{uw}$, which means that we are allowed to use induction. Given any initial position for v_u, consider the line ℓ through v_u, perpendicular to the bisector of C_3 (see Figure 3.9a). Since v_w cannot be further from w than v_u, the size



(a) (b)

Figure 3.9: (a) The regions where v_u (light) and v_w (dark) can lie. (b) The worst case when v_u lies on a given line ℓ.

of $\triangle_{v_u v_w}$ is maximized when v_w lies on the intersection of ℓ and the top boundary of $\triangle_{wu}$. We can increase $|\triangle_{v_u v_w}|$ further by moving v_u along ℓ until it reaches the bisector of C_3^w (see Figure 3.9b). Since the top boundary of $\triangle_{wu}$ and the bisector of C_3^w approach each other as they get closer to w, the size of $\triangle_{v_u v_w}$ is maximized when v_u lies on the bottom boundary of $\triangle_{wu}$ (ignoring for now that this would move v_u out of $\triangle_{uw}$). Now it is clear that $|\triangle_{v_u v_w}| < |\triangle_{uv_w}|$. Since we already established that $\triangle_{uv_w}$ is smaller than $\triangle_{uw}$ in the proof of Theorem 3.1, this holds for $\triangle_{v_u v_w}$ as well and we can use induction.

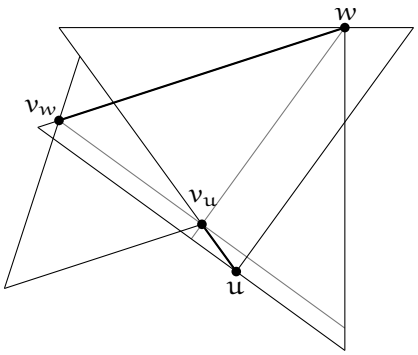


Figure 3.10: The worst case for a fixed position of w.

All that is left is to bound the total length of the path. Given any initial position of v_u, the path length is maximized when we place v_w at the intersection of ℓ and the top boundary of $\triangle_{wu}$, as this maximizes both $|\triangle_{v_u v_w}|$ and $|wv_w|$. When we move v_u away from v_w along ℓ by ε, $|uv_u|$ decreases by at most ε, while $|\triangle_{v_u v_w}|$ increases by $\sin\frac{3\pi}{5}/\sin\frac{3\pi}{10}\cdot\varepsilon > \varepsilon$. Since $c > 1$, this increases the total path length. Therefore the worst case again occurs when v_u lies on the bisector of C_3^w, as depicted in Figure 3.9b. Moving v_u down along the bisector of $\triangle_{wu}$ by ε decreases $|uv_u|$ by at most ε, while increasing $|wv_w|$ by $1/\sin\frac{3\pi}{10}\cdot\varepsilon > \varepsilon$ and increasing $|\triangle_{v_u v_w}|$. Therefore this increases the

total path length and the worst case occurs when v_u lies on the left boundary of $\triangle_{uw}$ (see Figure 3.10).

Finally, consider what happens when we move v_u ε towards u, while moving w and v_w such that the construction stays intact. This causes w to move to the right. Since v_u, w and the left corner of $\triangle_{uw}$ form an isosceles triangle with apex v_u, this also moves v_u ε further from w. We saw before that moving v_u away from w increases the size of $\triangle_{v_u v_w}$. Finally, it also increases $|wv_w|$ by $1/\sin\frac{3\pi}{10} \cdot \varepsilon > \varepsilon$. Thus, the increase in $|wv_w|$ cancels the decrease in $|uv_u|$ and the total path length increases. Therefore the worst case occurs when v_u lies almost on u and v_w lies in the corner of $\triangle_{wu}$, which is symmetric to the worst case of Case 1. Thus the theorem holds by the same analysis.

CASE 4e. w lies in $C_1^{v_u}$ and v_u lies in C_4^w. We split this case into three final sub-cases, based on the position of v_u. These cases are illustrated in Figure 3.11. Note that v_u cannot lie in C_2 or C_3 of v_w, as it lies above v_w. It also cannot lie in $C_4^{v_w}$, as $C_4^{v_w}$ is completely contained in C_4^u, whereas v_u lies in C_0^u. Thus the cases presented below are exhaustive.

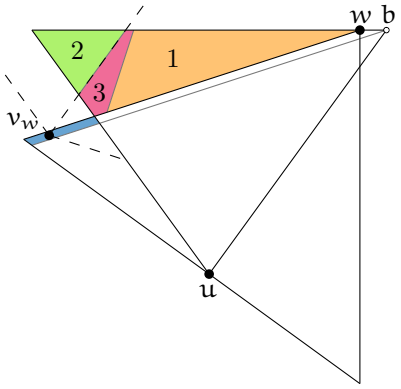


Figure 3.11: The three sub-cases for the position of v_u.

CASE 4e-1. $|\triangle_{wv_u}| \leqslant \frac{c-1}{c} \cdot |\triangle_{uw}|$. If $\triangle_{wv_u}$ is small enough, we can apply our inductive hypothesis to obtain a path between v_u and w of length at most $c \cdot |\triangle_{wv_u}|$. Since there is a direct edge between u and v_u, we obtain a path between u and w of length at most $|uv_u| + c \cdot |\triangle_{wv_u}|$. Any edge from u to a point inside $\triangle_{uw}$ has length at most $|\triangle_{uw}|$, so we can bound the length of the path as follows.

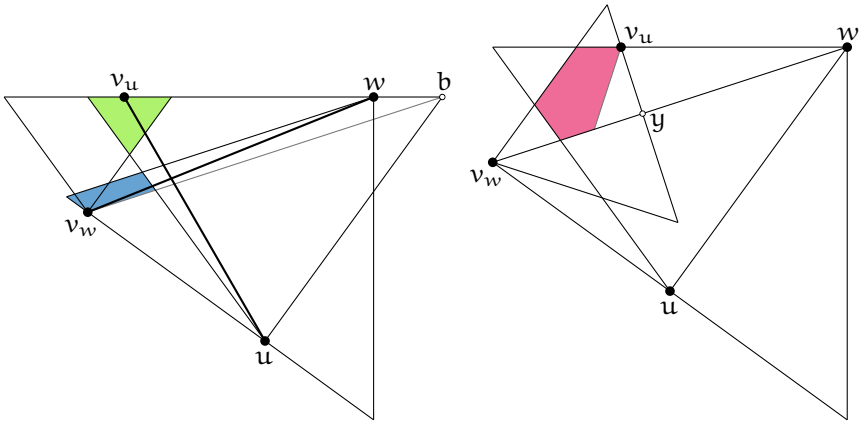
$$\begin{aligned} |uv_u| + c \cdot |\triangle_{wv_u}| &\leqslant |\triangle_{uw}| + c \cdot \frac{c-1}{c} \cdot |\triangle_{uw}| \\ &= |\triangle_{uw}| + (c-1) \cdot |\triangle_{uw}| \\ &= c \cdot |\triangle_{uw}| \end{aligned}$$

In the other two cases, we use induction on $\triangle_{v_w v_u}$ to obtain a path between v_w and v_u of length at most $c \cdot |\triangle_{v_w v_u}|$. The edges (u, v_u)

and (w, v_w) complete this to a (self-intersecting) path between u and w. We can bound the length of these edges by the size of the canonical triangles that contain them, as follows.

$$\begin{aligned} |uv_u| + |wv_w| &\leqslant |\triangle_{uw}| + |\triangle_{wu}| \\ &\leqslant |\triangle_{uw}| + \frac{1}{\cos\frac{\pi}{5}} \cdot |\triangle_{uw}| \\ &= \sqrt{5} \cdot |\triangle_{uw}| \end{aligned}$$

All that is left now is to bound the size of $\triangle_{v_w v_u}$ and express it in terms of $\triangle_{uw}$.



(a) (b)

Figure 3.12: (a) The situation that maximizes $\triangle_{v_w v_u}$ when v_u lies in $C_0^{v_w}$. (b) The worst case when v_u lies in $C_1^{v_w}$.

CASE 4e-2. v_u lies in $C_0^{v_w}$. In this case, the size of $\triangle_{v_w v_u}$ is maximal when v_u lies on the top boundary of $\triangle_{uw}$ and v_w lies at the lowest point in its possible region: the left corner of $\triangle_{bu}$ (see Figure 3.12a). Now we can express $|\triangle_{v_w v_u}|$ as follows.

$$\begin{aligned} |\triangle_{v_w v_u}| &= \frac{\sin\frac{\pi}{10}}{\sin\frac{7\pi}{10}} \cdot |bv_w| \\ &= \frac{\sin\frac{\pi}{10}}{\sin\frac{7\pi}{10}} \cdot \frac{1}{\cos\frac{\pi}{5}} \cdot |\triangle_{uw}| \\ &= 2\left(\sqrt{5}-2\right) \cdot |\triangle_{uw}| \end{aligned}$$

Since $2\left(\sqrt{5}-2\right) < 1$, we can use induction. The total path length is bounded by $c \cdot |\triangle_{uw}|$ for

$$c \geqslant \frac{\sqrt{5}}{1-2\left(\sqrt{5}-2\right)} = 2+\sqrt{5} \approx 4.236.$$

CASE 4e-3. v_u lies in $C_1^{v_w}$. Since $|\triangle_{wv_u}| > \frac{c-1}{c} \cdot |\triangle_{uw}|$, $\triangle_{v_w v_u}$ is maximal when v_w lies on the left corner of $\triangle_{wu}$ and v_u lies on the top boundary of $\triangle_{uw}$, such that $|\triangle_{wv_u}| = \frac{c-1}{c} \cdot |\triangle_{uw}|$ (see Figure 3.12b). Let y be the intersection of $\triangle_{v_w v_u}$ and $\triangle_{wu}$. Note that since v_w lies on the corner of $\triangle_{wu}$, y is also the midpoint of the side of $\triangle_{v_w v_u}$ opposite v_w. We can express the size of $\triangle_{v_w v_u}$ as follows.

$$\begin{aligned}
|\triangle_{v_w v_u}| &= \frac{|v_w y|}{\cos\frac{\pi}{5}} \\
&= \frac{|wv_w| - |wy|}{\cos\frac{\pi}{5}} \\
&= \frac{\frac{|\triangle_{uw}|}{\cos\frac{\pi}{5}} - \cos\frac{\pi}{10} \cdot |wv_u|}{\cos\frac{\pi}{5}} \\
&= \frac{\frac{|\triangle_{uw}|}{\cos\frac{\pi}{5}} - \cos\frac{\pi}{10} \cdot \frac{\sin\frac{3\pi}{10}}{\sin\frac{3\pi}{5}} \cdot |\triangle_{wv_u}|}{\cos\frac{\pi}{5}} \\
&= \frac{\frac{|\triangle_{uw}|}{\cos\frac{\pi}{5}} - \cos\frac{\pi}{10} \cdot \frac{\sin\frac{3\pi}{10}}{\sin\frac{3\pi}{5}} \cdot \frac{c-1}{c} \cdot |\triangle_{uw}|}{\cos\frac{\pi}{5}} \\
&= \left(\frac{1}{c} + 5 - 2\sqrt{5}\right) \cdot |\triangle_{uw}|
\end{aligned}$$

Thus we can use induction for $c > 1/\left(2\sqrt{5}-4\right) \approx 2.118$ and the total path length can be bounded by $c \cdot |\triangle_{uw}|$ for

$$c \geqslant \frac{\sqrt{5}+1}{2\sqrt{5}-4} = \frac{1}{2}\left(7+3\sqrt{5}\right) \approx 6.854. \qquad \square$$

Using this result, we can compute the exact spanning ratio.

THEOREM 3.3. *The Θ_5-graph has spanning ratio at most $\sqrt{50+22\sqrt{5}} \approx 9.960$.*

Proof. Given two vertices u and w, we know from Lemma 3.2 that there is a path between them of length at most $c \cdot \min(|\triangle_{uw}|, |\triangle_{wu}|)$, where $c = 2\left(2+\sqrt{5}\right) \approx 8.473$. This gives an upper bound on the spanning ratio of $c \cdot \min(|\triangle_{uw}|, |\triangle_{wu}|)/|uw|$. We assume without loss of generality that w lies in the right half of C_0^u. Let α be the angle between the bisector of C_0^u and the line uw (see Figure 3.2b). In the proof of Theorem 3.1, we saw that we can express $|\triangle_{wu}|$ and $|uw|$ in terms of α and $|\triangle_{uw}|$, as $|\triangle_{wu}| = (\cos(\frac{\pi}{5} - \alpha)/\cos\alpha) \cdot |\triangle_{uw}|$ and

$|uw| = (\cos\frac{\pi}{5}/\cos\alpha)\cdot|\triangle_{uw}|$, respectively. Using these expressions, we can write the spanning ratio in terms of α.

$$\begin{aligned}\frac{c\cdot\min(|\triangle_{uw}|,|\triangle_{wu}|)}{|uw|} &= \frac{c\cdot\min\left(|\triangle_{uw}|,\frac{\cos(\frac{\pi}{5}-\alpha)}{\cos\alpha}\cdot|\triangle_{uw}|\right)}{\frac{\cos\frac{\pi}{5}}{\cos\alpha}\cdot|\triangle_{uw}|}\\ &= \frac{c}{\cos\frac{\pi}{5}}\cdot\min\left(\cos\alpha,\cos\left(\tfrac{\pi}{5}-\alpha\right)\right)\end{aligned}$$

To get an upper bound on the spanning ratio, we need to maximize the minimum of $\cos\alpha$ and $\cos\left(\frac{\pi}{5}-\alpha\right)$. Since for $\alpha\in[0,\pi/5]$, one is increasing and the other is decreasing, this maximum occurs at $\alpha=\pi/10$, where they are equal. Thus, our upper bound becomes

$$\frac{c}{\cos\frac{\pi}{5}}\cdot\cos\tfrac{\pi}{10} = \sqrt{50+22\sqrt{5}}.$$ □

3.4 LOWER BOUND

In this section, we derive a lower bound on the spanning ratio of the Θ_5-graph.

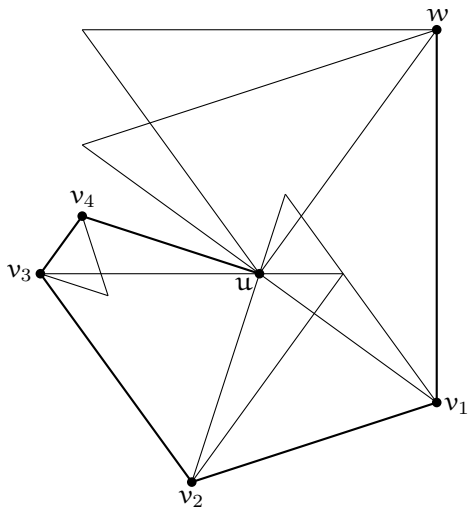


Figure 3.13: A path with a large spanning ratio.

THEOREM 3.4. *The Θ_5-graph has spanning ratio at least $\frac{1}{2}(11\sqrt{5}-17)\approx 3.799$.*

Proof. For the lower bound, we present and analyze a path between two vertices that has a large spanning ratio. The path has the following structure (illustrated in Figure 3.13).

The path can be thought of as being directed from w to u. First, we place w in the right corner of $\triangle_{uw}$. Then we add a vertex v_1 in the bottom corner of $\triangle_{wu}$. We repeat this two more times, each time

adding a new vertex in the corner of $\triangle_{v_i u}$ furthest from u. The final vertex v_4 is placed on the top boundary of $C_1^{v_3}$, such that u lies in $C_1^{v_4}$. Since we know all the angles involved, we can compute the length of each edge, taking $|uw| = 1$ as baseline.

$$|wv_1| = \frac{1}{\cos\frac{\pi}{5}} \qquad |v_1v_2| = |v_2v_3| = 2\sin\tfrac{\pi}{5}\tan\tfrac{\pi}{5}$$
$$|v_3v_4| = \frac{\sin\frac{\pi}{10}}{\sin\frac{3\pi}{5}}\tan\tfrac{\pi}{5} \qquad |v_4u| = \frac{\sin\frac{3\pi}{10}}{\sin\frac{3\pi}{5}}\tan\tfrac{\pi}{5}$$

Since we set $|uw| = 1$, the spanning ratio is simply $|wv_1| + |v_1v_2| + |v_2v_3| + |v_3v_4| + |v_4u| = \frac{1}{2}(11\sqrt{5} - 17) \approx 3.798$. Note that the Θ_5-graph with just these 5 vertices would have a far smaller spanning ratio, as there would be a lot of shortcut edges. However, a graph where this path is the shortest path between two vertices can be found in Figure 3.14, and its construction is described in Table 3.1. □

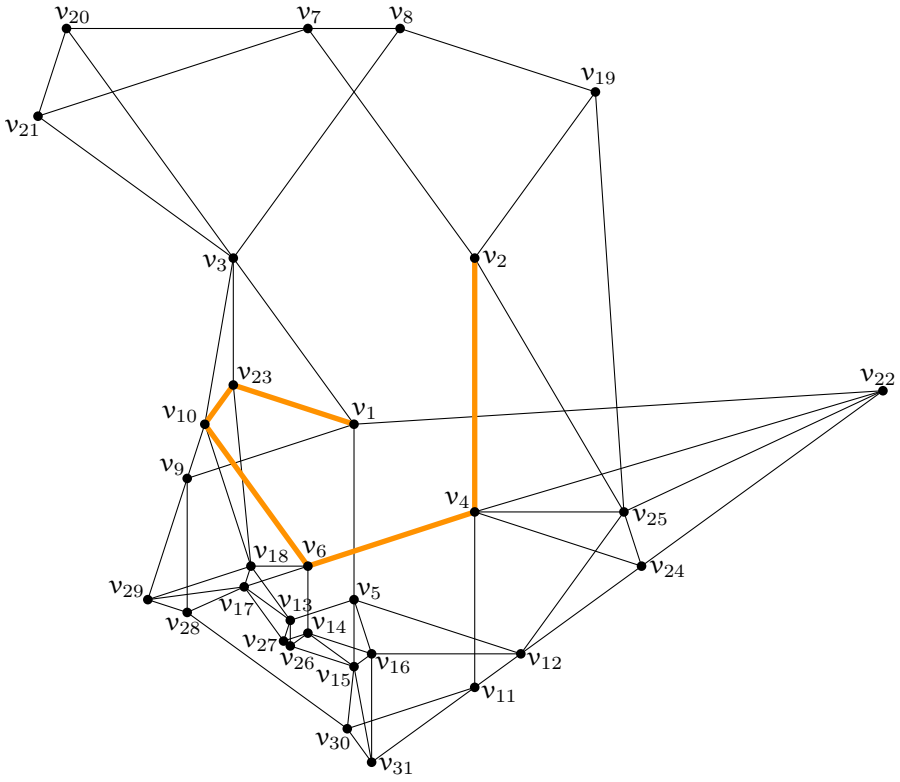


Figure 3.14: A Θ_5-graph with a spanning ratio that matches the lower bound. The shortest path between v_1 and v_2 is indicated in bold.

Table 3.1: Stepwise construction of a Θ_5-graph with a spanning ratio that matches the lower bound (see Figure 3.14).

#	ACTION	SHORTEST PATH
1	Start with a vertex v_1.	-
2	Add v_2 in C_0^u, such that v_2 is arbitrarily close to the top right corner of $\triangle_{v_1v_2}$.	v_1v_2
3	Remove edge (v_1, v_2) by adding two vertices, v_3 and v_4, arbitrarily close to the counter-clockwise corners of $\triangle_{v_1v_2}$ and $\triangle_{v_2v_1}$.	$v_1v_4v_2$
4	Remove edge (v_1, v_4) by adding two vertices, v_5 and v_6, arbitrarily close to the clockwise corner of $\triangle_{v_1v_4}$ and the counter-clockwise corner of $\triangle_{v_4v_1}$.	$v_1v_3v_2$
5	Remove edge (v_2, v_3) by adding two vertices, v_7 and v_8, arbitrarily close to the clockwise corner of $\triangle_{v_2v_3}$ and the counter-clockwise corner of $\triangle_{v_3v_2}$.	$v_1v_6v_4v_2$
6	Remove edge (v_1, v_6) by adding two vertices, v_9 and v_{10}, arbitrarily close to the clockwise corner of $\triangle_{v_1v_6}$ and the counter-clockwise corner of $\triangle_{v_6v_1}$.	$v_1v_5v_4v_2$
7	Remove edge (v_4, v_5) by adding two vertices, v_{11} and v_{12}, arbitrarily close to the counter-clockwise corner of $\triangle_{v_4v_5}$ and the clockwise corner of $\triangle_{v_5v_4}$.	$v_1v_5v_6v_4v_2$
8	Remove edge (v_5, v_6) by adding two vertices, v_{13} and v_{14}, arbitrarily close to the counter-clockwise corner of $\triangle_{v_5v_6}$ and the clockwise corner of $\triangle_{v_6v_5}$.	$v_1v_5v_{14}v_6v_4v_2$
9	Remove edge (v_5, v_{14}) by adding two vertices, v_{15} and v_{16}, arbitrarily close to the counter-clockwise corner of $\triangle_{v_5v_{14}}$ and the clockwise corner of $\triangle_{v_{14}v_5}$.	$v_1v_5v_{13}v_6v_4v_2$
10	Remove edge (v_6, v_{13}) by adding two vertices, v_{17} and v_{18}, arbitrarily close to the clockwise corner of $\triangle_{v_6v_{13}}$ and the counter-clockwise corner of $\triangle_{v_{13}v_6}$.	$v_1v_3v_8v_2$
11	Remove edge (v_2, v_8) by adding a vertex v_{19} in the union of, and arbitrarily close to the intersection point of $\triangle_{v_2v_8}$ and $\triangle_{v_8v_2}$.	$v_1v_3v_7v_2$

#	ACTION	SHORTEST PATH
12	Remove edge (v_3, v_7) by adding two vertices, v_{20} and v_{21}, arbitrarily close to the counter-clockwise corner of $\triangle_{v_3v_7}$ and the clockwise corner of $\triangle_{v_7v_3}$.	$v_1v_5v_{12}v_2$
13	Remove edge (v_2, v_{12}) by adding a vertex v_{22} arbitrarily close to the counter-clockwise corner of $\triangle_{v_2v_{12}}$.	$v_1v_{10}v_6v_4v_2$
14	Remove edge (v_1, v_{10}) by adding a vertex v_{23} in the union of $\triangle_{v_1v_{10}}$ and $\triangle_{v_{10}v_1}$, arbitrarily close to the top boundary of $C_1^{v_{10}}$, and such that v_1 lies in $C_1^{v_{23}}$, arbitrarily close to the bottom boundary.	$v_1v_5v_{12}v_4v_2$
15	Remove edge (v_4, v_{12}) by adding two vertices, v_{24} and v_{25}, arbitrarily close to the counter-clockwise corner of $\triangle_{v_4v_{12}}$ and the clockwise corner of $\triangle_{v_{12}v_4}$.	$v_1v_5v_{13}v_{14}v_6v_4v_2$
16	Remove edge (v_{13}, v_{14}) by adding two vertices, v_{26} and v_{27}, arbitrarily close to the clockwise corner of $\triangle_{v_{13}v_{14}}$ and the counter-clockwise corner of $\triangle_{v_{14}v_{13}}$.	$v_1v_9v_{18}v_6v_4v_2$
17	Remove edge (v_9, v_{18}) by adding two vertices, v_{28} and v_{29}, arbitrarily close to the clockwise corner of $\triangle_{v_9v_{18}}$ and the counter-clockwise corner of $\triangle_{v_{18}v_9}$.	$v_1v_5v_{16}v_{11}v_4v_2$
18	Remove edge (v_{11}, v_{16}) by adding two vertices, v_{30} and v_{31}, arbitrarily close to the counter-clockwise corner of $\triangle_{v_{11}v_{16}}$ and the clockwise corner of $\triangle_{v_{16}v_{11}}$.	$v_1v_{23}v_{10}v_6v_4v_2$

3.5 CONCLUSIONS

We showed that there is a path between every pair of vertices in Θ_5, and this path has length at most $\sqrt{50+22\sqrt{5}} \approx 9.960$ times the straight-line distance between the vertices. This is the first constant upper bound on the spanning ratio of the Θ_5-graph, proving that it is a geometric spanner. We also presented a Θ_5-graph with spanning ratio arbitrarily close to $\frac{1}{2}(11\sqrt{5}-17) \approx 3.799$, thereby giving a lower bound on the spanning ratio. There is still a significant gap between these bounds, which is caused by the upper bound proof mostly ignoring the main obstacle to improving the lower bound: that every edge requires at least one of its canonical triangles to be empty. Hence we believe that the true spanning ratio is closer to the lower bound.

While our proof for the upper bound on the spanning ratio returns a spanning path between the two vertices, it requires knowledge of the neighbours of both the current vertex and the destination vertex. This means that the proof does not lead to a local routing strategy that can be applied in, say, a wireless setting. This raises the question whether it is possible to route *competitively* on this graph, i.e. to discover a spanning path from one vertex to another by using only information local to the current vertices visited so far.

BIBLIOGRAPHY

[1] Luis Barba, Prosenjit Bose, Mirela Damian, Rolf Fagerberg, Wah Loon Keng, Joseph O'Rourke, André van Renssen, Perouz Taslakian, Sander Verdonschot, and Ge Xia. New and improved spanning ratios for Yao graphs. *Journal of Computational Geometry*, 6(2):19–53, 2015. Special issue for SoCG 2014.

[2] Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, André van Renssen, and Sander Verdonschot. On the stretch factor of the Theta-4 graph. In *Proceedings of the 13th Algorithms and Data Structures Symposium (WADS 2013)*, pages 109–120, 2013.

[3] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between theta-graphs, Delaunay triangulations, and orthogonal surfaces. In *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010)*, pages 266–278, 2010.

[4] Prosenjit Bose, Mirela Damian, Karim Douïeb, Joseph O'Rourke, Ben Seamone, Michiel Smid, and Stefanie Wuhrer. $\pi/2$-angle Yao graphs are spanners. *International Journal of Computational Geometry & Applications*, 22(1):61–82, 2012.

[5] Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The θ_5-graph is a spanner. In *Proceedings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2013)*, pages 100–114, 2013.

[6] Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The θ_5-graph is a spanner. *Computational Geometry: Theory and Applications*, 48(2):108–119, 2015.

[7] L. Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[8] Mirela Damian and Kristin Raudonis. Yao graphs span theta graphs. *Discrete Mathematics, Algorithms and Applications*, 4(2):1250024, 16, 2012.

[9] Nawar M. El Molla. *Yao spanners for wireless ad hoc networks*. PhD thesis, Villanova University, 2009.

[10] Iyad Kanj. Geometric spanners: Recent results and open directions. In *Proceedings of the 3rd International Conference on Communications and Information Technology (ICCIT 2013)*, pages 78–82, 2013.

[11] Pat Morin and Sander Verdonschot. On the average number of edges in Theta graphs. In *Proceedings of the 11th Meeting on Analytic Algorithmics and Combinatorics (ANALCO14)*, 2014.

[12] SATEL Oy. What is a radio modem? http://www.satel.com/products/what-is-a-radio-modem. Accessed 13 Dec 2013.

[13] Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

[14] Peter Rysavy. Wireless broadband and other fixed-wireless systems. http://www.networkcomputing.com/netdesign/bb1.html. Accessed 13 Dec 2013.

4

COMPETITIVE ROUTING IN THE HALF-Θ_6-GRAPH

In this chapter, we present a deterministic local routing algorithm that is guaranteed to find a path between any pair of vertices in a half-Θ_6-graph (the half-Θ_6-graph is equivalent to the Delaunay triangulation where the empty region is an equilateral triangle). The length of the path is at most $5/\sqrt{3} \approx 2.887$ times the Euclidean distance between the pair of vertices. Moreover, we show that no local routing algorithm can achieve a better routing ratio, thereby proving that our routing algorithm is optimal. This is somewhat surprising because the spanning ratio of the half-Θ_6-graph is 2, meaning that even though there always exists a path whose lengths is at most twice the Euclidean distance, we cannot always find such a path when routing locally.

Since every triangulation can be embedded in the plane as a half-Θ_6-graph using $O(\log n)$ bits per vertex coordinate via Schnyder's embedding scheme [24], our result provides a competitive local routing algorithm for every such embedded triangulation. Finally, we show how our routing algorithm can be adapted to provide a routing ratio of $15/\sqrt{3} \approx 8.661$ on two bounded degree subgraphs of the half-Θ_6-graph.

The results in this chapter were first published in the proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA 2012) [9], and the proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012) [10]. A paper based on this chapter has been accepted for publication in the SIAM Journal on Computing [8]. This chapter is the result of joint work with Prosenjit Bose, Rolf Fagerberg, and André van Renssen.

4.1 INTRODUCTION

A fundamental problem in networking is the routing of a message from one vertex to another in a graph. What makes routing more challenging is that often in a network the routing strategy must be *local*. Informally, a routing strategy is *local* when the routing algorithm must choose the next vertex to forward a message to based solely on knowledge of the current and destination vertex, and all vertices directly connected to the current vertex. Routing algorithms are considered *geometric* when the underlying graph is embedded in the plane, with edges being straight line segments connecting pairs of points and weighted by the Euclidean distance between their endpoints. Geometric routing algorithms are important in wireless sensor

networks (see [21] and [23] for surveys of the area), since they offer routing strategies that use the coordinates of the vertices to help guide the search as opposed to using the more traditional routing tables.

Papadimitriou and Ratajczak [22] posed a tantalizing question in this area that led to a flurry of activity: Does every 3-connected planar graph have a straight-line embedding in the plane that admits a local routing strategy? They were particularly interested in embeddings that admit a *greedy* strategy, where a message is always forwarded to the vertex whose distance to the destination is the smallest among all vertices in the neighbourhood of the current vertex, including the current vertex. They provided a partial answer by showing that 3-connected planar graphs can always be embedded in $\mathbb{R}^3$ such that they admit a greedy routing strategy. They also showed that the class of complete bipartite graphs, $K_{k,6k+1}$ for all $k \geqslant 1$ cannot be embedded such that greedy routing always succeeds since every embedding has at least one vertex that is not connected to its nearest neighbour. Bose and Morin [11] showed that greedy routing always succeeds on Delaunay triangulations. In fact, a slightly restricted greedy routing strategy known as *greedy-compass* is the first local routing strategy shown to succeed on all triangulations [6]. Dhandapani [14] proved the existence of an embedding that admits greedy routing for every triangulation and Angelini et al. [1] provided a constructive proof. Leighton and Moitra [20] settled Papadimitriou and Ratajczak's question by showing that every 3-connected planar graph can be embedded in the plane such that greedy routing succeeds. One drawback of these embedding algorithms is that the coordinates require $\Omega(n \log n)$ bits per vertex. To address this, He and Zhang [17] and Goodrich and Strash [16] gave succinct embeddings using only $O(\log n)$ bits per vertex. Recently, He and Zhang [18] showed that every 3-connected plane graph admits a succinct embedding with convex faces on which a slightly modified greedy routing strategy always succeeds.

In light of these recent successes, it is surprising to note that the above routing strategies have solely concentrated on finding an embedding that guarantees that a local routing strategy will succeed, but pay little attention to the quality of the resulting path. For example, none of the above routing strategies have been shown to be *competitive*. A geometric routing strategy is said to be competitive if the length of the path found by the routing strategy is not more than a constant times the Euclidean distance between its endpoints. This constant is called the *routing ratio*. Bose and Morin [11] show that many local routing strategies are not competitive, but show how to route competitively on the Delaunay triangulation. However, Dillencourt [15] showed that not all triangulations can be embedded in the plane as Delaunay triangulations. This raises the following question: can *every* triangulation be embedded in the plane such that it admits

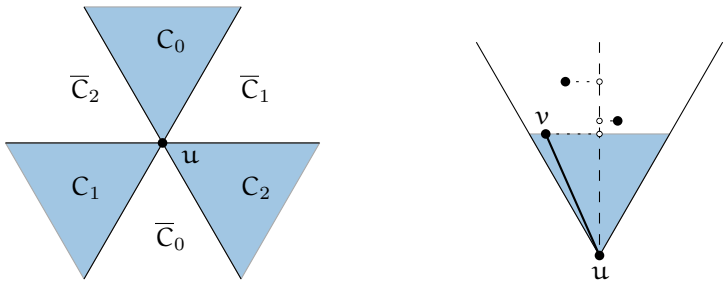
a competitive local routing strategy? We answer this question in the affirmative.

The half-Θ_6-graph was introduced by Bonichon et al. [4], who showed that it is identical to the Delaunay triangulation where the empty region is an equilateral triangle. Although both graphs are identical, the local definition of the half-Θ_6-graph makes it more useful in the context of routing. We formally define the half-Θ_6-graph in the next section. Our main result is a deterministic local routing algorithm that is guaranteed to find a path between any pair of vertices in a half-Θ_6-graph whose length is at most $5/\sqrt{3} \approx 2.887$ times the Euclidean distance between the pair of vertices. On the way to proving our main result, we uncover some local properties of spanning paths in the half-Θ_6-graph. Since Schnyder [24] showed that every triangulation can be embedded in the plane as a half-Θ_6-graph using $O(\log n)$ bits per vertex coordinate, our main result implies that every triangulation has an embedding that admits a competitive local routing algorithm. Moreover, we show that no local routing algorithm can achieve a better routing ratio on a half-Θ_6-graph, implying that our routing algorithm is optimal. This is somewhat surprising because Chew [13] showed that the spanning ratio of the half-Θ_6-graph is 2. Thus, our lower bound provides a separation between the spanning ratio of the half-Θ_6-graph and the best achievable routing ratio on the half-Θ_6-graph. We believe that this is the first separation between the spanning ratio and routing ratio of any graph. It also makes the half-Θ_6-graph one of the few graphs for which tight spanning and routing ratios are known. Finally, we show how our routing algorithm can be adapted to provide a routing ratio of $15/\sqrt{3} \approx 8.661$ on two bounded degree subgraphs of the half-Θ_6-graph introduced by Bonichon et al. [5]. To the best of our knowledge, this is the first competitive routing algorithm on a bounded-degree plane graph.

4.2 PRELIMINARIES

In this section we describe the construction of the half-Θ_6-graph and introduce a few related concepts. Readers who are not familiar with general Θ-graphs may want to read Chapter 2 first. Note that some of the notation in this chapter differs from the notation introduced in Chapter 2. All such differences will be explained in this section.

As the name implies, the half-Θ_6-graph is closely related to the Θ_6-graph. The difference is that every other cone is ignored. To reflect this, the cones are relabelled from $C_0, \ldots, C_5$ to $C_0, \overline{C}_1, C_2, \overline{C}_0, C_1, \overline{C}_2$ (see Figure 4.1a). The cones C_0, C_1 and C_2 are called *positive*, while the others are called *negative*. Note that corresponding positive and negative cones are opposite each other. Combined with their symmetry, this implies that if u lies in $\overline{C}_0^v$ (shorthand for cone $\overline{C}_0$ with apex v), then v must lie in C_0^u.



(a) (b)

Figure 4.1: (a) The positive (highlighted) and negative cones around a vertex u. (b) In each positive cone, u connects to the vertex with the closest projection on the bisector of that cone.

To build the half-Θ_6-graph, we consider each positive cone of every vertex, and add an edge to the closest vertex in that cone (according to the projection onto the bisector, see Figure 4.1b). That is, edges are added to the closest vertex in C_0, C_1, and C_2, but not in the other cones. See Figure 4.2 for an example half-Θ_6-graph. For simplicity, we assume that no two points lie on a line parallel to a cone boundary, guaranteeing that each vertex connects to exactly one vertex in each positive cone. Hence the graph has at most $3n$ edges in total.

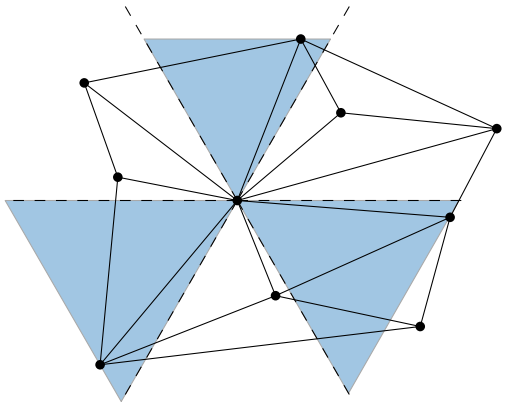


Figure 4.2: An example half-Θ_6-graph.

We slightly modify the concept of *canonical triangle* to take the distinction between positive and negative cones into account. Given two vertices u and v, we now define their canonical triangle as $\triangle_{uv}$ if v lies in a positive cone of u, and $\triangle_{vu}$ if u lies in a positive cone of v. Note that either v lies in a positive cone of u, or u lies in a positive cone of v, so there is exactly one canonical triangle (either $\triangle_{uv}$ or $\triangle_{vu}$) for the pair. With this definition, the construction of the half-Θ_6-graph can alternatively be described as adding an edge between two vertices if and only if their canonical triangle is empty. This property will play an important role in our proofs.

4.3 SPANNING RATIO OF THE HALF-Θ_6-GRAPH

Bonichon et al. [4] showed that the half-Θ_6-graph is a geometric spanner with spanning ratio 2 by showing it is equivalent to the Delaunay triangulation based on empty equilateral triangles, which is known to have spanning ratio 2 [13]. This correspondence also shows that the half-Θ_6-graph is internally triangulated: every face except for the outer face is a triangle (this follows from the duality with the Voronoi diagram, along with the fact that all vertices in the Voronoi diagram have degree 3, provided that no 4 points lie on the same equilateral triangle). In this section, we provide an alternative proof of the spanning ratio of the half-Θ_6-graph. Our proof shows that between any pair of points, there always exists a path with spanning ratio 2 that lies in the canonical triangle. This property plays an important role in our routing algorithm, which we describe in Section 4.5.

For a pair of vertices u and w, our bound is expressed in terms of the angle α between the line from u to w and the bisector of their canonical triangle (see Figure 4.3).

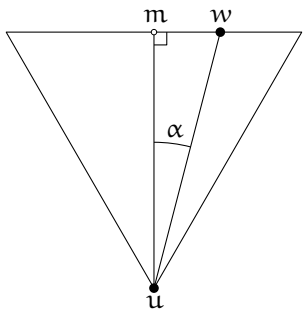


Figure 4.3: Two vertices u and w with their canonical triangle $\triangle_{uw}$. The angle α is the unsigned angle between the line uw and the bisector of the cone containing w.

THEOREM 4.1. *Let u and w be vertices with w in a positive cone of u. Let m be the midpoint of the side of $\triangle_{uw}$ opposing u, and let $\alpha \leqslant \pi/6$ be the smaller of the two unsigned angles between the segments uw and um. Then the half-Θ_6-graph contains a path between u and w of length at most*

$$(\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|,$$

where all vertices on this path lie in $\triangle_{uw}$.

The expression $\sqrt{3} \cdot \cos\alpha + \sin\alpha$ is increasing for $\alpha \in [0, \pi/6]$. By inserting the extreme value $\pi/6$ for α, we arrive at the following.

COROLLARY 4.2. *The spanning ratio of the half-Θ_6-graph is 2.*

We note that the bounds of Theorem 4.1 and Corollary 4.2 are tight: for all values of $\alpha \in [0, \pi/6]$ there exists a point set for which the shortest path in the half-Θ_6-graph for some pair of vertices u and

w has length arbitrarily close to $(\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$. A simple example appears later in the proof of Theorem 4.3.

Proof of Theorem 4.1. Given two vertices u and w, we assume without loss of generality that w lies in C_0^u. We prove the theorem by induction on the rank, when ordered by area, of the triangles $\triangle_{xy}$ for all pairs of points x and y where y lies in a positive cone of x. Let a and b be the upper left and right corner of $\triangle_{uw}$, and let $A = \triangle_{uw} \cap C_1^w$ and $B = \triangle_{uw} \cap C_2^w$, as illustrated in Figure 4.4.

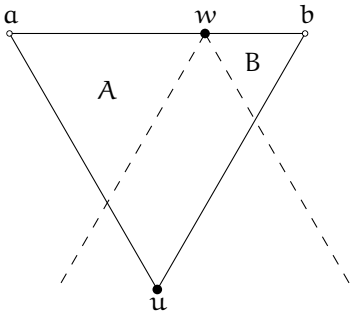


Figure 4.4: The corners a and b, and the regions A and B.

Our inductive hypothesis is the following, where $\delta(u, w)$ denotes the length of the shortest path from u to w in the part of the half-Θ_6-graph induced by the vertices in $\triangle_{uw}$.

1. If A is empty, then $\delta(u, w) \leqslant |ub| + |bw|$.
2. If B is empty, then $\delta(u, w) \leqslant |ua| + |aw|$.
3. If neither A nor B is empty, then $\delta(u, w) \leqslant \max\{|ua| + |aw|, |ub| + |bw|\}$.

We first note that this induction hypothesis implies Theorem 4.1: using the side of $\triangle_{uw}$ as the unit of length, we have from Figure 4.3 that $|wm| = |uw| \cdot \sin\alpha$ and $\sqrt{3}/2 = |um| = |uw| \cdot \cos\alpha$. Hence the induction hypothesis gives us that $\delta(u, w)$ is at most $1 + 1/2 + |wm| = \sqrt{3} \cdot (\sqrt{3}/2) + |wm| = (\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$, as required.

BASE CASE. $\triangle_{uw}$ has rank 1. Since there are no smaller canonical triangles, w must be the closest vertex to u. Hence the edge (u, w) is in the half-Θ_6-graph, and $\delta(u, w) = |uw|$. Using the triangle inequality, we have $|uw| \leqslant \min\{|ua| + |aw|, |ub| + |bw|\}$, so the induction hypothesis holds.

INDUCTION STEP. We assume that the induction hypothesis holds for all pairs of points with canonical triangles of rank up to i. Let $\triangle_{uw}$ be a canonical triangle of rank $i + 1$.

If (u, w) is an edge in the half-Θ_6-graph, the induction hypothesis follows by the same argument as in the base case. If there is no

edge between u and w, let v be the vertex closest to u in the positive cone C_0^u, and let a' and b' be the upper left and right corner of $\triangle_{uv}$. By definition, $\delta(u,w) \leqslant |uv| + \delta(v,w)$, and by the triangle inequality, $|uv| \leqslant \min\{|ua'| + |a'v|, |ub'| + |b'v|\}$.

We perform a case distinction on the location of v: (a) v lies neither in A nor in B, (b) v lies inside A, and (c) v lies inside B. The case where v lies inside B is analogous to the case where v lies inside A, so we only discuss the first two cases, which are illustrated in Figure 4.5.

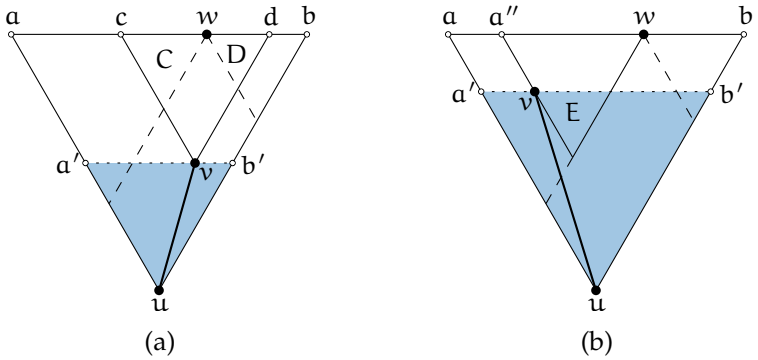


Figure 4.5: The two cases: (a) v lies in neither A nor B, (b) v lies in A.

CASE (a). Let c and d be the upper left and right corner of $\triangle_{vw}$, and let $C = \triangle_{vw} \cap C_1^w$ and $D = \triangle_{vw} \cap C_2^w$ (see Figure 4.5a). Since $\triangle_{vw}$ has smaller area than $\triangle_{uw}$, we apply the inductive hypothesis on $\triangle_{vw}$. Our task is to prove all three statements of the inductive hypothesis for $\triangle_{uw}$.

1. If A is empty, then C is also empty, so by induction $\delta(v,w) \leqslant |vd| + |dw|$. Since v, d, b, and b' form a parallelogram, we have:

$$\begin{aligned} \delta(u,w) &\leqslant |uv| + \delta(v,w) \\ &\leqslant |ub'| + |b'v| + |vd| + |dw| \\ &= |ub| + |bw|, \end{aligned}$$

   which proves the first statement of the induction hypothesis. This argument is illustrated in Figure 4.6a.

2. If B is empty, an analogous argument proves the second statement of the induction hypothesis.

3. If neither A nor B is empty, by induction we have $\delta(v,w) \leqslant \max\{|vc| + |cw|, |vd| + |dw|\}$. Assume, without loss of generality, that the maximum of the right hand side is attained by its second argument $|vd| + |dw|$ (the other case is analogous).

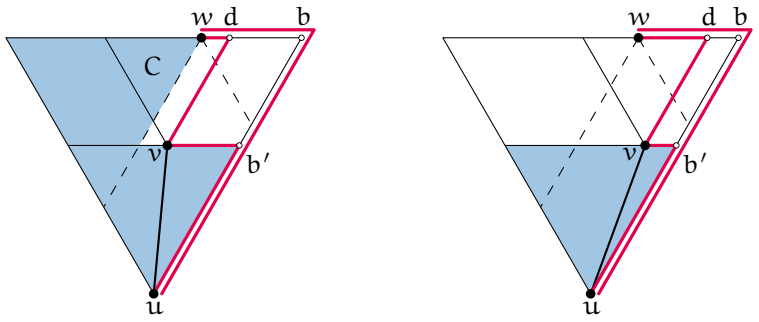


Figure 4.6: Visualization of the path inequalities in two cases: (a) v lies in neither A nor B and one of A or B is empty (cases a.1 and a.2 in our proof), (b) v lies in neither A nor B and neither is empty (case a.3). The paths occurring in the equations are drawn with thick red lines, and light blue areas indicate empty regions.

Since vertices v, d, b, and b' form a parallelogram, we have that:

$$\begin{aligned}\delta(u,w) &\leqslant |uv| + \delta(v,w)\\ &\leqslant |ub'| + |b'v| + |vd| + |dw|\\ &\leqslant |ub| + |bw|\\ &\leqslant \max\{|ua| + |aw|, |ub| + |bw|\},\end{aligned}$$

which proves the third statement of the induction hypothesis. This argument is illustrated in Figure 4.6b.

CASE (b). Let $E = \triangle_{uv} \cap \triangle_{wv}$, and let a'' be the upper left corner of $\triangle_{wv}$ (see Figure 4.5b). Since v is the closest vertex to u in one of its positive cones, $\triangle_{uv}$ is empty and hence E is also empty. Since $\triangle_{wv}$ is smaller than $\triangle_{uw}$, we can apply induction on it. As E is empty, the first statement of the induction hypothesis for $\triangle_{wv}$ applies, giving us that $\delta(v,w) \leqslant |va''| + |a''w|$. Since $|uv| \leqslant |ua'| + |a'v|$ and v, a'', a, and a' form a parallelogram, we have that $\delta(u,w) \leqslant |ua| + |aw|$, proving the second and third statement in the induction hypothesis for $\triangle_{uw}$. This argument is illustrated in Figure 4.7. Since v lies in A, the first statement in the induction hypothesis for $\triangle_{uw}$ is vacuously true. □

4.4 REMARKS ON THE SPANNING RATIO

The Θ_6-graph, introduced by Keil and Gutwin [19], is similar to the half-Θ_6-graph except that all 6 cones are positive cones. Thus, Θ_6 is the union of two copies of the half-Θ_6-graph, where one half-Θ_6-graph is rotated by $\pi/3$ radians. The half-Θ_6-graph and Θ_6 both have a spanning ratio of 2, with lower bound examples showing that it is tight for both graphs. This is surprising since Θ_6 can have twice the number of edges of the half-Θ_6-graph.

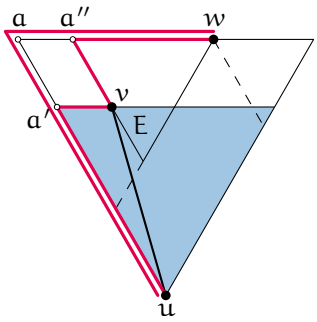


Figure 4.7: Visualization of the path inequalities when v lies in A or B (case b).

Note that since Θ_6 consists of two rotated copies of the half-Θ_6-graph, one question that comes to mind is what is the best spanning ratio if one is to construct a graph consisting of two rotated copies of the half-Θ_6-graph? Can one do better than a spanning ratio of 2? Consider the following construction. Build two half-Θ_6-graphs as described in Section 4.2, but rotate each cone of the second graph by $\pi/6$ radians. For each pair of vertices, there is a path of length at most $\sqrt{3}\cos\alpha + \sin\alpha$ times the Euclidean distance between them, where α is the angle between the line connecting the vertices in question, and the closest bisector. Since this function is increasing, the spanning ratio is defined by the maximum possible angle to the closest bisector, which is $\pi/12$ radians, giving a spanning ratio of roughly 1.932.

By using k copies, we improve the spanning ratio even further: if each is rotated by $\pi/(3k)$ radians, we get a spanning ratio of $\sqrt{3}\cos\frac{\pi}{6k} + \sin\frac{\pi}{6k}$. This is better than the known upper bounds for Θ_{3k} [12] and Y_{3k} [3] for $k \leqslant 4$.

4.5 ROUTING IN THE HALF-Θ_6-GRAPH

In this section, we give matching upper and lower bounds for the routing ratio on the half-Θ_6-graph. We begin by defining our model. Formally, a routing algorithm A is a deterministic k-local, m-memory routing algorithm, if the vertex to which a message is forwarded from the current vertex s is a function of s, t, $N_k(s)$, and M, where t is the destination vertex, $N_k(s)$ is the k-neighbourhood of s and M is a memory of size m, stored with the message. The k-neighbourhood of a vertex s is the set of vertices in the graph that can be reached from s by following at most k edges. For our purposes, we consider a unit of memory to consist of a $\log_2 n$ bit integer or a point in $\mathbb{R}^2$. Our model also assumes that the only information stored at each vertex of the graph is $N_k(s)$. Since our graphs are geometric, we identify each vertex by its coordinates in the plane. Note that while many local routing models allow the algorithm to use the location of the

source vertex (where the routing algorithm started) in addition to the current vertex and destination vertex, our model does not.

A routing algorithm is d-*competitive* provided that the total distance travelled by the message is never more than d times the Euclidean distance between source and destination. Analogous to the spanning ratio, the *routing ratio* of an algorithm is the smallest d for which it is d-competitive.

We present a deterministic 1-local 0-memory routing algorithm that achieves the upper bounds, but our lower bounds hold for any deterministic k-local 0-memory algorithm, provided k is a constant. Our bounds are expressed in terms of the angle α between the line from the source to the destination and the bisector of their canonical triangle (see Figure 4.3).

THEOREM 4.3. *Let u and w be two vertices, with w in a positive cone of u. Let m be the midpoint of the side of $\triangle_{uw}$ opposing u, and let α be the unsigned angle between the lines uw and um. There is a deterministic 1-local 0-memory routing algorithm on the half-Θ_6-graph for which every path followed has length at most*

i) $(\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$ *when routing from u to w,*

ii) $(5/\sqrt{3} \cdot \cos\alpha - \sin\alpha) \cdot |uw|$ *when routing from w to u,*

and this is best possible for deterministic k-local, 0-memory routing algorithms, where k is constant.

The first expression is increasing for $\alpha \in [0, \pi/6]$, while the second expression is decreasing. Inserting the extreme values $\pi/6$ and 0 for α, we get the following worst case version of Theorem 4.3.

COROLLARY 4.4. *Let u and w be two vertices, with w in a positive cone of u. There is a deterministic 1-local 0-memory routing algorithm on the half-Θ_6-graph with routing ratio*

i) *2 when routing from u to w,*

ii) *$5/\sqrt{3} \approx 2.887$ when routing from w to u,*

and this is best possible for deterministic k-local, 0-memory routing algorithms, where k is constant.

Since the spanning ratio of the half-Θ_6-graph is 2, the second lower bound shows a separation between the spanning ratio and the best possible routing ratio in the half-Θ_6-graph.

Since every triangulation can be embedded in the plane as a half-Θ_6-graph using $O(\log n)$ bits per vertex via Schnyder's embedding scheme [24], an important implication of Theorem 4.3 is the following.

COROLLARY 4.5. *Every n-vertex triangulation can be embedded in the plane using $O(\log n)$ bits per coordinate such that the embedded triangulation admits a deterministic 1-local routing algorithm with routing ratio at most $5/\sqrt{3} \approx 2.887$.*

4.5.1 *Positive routing*

In the remainder of this section we prove Theorem 4.3. We first consider the case where the destination lies in a positive cone of the source. We start with a proof of the lower bound, followed by a description of the routing algorithm and a proof of the upper bound.

LEMMA 4.6 (Lower bound for positive routing). *Let u and w be two vertices, with w in a positive cone of u. Let m be the midpoint of the side of $\triangle_{uw}$ opposing u, and let α be the unsigned angle between the lines uw and um. For any deterministic k-local, 0-memory routing algorithm, there are instances for which the path followed has length at least $(\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$ when routing from u to w.*

Proof. Let the side of $\triangle_{uw}$ be the unit of length. From Figure 4.3, we have $|wm| = |uw| \cdot \sin\alpha$ and $\sqrt{3}/2 = |um| = |uw| \cdot \cos\alpha$. From Figure 4.8, the spanning ratio of the half-Θ_6-graph is at least $1 + 1/2 + |wm| = \sqrt{3} \cdot (\sqrt{3}/2) + |wm| = (\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$, since the point in the upper left corner of $\triangle_{uw}$ can be moved arbitrarily close to the corner. As there is no shorter path between u and w, this is a lower bound for *any* routing algorithm. □

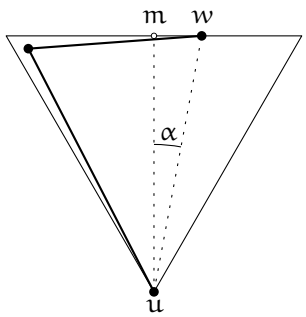


Figure 4.8: The lower bound example when routing to a vertex in a positive cone.

ROUTING ALGORITHM. While routing, let s denote the current vertex and let t denote the fixed destination (i.e. t corresponds to w in Theorem 4.3). To be deterministic, 1-local, and 0-memory, the routing algorithm needs to determine which edge (s, v) to follow next based only on s, t, and the neighbours of s. We say we are *routing positively* when t is in a positive cone of s, and *routing negatively* when t is in a negative cone. (Note the distinction between "positive routing" and "routing positively": the first describes the conditions *at the start* of the routing process, while the second does so *during* the routing process. In other words, positive routing describes a routing process that starts by routing positively. It is very common for positive routing to include situations where we are routing negatively, see e.g. the bottom part of Figure 4.11.)

For ease of description, we assume without loss of generality that t is in cone C_0^s when routing positively, and in cone $\overline{C}_0^s$ when routing negatively. When routing positively, $\triangle_{st}$ intersects only C_0^s among the cones of s. When routing negatively, $\triangle_{ts}$ intersects $\overline{C}_0^s$, as well as the two positive cones C_1^s and C_2^s. Let $X_0 = \overline{C}_0^s \cap \triangle_{ts}$, $X_1 = C_1^s \cap \triangle_{ts}$, and $X_2 = C_2^s \cap \triangle_{ts}$. Let a be the corner of $\triangle_{ts}$ contained in X_1 and b the corner of $\triangle_{ts}$ contained in X_2. These definitions are illustrated in Figure 4.9.

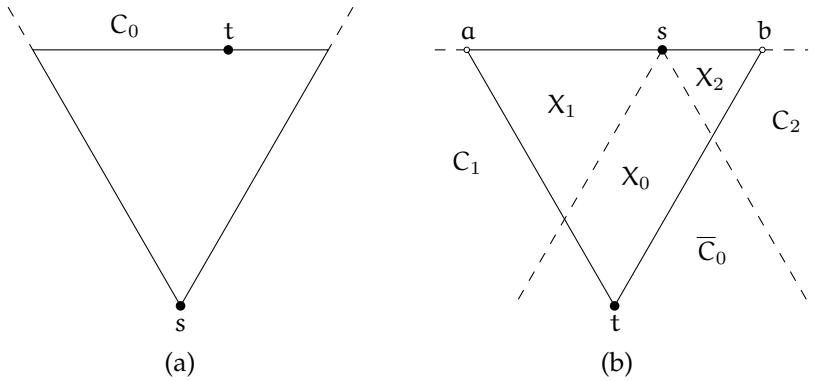


Figure 4.9: Routing terminology when (a) routing positively and (b) routing negatively.

The routing algorithm will only follow edges (s, v) where v lies in the canonical triangle of s and t. Routing positively is straightforward since there is exactly one edge (s, v) with $v \in \triangle_{st}$, by the construction of the half-Θ_6-graph. The challenge is to route negatively. When routing negatively, at least one edge (s, v) with $v \in \triangle_{ts}$ exists, since by Theorem 4.1, s and t are connected by a path in $\triangle_{ts}$. The core of our routing algorithm is how to choose which edge to follow when there is more than one. Intuitively, when routing negatively, our algorithm tries to select an edge that makes measurable progress towards the destination. When no such edge exists, we are forced to take an edge that does not make measurable progress, however we are able to then deduce that certain regions within the canonical triangle are empty. This allows us to bound the total distance travelled while not making measurable progress. We provide a formal description of our routing algorithm below.

Our routing algorithm can be in one of four cases. We call the situation when routing positively case A, and divide the situation when routing negatively into three further cases: both X_1 and X_2 are empty (case B), either X_1 or X_2 is empty (case C), or neither is empty (case D). Since X_1 and X_2 correspond to positive cones of s, each contains the endpoint of at most one edge (s, v). These edges contain a lot of information about the regions X_1 and X_2. In particular, if there is no edge in the corresponding cone, then the entire cone must be empty. And if there is an edge, but its endpoint lies outside of the region, the region is guaranteed to be empty. This allows our algorithm to *locally*

determine if X_1 and X_2 are empty, and therefore which case we are in.

Since we are routing to a destination in a positive cone of the source, our routing algorithm starts in case A. Routing in this case is straightforward, as there is only one edge (s, v) with v in $\triangle_{ts}$ that we can follow. We now turn our attention to routing in cases B and C (it turns out case D never occurs when routing to a destination in a positive cone of the source; we come back to it when describing negative routing in Section 4.5.2).

In case B, both X_1 and X_2 are empty, so there must be edges (s, v) with $v \in X_0$, as s and t are connected by a path in $\triangle_{ts}$ by Theorem 4.1. If $|as| \geqslant |sb|$, the routing algorithm follows the last edge in clockwise order around s; if $|as| < |sb|$, it follows the first edge. In short, when both sides of $\triangle_{ts}$ are empty, the routing algorithm favours staying close to the largest empty side of $\triangle_{ts}$. Note that $|as|$ and $|sb|$ can be computed locally from the coordinates of s and t.

In case C, exactly one of X_1 or X_2 is empty. If there exist edges (s, v) with $v \in X_0$, the routing algorithm will follow one of these, choosing among them in the following way: If X_1 is empty, it chooses the last edge in clockwise order around s. Else X_2 is empty, and it chooses the first edge in clockwise order around s. In short, the routing algorithm favours staying close to the empty side of $\triangle_{ts}$. If no edges (s, v) with $v \in X_0$ exist, the routing algorithm follows the single edge (s, v) with v in X_1 or X_2.

UPPER BOUND. The proof of the upper bound uses a potential function ϕ, defined as follows for each of the cases A, B, and C. For the potential in case C, $x \in \{a, b\}$ is the corner contained in the non-empty one of the two areas X_1 and X_2.

$$\begin{aligned} \text{Case A:} \quad \phi &= |sa| + \max(|at|, |tb|) \\ \text{Case B:} \quad \phi &= |ta| + \min(|as|, |sb|) \\ \text{Case C:} \quad \phi &= |ta| + |sx| \end{aligned}$$

This definition is illustrated in Figure 4.10. We will refer to the first term of ϕ (i.e., $|sa|$ in case A, $|ta|$ in cases B, and C) as the *vertical part* of ϕ and to the rest as the *horizontal part*. Note that since all sides of the canonical triangle have equal length, a and b are interchangeable in the vertical part. The proof makes extensive use of the following observation about equilateral triangles:

OBSERVATION 4.7. *In an equilateral triangle, the diameter (the longest distance defined by any two points in the triangle) is equal to the side length.*

Our aim is to prove the following claim: for any routing step, the reduction in ϕ is at least as large as the length of the edge followed. This allows us to 'pay' for each edge with the difference in potential,

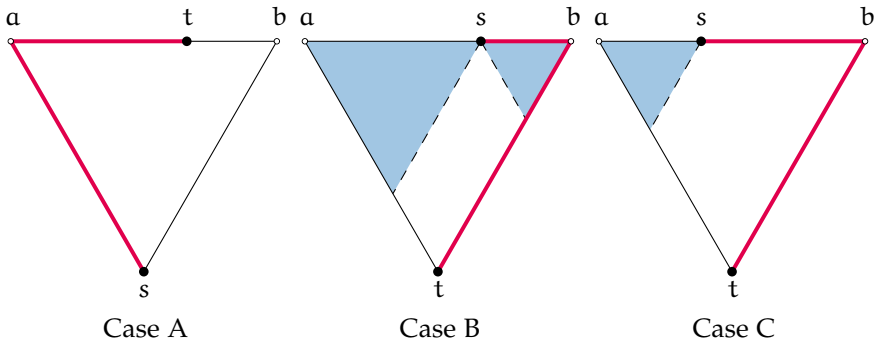


Figure 4.10: The potential ϕ in each case. The thick lines designate potential and shaded areas are empty.

thereby bounding the total length of the path by the initial potential. We do this by case analysis of the possible routing steps.

CASE A. For a routing step starting in case A, v can be in a negative or a positive cone of t. The first situation leads to case A again. The second leads to case B or C, since the area of $\triangle_{st}$ between s and v must be empty by construction of the half-Θ_6-graph. These situations are illustrated in Figure 4.11.

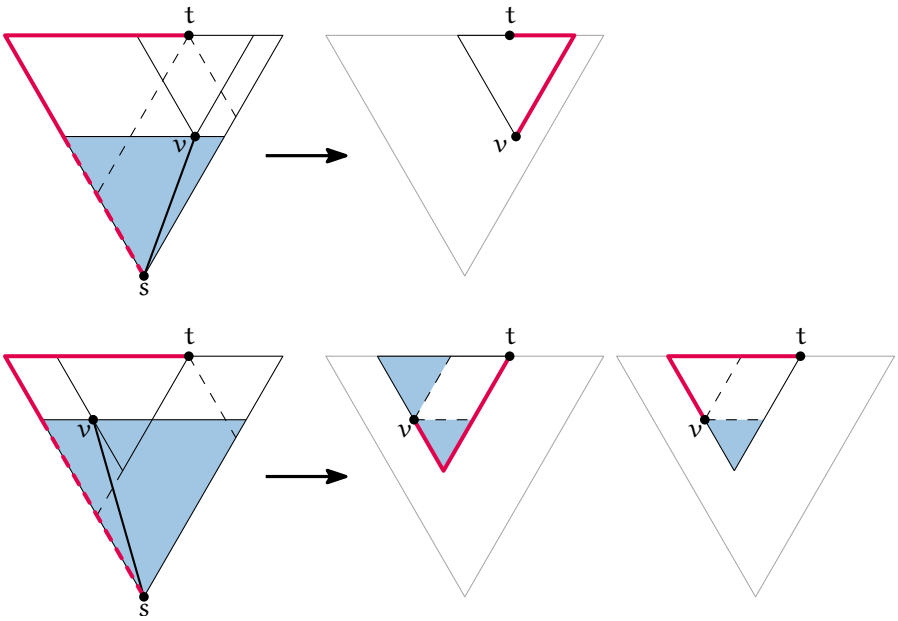


Figure 4.11: Routing in case A. (Top) v lies in a negative cone of t, (Bottom) v lies in a positive cone of t. Dashed red lines indicate which parts of the potential are used to pay for the edge.

If we remain in case A after following edge (s, v), the reduction of the vertical part of ϕ (dashed in Figure 4.11a) is at least as large as $|sv|$ by Observation 4.7. Therefore we can use it to pay for this step. Since $\triangle_{vt}$ is contained in $\triangle_{st}$, both $|at|$ and $|bt|$ decrease. Thus the

horizontal part of ϕ decreases too, as it is the maximum of the two. Hence the claim holds for this situation.

For the situation ending in case C (the second illustration after the arrow in Figure 4.11b), we again use the reduction of the vertical part of ϕ to pay for the step. The rest of the vertical part precisely covers the new horizontal part. Since $\triangle_{tv}$ is contained in $\triangle_{st}$, the new vertical part is a portion of either ta or tb. This can be covered by the current horizontal part, as it is the maximum of $|ta|$ and $|tb|$. Thus the claim holds for this situation as well. Finally, for the situation ending in case B, the final value of ϕ is at most that of the situation ending in case C, so again the claim holds.

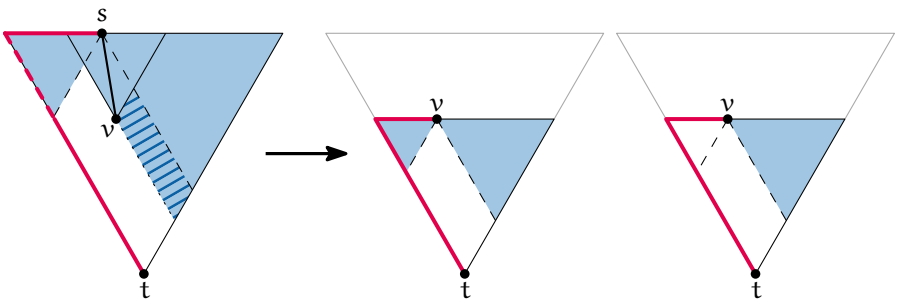


Figure 4.12: Routing in case B.

CASE B. A routing step starting in case B (illustrated in Figure 4.12) cannot lead to case A, as the step stays within $\triangle_{ts}$. We first show that it always results in Case B or C, meaning that at least one of X_1 or X_2 is empty again. The algorithm follows an edge (s, v) with $v \in X_0$. If s is to the left of t, it follows the first edge in clockwise order around s, otherwise it follows the last one. We consider only the case where s is to the left of t, the other case is symmetric. By the construction of the half-Θ_6-graph, the existence of the edge (s, v) implies that $\triangle_{vs}$ is empty. It follows that the hatched area in Figure 4.12 is also empty: if not, the topmost point in it would have an edge to s, while coming before v in the clockwise order around s, contradicting the choice of v by the routing algorithm. Therefore X_2 will again be empty, resulting in case B or C.

By Observation 4.7, the reduction in the vertical part of ϕ is at least as large as $|sv|$. In addition, the horizontal part of ϕ can only decrease. If it remains on the same side of the triangle, this follows from the fact that v lies in X_0 and $\triangle_{tv}$ is contained in $\triangle_{ts}$. And the only case where the potential switches sides, is when we end up in case B again but the other side is shorter than the current one, reducing the potential even further. Hence the claim holds.

CASE C. As in the previous case, a routing step starting in case C cannot lead to case A and we show that it cannot lead to case D, either. There are two situations, depending on whether edges (s, v)

with $v \in X_0$ exist. For the situation where such edges do exist (illustrated in the top part of Figure 4.13), the analysis is exactly the same as for a routing step starting in case B.

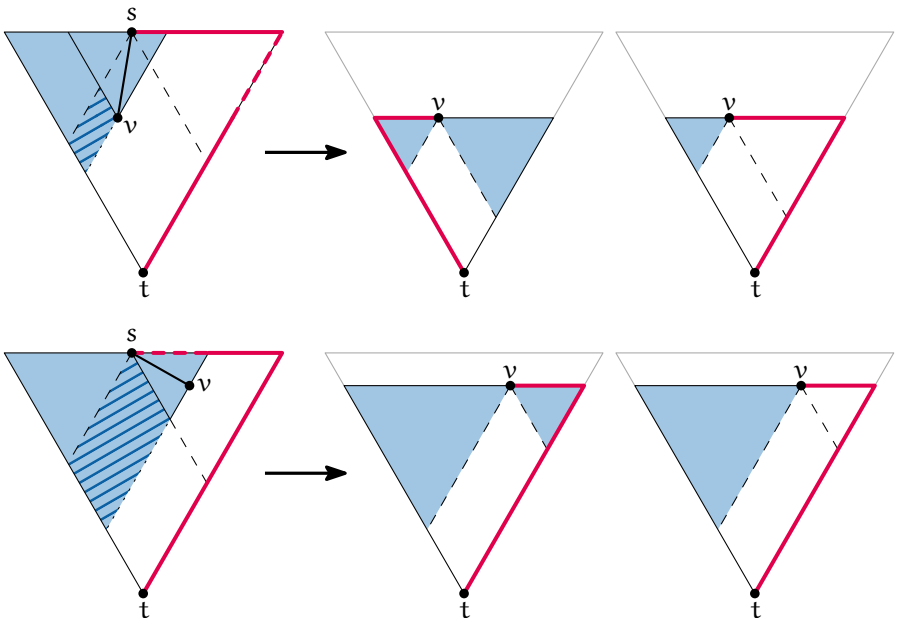


Figure 4.13: Routing in case C.

For the situation where edges (s, v) with $v \in X_0$ do not exist, the start of the step is illustrated on the left of the arrow in the bottom part of Figure 4.13. Again, $\triangle_{sv}$ must be empty by the construction of the half-Θ_6-graph, which implies that the hatched area must also be empty: if not, the topmost point in it would have an edge to s, contradicting that edges (s, v) with $v \in X_0$ do not exist. Thus, the routing step can only lead to case B or C. Looking at the potential, the vertical part can only decrease, and by Observation 4.7, the reduction of the horizontal part of ϕ is at least as large as $|sv|$. Thus we can pay for this step as well and the claim holds in both situations.

LEMMA 4.8 (Upper bound for positive routing). *Let u and w be two vertices, with w in a positive cone of u. Let m be the midpoint of the side of $\triangle_{uw}$ opposing u, and let α be the unsigned angle between the lines uw and um. There is a deterministic 1-local 0-memory routing algorithm on the half-Θ_6-graph for which every path followed has length at most $(\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$ when routing from u to w.*

Proof. That the algorithm is deterministic, 1-local, and 0-memory follows from the description of the algorithm, so we only need to prove the bound on the distance. We showed that for any routing step, the reduction in ϕ is at least as large as the length of the edge followed. Since ϕ is always non-negative, this implies that no path followed can be longer than the initial value of ϕ. As all edges have strictly positive length, the routing algorithm must terminate. Since

we are routing to a vertex in a positive cone, we start in case A, with an initial potential of $|ua| + \max(|aw|, |wb|)$. Taking the side of $\triangle_{uw}$ as the unit of length reduces this to $1 + 1/2 + |wm|$, and using the same analysis as in Lemma 4.6, we obtain the desired bound of $(\sqrt{3} \cdot \cos\alpha + \sin\alpha) \cdot |uw|$. □

4.5.2 *Negative routing*

Next we turn our attention to the case when we are routing to a destination in a negative cone of the source. We start by deriving a lower bound, then present the required extensions to our routing algorithm and finish with the matching upper bound.

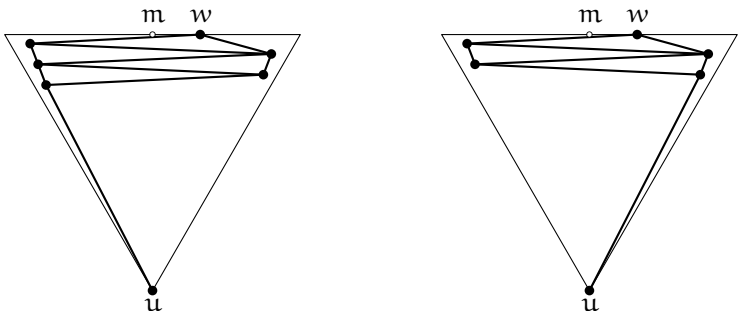


Figure 4.14: The lower bound instances for routing to a vertex in a negative cone.

LEMMA 4.9 (Lower bound for negative routing). *Let u and w be two vertices, with w in a positive cone of u. Let m be the midpoint of the side of $\triangle_{uw}$ opposing u, and let α be the unsigned angle between the lines uw and um, and let k be a constant. For any deterministic k-local 0-memory routing algorithm, there are instances for which the path followed has length at least $(5/\sqrt{3} \cdot \cos\alpha - \sin\alpha) \cdot |uw|$ when routing from w to u.*

Proof. Consider the two instances in Figure 4.14. Any deterministic 1-local 0-memory routing algorithm has information about direct neighbours only. Hence, it cannot distinguish between the two instances when routing out of w. This means that it routes to the same neighbour of w in both instances, and either choice of neighbour leads to a non-optimal route in one of the two instances. The smallest loss occurs when the choice is towards the closest corner of $\triangle_{uw}$, for which Figure 4.14a is the bad instance. If we let the side of $\triangle_{uw}$ be the unit of length, this gives a lower bound of $(1/2 - |wm|) + 1 + 1 = 5/2 - |wm|$, since the points in the corners of $\triangle_{uw}$ can be moved arbitrarily close to the corners while keeping their relative positions. Using that $|wm| = |uw| \cdot \sin\alpha$ and $\sqrt{3}/2 = |um| = |uw| \cdot \cos\alpha$, the lower bound reduces to $(5/\sqrt{3} \cdot \cos\alpha - \sin\alpha) \cdot |uw|$. By appropriately adding $\Omega(k)$ points close to the corners such that u is not in the k-

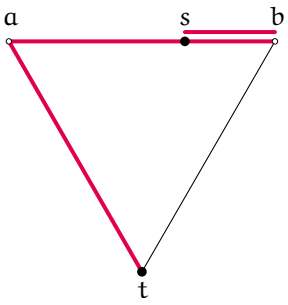
neighbourhood of w, the lower bound holds for any deterministic k-local 0-memory routing algorithm. □

ROUTING ALGORITHM. The only difference with the routing algorithm we used for positive routing lies in the initial case. Since our destination is in a negative cone, we start in one of the negative cases. This time, besides cases B and C, where both or one of X_1 and X_2 are empty, we also need case D, where neither is empty. Recall that in the previous section, we showed that a routing step starting in case A, B, or C can never result in case D. Thus, if the routing process starts in case D, it never returns there once it enters case A, B, or C.

In case D, the routing algorithm first tries to follow an edge (s, v) with $v \in X_0$. If several such edges exist, an arbitrary one of these is followed. If no such edge exists, the routing algorithm follows the single edge (s, v) with v in the smaller of X_1 and X_2. In short, the routing algorithm favours moving towards the closest corner of $\triangle_{ts}$ when it is not able to move towards t. Note that, in the instances of Figure 4.14, this choice ensures that the first routing step incurs the smallest loss in the worst case, making it possible to meet the lower bound of Lemma 4.9. We now show that our algorithm achieves this lower bound in all cases.

UPPER BOUND. The potential in case D is given below. It mirrors the lower bound path, in that it allows walking towards the closest corner, crossing the triangle, then walking down to t. This is the highest potential among the four cases.

$$\text{Case D:} \quad \phi = |ta| + |ab| + \min(|as|, |sb|)$$



Case D

Figure 4.15: The potential ϕ in case D.

As before, we want to show that for any routing step, the reduction in ϕ is at least as large as the length of the edge followed. Since we already did this for cases A, B, and C, and none of them can lead to case D, all that is left is to prove it for case D.

CASE D. A routing step starting in case D cannot lead to case A, as the step stays within $\triangle_{ts}$, but it may lead to case B, C, or D. There are two situations, depending on whether edges (s, v) with $v \in X_0$ exist or not. These are illustrated in Figure 4.16.

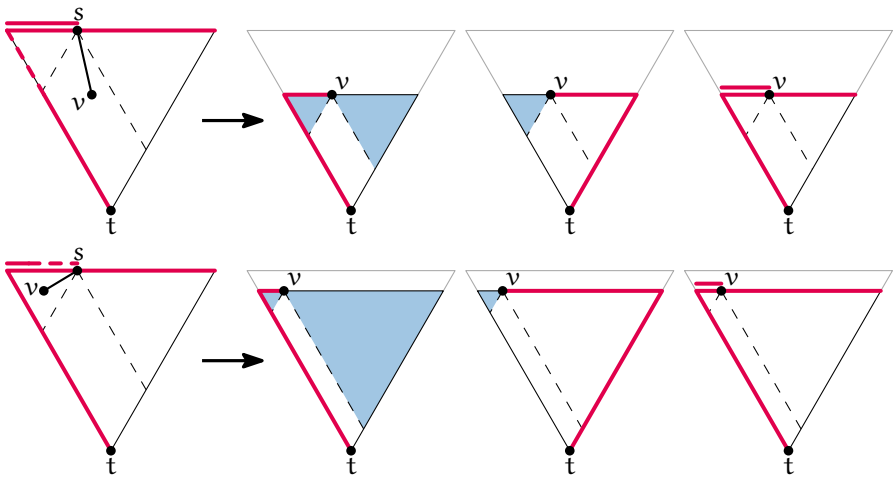


Figure 4.16: Routing in case D. The endpoint v of the edge followed lies in X_0 (Top), or the smaller of X_1 and X_2 (Bottom).

In the first situation, where we follow an edge (s, v) with $v \in X_0$, the reduction of the vertical part of ϕ is at least as large as $|sv|$ by Observation 4.7. The horizontal part of ϕ can only decrease, as $\triangle_{tv}$ is fully contained in $\triangle_{ts}$ and v lies in X_0. In the second situation, where the endpoint of our edge lies in the smaller of X_1 and X_2, these roles switch, with the reduction of the horizontal part of ϕ being at least as large as $|sv|$ and the vertical part of ϕ only decreasing. In both situations, the statement is proven.

LEMMA 4.10 (Upper bound for negative routing). *Let u and w be two vertices, with w in a positive cone of u. Let m be the midpoint of the side of $\triangle_{uw}$ opposing u, and let α be the unsigned angle between the lines uw and um. There is a deterministic 1-local 0-memory routing algorithm on the half-Θ_6-graph for which every path followed has length at most $(5/\sqrt{3} \cdot \cos\alpha - \sin\alpha) \cdot |uw|$ when routing from w to u.*

Proof. Since the choices that the routing algorithm makes are completely determined by the neighbours of s and the location of s and t, the algorithm is indeed deterministic, 1-local, and 0-memory. To bound the length of the resulting path, we again showed that for any routing step, the reduction in ϕ is at least as large as the length of the edge followed. As in the proof of Lemma 4.8, this implies that the routing algorithm terminates and that the total length of the path followed is bounded by the initial value of ϕ. Since our destination lies in a negative cone, we start in one of the cases B, C, or D. Of these three cases, case D has the largest initial potential of $|ta| + |ab| + \min(|as|, |sb|)$. Taking the side of $\triangle_{uw}$ as the unit of

length reduces this to $1 + 1 + 1/2 - |wm| = 5/2 - |wm|$, and using the same analysis as in Lemma 4.9, we obtain the desired bound of $(5/\sqrt{3} \cdot \cos\alpha - \sin\alpha) \cdot |uw|$. □

As Theorem 4.3 follows from Lemmas 4.6, 4.8, 4.9, and 4.10, this concludes our proof.

4.6 A STATEFUL ALGORITHM

Next we present a slightly different routing algorithm from the one in the previous section. The main difference between the two algorithms is that this one maintains one piece of information as state, making it $O(1)$-memory instead of 0-memory. The information that is stored is a *preferred side*, and it is either nil, X_1, or X_2. Intuitively, the new algorithm follows the original algorithm until it is routing negatively and determines that either X_1 or X_2 is empty. At that point, the algorithm sets the empty side as the preferred side and picks the rest of the edges in such a way that the preferred side remains empty. Thus, the algorithm maintains as invariant that if the preferred side is set (not nil), that region is empty. Furthermore, once the preferred side is set, it stays fixed until the algorithm reaches the destination. This algorithm simplifies the cases a little, but more importantly, it allows the algorithm to check far fewer edges while routing. This is crucial, as the new algorithm forms the basis for routing algorithms on versions of the half-Θ_6-graph with some edges removed to bound the maximum degree, described in the next section.

We now present the details of this stateful version of the routing algorithm. Recall that we are trying to find a path from a current vertex s to a destination vertex t. For ease of description, we again assume without loss of generality that t lies in C_0 or $\overline{C}_0$ of s. If t lies in $\overline{C}_0$, the cones around s split $\triangle_{ts}$ into three regions X_0, X_1, and X_2, as in Figure 4.9. For brevity, we use "an edge in X_0" to denote an edge incident to s with the other endpoint in X_0. The cases are as follows:

- If t lies in a positive cone of s, we are in case $\mathcal{A}$.
- If t lies in a negative cone of s and no preferred side has been set yet, we are in case $\mathcal{B}$.
- If t lies in a negative cone of s and a preferred side has been set, we are in case $\mathcal{C}$.

These cases are closely related to the cases in the stateless algorithm. Cases $\mathcal{A}$ and $\mathcal{B}$ correspond to cases A and D, respectively, while case $\mathcal{C}$ merges cases B and C from the original algorithm into a single case, where only one side's emptiness is tracked. This is reflected in the routing strategy for each case:

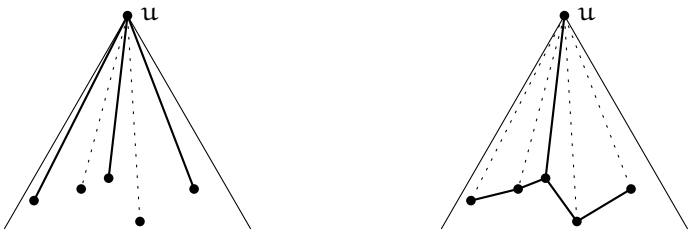
- In case $\mathcal{A}$, follow the unique edge (s, v) in the positive cone containing t. If t lies in a negative cone of v, set the preferred side to the region (X_1 or X_2 of v) that is contained in $\triangle_{sv}$, as this is now known to be empty (see Figure 4.11b).
- In case $\mathcal{B}$, if there are edges in X_0, follow an arbitrary one. Otherwise, if there is an edge in the smaller of X_1 and X_2, follow that edge. Otherwise, follow the edge in the larger of X_1 and X_2 and set the other as the preferred side. By Theorem 4.1, at least one of these edges must exist.
- In case $\mathcal{C}$, if there are edges in X_0, follow the one closest to the preferred side in cyclic order around s. Otherwise, follow the edge in the positive cone that is not on the preferred side. Again, at least one of these edges must exist.

The proof in Section 4.5 can be adapted to show that this routing algorithm achieves the same upper bounds. In short, the proof is simplified to only use a potential as defined for cases A, C, and D, and only a subset of the illustrations in Figures 4.11, 4.13, and 4.16 are relevant. We omit the repetitive details.

4.7 BOUNDING THE MAXIMUM DEGREE

Each vertex in the half-Θ_6-graph has at most one incident edge in each positive cone, but it can have an unbounded number of incident edges in its negative cones. In this section, we describe two transformations that allow us to bound the total degree of each vertex. The transformations are adapted from Bonichon et al. [5].

The first transformation discards all edges in each negative cone, except for three: the first and last edges in clockwise order around the vertex and the edge to the "closest" vertex, meaning the vertex whose projection on the bisector of the cone is closest (see Figure 4.17a). This results in a subgraph with maximum degree 12, which we call G_{12}.



(a) (b)

Figure 4.17: The construction for G_{12} (a) and G_9 (b). Solid edges are kept, while dotted edges are discarded if no other vertex wants to keep them.

To reduce the degree even further, we note that since the half-Θ_6-graph is internally triangulated, consecutive neighbours of u within

a negative cone are connected by edges. We call the path formed by these edges the *canonical path*. Instead of keeping three edges per negative cone, we now keep only the edge to the closest vertex, but force the edges of the canonical path to be kept as well (see Figure 4.17b). We call the resulting graph G_9. Bonichon et al. [5] showed that all edges on the canonical path are either first or last in a negative cone, making G_9 a subgraph of G_{12}. Note that since the half-Θ_6-graph is planar, both subgraphs are planar as well. They also proved that G_9 is a 3-spanner of the half-Θ_6-graph with maximum degree 9. Since the half-Θ_6-graph is a 2-spanner and G_9 is a subgraph of G_{12}, this shows that both G_9 and G_{12} are 6-spanners of the complete Euclidean graph. We give an adapted version of the proof of the spanning ratio of G_9 below.

THEOREM 4.11. *G_9 is a 3-spanner of the half-Θ_6-graph.*

Proof. Consider an edge (s, v) in the half-Θ_6-graph and assume, without loss of generality, that v lies in a negative cone of s (if not, we can swap the roles of s and v). Now consider the path between them in G_9 consisting of the edge from s to the vertex closest to s, followed by the edges on the canonical path between the closest vertex and v. We will refer to this path as the *approximation path*, and we show that it has length at most $3 \cdot |sv|$.

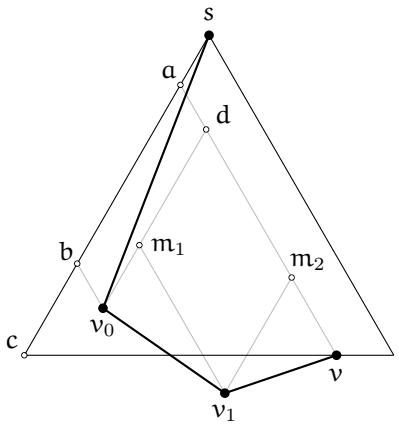


Figure 4.18: The approximation path.

Let v_0 be the closest vertex and let $v_1, \ldots, v_k = v$ be the other vertices on the approximation path. We assume without loss of generality that s lies in C_0 of v and that v lies to the right of v_0. We shoot rays parallel to the boundaries of C_0 from each vertex on the approximation path. Let m_i be the intersection of the right ray of v_{i-1} and the left ray of v_i (see Figure 4.18). These intersections must exist, as s is the closest vertex in $C_0^{v_i}$, for each v_i. Let a and b be the intersections of the left boundary of $\overline{C}_0^s$ with the left rays of v and v_0, respectively, and let c be the intersection of this left boundary with the horizontal line through v. Finally, let d be the intersection of the right ray of v_0

and the left ray of v. We can bound the length of the approximation path as follows:

$$\begin{aligned}
&|sv_0| + \sum_{i=1}^{k} |v_{i-1}v_i| \\
&\leqslant |sb| + |bv_0| + \sum_{i=1}^{k} |v_{i-1}m_i| + \sum_{i=1}^{k} |m_iv_i| \\
&= |sb| + |bv_0| + |ab| + |dv| \quad \{\text{by projection}\} \\
&= |sb| + |ab| + |av| \\
&\leqslant |sc| + 2 \cdot |cv|.
\end{aligned}$$

The last inequality follows from the fact that v_0 is the closest vertex to s. Let α be $\angle csv$. Some basic trigonometry gives us that $|sc| = \frac{2}{\sqrt{3}} \cdot \sin\left(\alpha + \frac{\pi}{3}\right) \cdot |sv|$ and $|cv| = \frac{2}{\sqrt{3}} \cdot \sin\alpha \cdot |sv|$. Thus the approximation path is at most $\frac{2}{\sqrt{3}} \cdot \left(\sin\left(\alpha + \frac{\pi}{3}\right) + 2 \cdot \sin\alpha\right)$ times as long as (s, v). Since this function is increasing in $[0, \frac{\pi}{3}]$, the maximum is achieved for $\alpha = \pi/3$, where it is 3. Therefore every edge of the half-Θ_6-graph can be approximated by a path that is at most 3 times as long and the theorem follows. □

Note that the part of the approximation path that lies on the canonical path has length at most $2 \cdot |cv| = \frac{4}{\sqrt{3}} \cdot \sin\alpha \cdot |sv|$. This function is also increasing in $[0, \frac{\pi}{3}]$ and its maximal value is 2, so the total length of this part is at most $2 \cdot |sv|$.

4.7.1 *Routing in* G_{12}

The stateful algorithm in Section 4.6 constructs a path between two vertices in the half-Θ_6-graph. We cannot directly follow this path in G_{12}, as some of the edges may have been removed. Hence, we need to find a new path in G_{12} that approximates the path in the half-Θ_6-graph, taking the missing edges into account. This often amounts to following the approximation path for edges that are in the path in the half-Θ_6-graph, but were removed to create G_{12}. In addition, some of the information the algorithm uses to decide which edge to follow relies on the presence or absence of edges in the half-Θ_6-graph. Since the absence of these edges in G_{12} does not tell us whether or not they were present in the half-Θ_6-graph, we need to find a new way to make these decisions.

First, note that the only information we need to determine in which of the three cases we are, are the coordinates of s and t and whether the preferred side has been set or not. Therefore we can still make this distinction in G_{12}. The following five headlines refer to steps of the stateful algorithm on the half-Θ_6-graph, and the text after a headline

describes how to simulate that step in G_{12}. We discuss modifications for G_9 in Section 4.7.2.

FOLLOW AN EDGE (s, v) IN A POSITIVE CONE C. If the edge of the half-Θ_6-graph is still present in G_{12}, we simply follow it. If it is not, the edge was removed because s is on the canonical path of v and it is not the closest, first or last vertex on the path. Since G_{12} is a supergraph of G_9, we know that all of the edges of the canonical path are kept and every vertex on the path originally had an edge to v in C. Therefore it suffices to traverse the canonical path in one direction until we reach a vertex with an edge in C, and follow this edge. Since the edges connecting v to the first and last vertices on the path are always kept, the edge we find in this way must lead to v. Note that the edges of the canonical path are easy to identify, as they are the closest edges to C in cyclic order around s (one on either side of C).

This method is guaranteed to reach v, but we want to find a *competitive* path to v. Therefore we use exponential search along the canonical path: we start by following the shorter of the two edges of the canonical path incident to s. If the endpoint of this edge does not have an edge in C, we return to s and travel twice the length of the first edge in the other direction. We keep returning to s and doubling the maximum travel distance until we find a vertex x that does have an edge in C. If x is not the closest to v, by the triangle inequality, following its edge to v is shorter than continuing our search until we reach the closest and following its edge. So for the purpose of bounding the distance travelled, we can assume that x is closest to v. Let d be the distance between s and x along the canonical path. By using exponential search to find x, we travel at most 9 times this distance [2] and afterwards we follow (x, v). From the proof of Theorem 4.11, we know that $d \leqslant 2 \cdot |sv|$ and $d + |xv| \leqslant 3 \cdot |sv|$. Thus the total length of our path is at most $9 \cdot d + |xv| = 8 \cdot d + (d + |xv|) \leqslant 16 \cdot |sv| + 3 \cdot |sv| = 19 \cdot |sv|$.

DETERMINE IF THERE ARE EDGES IN X_0. In the regular half-Θ_6-graph we can look at all our neighbours and see if any of them lie in X_0. However, in G_{12}, these edges may have been removed. Fortunately, we can still determine if they existed in the original half-Θ_6-graph. To do this, we look at the vertices of the canonical path in this cone that are first and last in clockwise order around s. If these vertices do not exist, s did not have any incoming edges in this cone, so there can be no edges in X_0. If the first and last are the same vertex, this was the only incoming edge to s from this cone, so we simply check if its endpoint lies in X_0. The interesting case is when the first and last exist and are distinct. If either of them lies in X_0, we have our answer, so assume that both lie outside of X_0. Since they were con-

nected to s, they cannot have t in their positive cone, so they must lie in one of two regions, which we call S_1 and S_2 (see Figure 4.19).

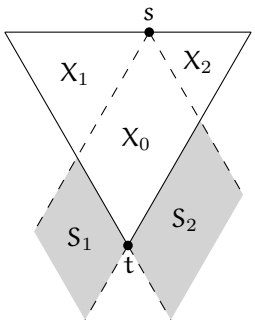


Figure 4.19: Possible regions for the first and last vertex.

If both the first and last lie in S_2, there can be no edge in X_0, since any vertex of the canonical path in X_0 either lies in cone C_0 of the last vertex, or would come after the last vertex in clockwise order around s. Both yield a contradiction. If both lie in S_1, a similar argument using the first vertex applies.

On the other hand, if the first lies in S_2 and the last in S_1, both X_1 and X_2 have to be empty, since both vertices are connected to s. Now we are in one of two cases: either X_0 is also empty, or it is not. If there are no vertices in X_0 (different from t and s), t must have had an edge to s. On the other hand, if there are other vertices in X_0, the topmost of these vertices must have had an edge to s. In either case, there must have been an edge in X_0. This shows that we can check whether there was an edge in X_0 in the half-Θ_6-graph using only the coordinates of the first and last vertex.

FOLLOW AN ARBITRARY EDGE IN X_0. If the half-Θ_6-graph has edges in X_0, we simulate following an arbitrary one of these by first following the edge to the closest vertex in the negative cone. If this vertex is in X_0, we are done. Otherwise, we follow the canonical path in the direction of X_0 and stop once we are inside. This traverses exactly the approximation path of the edge, and hence travels a distance of at most 3 times the length of the edge.

DETERMINE IF THERE IS AN EDGE IN X_1 OR X_2. Since these regions are symmetric, we will consider only the case for X_1. Since X_1 is contained in a positive cone of s, it contains at most one edge incident to s. If the edge is present in G_{12}, we can simply test whether the other endpoint lies in X_1. However, if s does not have a neighbour in this cone (see Figure 4.20), we need to find out whether it used to have one in the original half-Θ_6-graph and if so, whether it was in X_1. Since this step is only needed in case $\mathcal{B}$ after we determine that there are no edges in X_0, we can use this information to guide our search. Specifically, we know that if we find an edge, we should follow it.

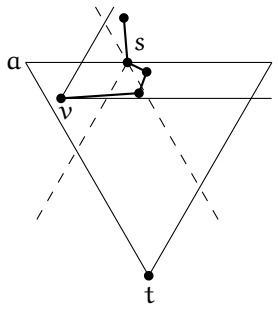


Figure 4.20: A vertex v in X_1.

Therefore we simply attempt to follow the edge in this cone, using the exponential search method for following an edge in a positive cone described earlier. Let x be the first vertex we encounter that still has an edge (x, w) in C_1. If in the half-Θ_6-graph, s had an edge (s, v) in X_1, then we know (from the arguments presented earlier for following an edge in a positive cone) that w is v. As such, w must lie in X_1. We also know (from the proof of Theorem 4.11) that the distance along the canonical path from s to x is at most $2 \cdot |sv|$, which is bounded by $2 \cdot |as|$ since v lies in X_1. In this case, we follow the edge from x to v. Conversely, if we do not find any vertex with an edge in C_1 within a distance of $2 \cdot |as|$ from s, or we do, but the endpoint w of the edge does not lie in X_1, then we can return to s and conclude that it did not have an edge in X_1 in the half-Θ_6-graph and therefore X_1 must be empty.

If there was an edge in X_1, we travelled the same distance as if we were simply following the edge: at most $19 \cdot |sv|$. If we return to s unsuccessfully, we travelled at most $20 \cdot |as|$: 9 times $2 \cdot |as|$ during the exponential search and $2 \cdot |as|$ to return to s.

FOLLOW THE EDGE IN X_0 CLOSEST TO THE PREFERRED SIDE IN CLOCKWISE ORDER. To follow this edge, we first follow the edge to the closest vertex. If this lands us in X_0, we then follow the canonical path towards the preferred side and stop at the last vertex on the canonical path that is in X_0. If the closest is not in X_0, we follow the canonical path towards X_0 and stop at the first or last vertex in X_0, depending on which side of X_0 we started on. This follows the approximation path of the edge, so the distance travelled is at most 3 times the length of the edge.

ROUTING RATIO. This shows that we can simulate the stateful routing algorithm on G_{12}. As state in the message, we need to store not only the preferred side, but also information for the exponential search, including distance travelled. The exact routing ratios are as follows.

THEOREM 4.12. *Let u and w be two vertices, with w in a positive cone of u. There exists a deterministic 1-local $O(1)$-memory routing algorithm on G_{12} with routing ratio*

i) $19 \cdot 2 = 38$ *when routing from u to w,*

ii) $19 \cdot 5/\sqrt{3} \approx 54.849$ *when routing from w to u.*

Proof. As shown above, we can simulate every edge followed by the algorithm by travelling at most 19 times the length of the edge. The only additional cost is incurred in case $\mathcal{B}$, when we try to follow an edge in the smaller of X_1 and X_2, but this edge does not exist. In this case, we travel an additional $20 \cdot |as|$, where a is the corner closest to s. Fortunately, this can happen at most once during the execution of the algorithm, as it prompts the transition to case $\mathcal{C}$, after which the algorithm never returns to case $\mathcal{B}$. Looking at the proof for the upper bound in Section 4.5 (specifically, the second case in Figure 4.16b), we observe that in the transition from case D to C, there is $2 \cdot |as|$ of unused potential. Since we are trying to show a routing ratio of 19 times the original, we can charge the additional $20 \cdot |as|$ to the $38 \cdot |as|$ of unused potential. □

4.7.2 *Routing in G_9*

In this subsection, we explain how to modify the previously described simulation strategies so that they work for G_9, where the first and last edges are not guaranteed to be present. We discuss only those steps that rely on the presence of these edges. To route successfully in this setting, we need to change our model slightly. We now let every vertex store a constant amount of information in addition to the information about its neighbours.

FOLLOW AN EDGE (s, v) IN A POSITIVE CONE. Because the first and last edges are not always kept, we cannot guarantee that the first vertex we reach with an edge in this positive cone is still part of the same canonical path. This means that the edge could connect to some arbitrary vertex, far away from v. Therefore our original exponential search solution does not work. Instead, we store one bit of information at s (per positive cone), namely in which direction we have to follow the canonical path to reach the closest vertex to v. Knowing this, we just follow the canonical path in the indicated direction until we reach a vertex with an edge in this positive cone. This vertex must be the closest, so it gives us precisely the approximation path and therefore we travel at most $3 \cdot |sv|$.

DETERMINE IF THERE ARE EDGES IN X_0. In G_{12}, this test was based on the coordinates of the endpoints of the first and last edge. Since these might be missing in G_9, we store the coordinates of these

vertices at s. This allows us to perform the check without increasing the distance travelled.

DETERMINE IF THERE IS AN EDGE IN X_1 OR X_2. As in the positive routing simulation, we now know where to go to find the closest. Therefore we simply follow the canonical path in this direction from s and stop when we reach a vertex with an edge in the correct positive cone, or when we have travelled $2 \cdot |as|$. If there is an edge, we follow exactly the approximation path, giving us 3 times the length of the edge. If there is no edge, we travel $2 \cdot |as|$ back and forth, for a total of $4 \cdot |as|$.

ROUTING RATIO. Since the other simulation strategies do not rely on the presence of the first or last edges, we can now analyze the routing ratio obtained on G_9.

THEOREM 4.13. *Let u and w be two vertices, with w in a positive cone of u. By storing $O(1)$ additional information at each vertex, there exists a deterministic 1-local $O(1)$-memory routing algorithm on G_9 and G_{12} with routing ratio*

i) $3 \cdot 2 = 6$ *when routing from u to w,*

ii) $3 \cdot 5/\sqrt{3} \approx 8.661$ *when routing from w to u.*

Proof. The simulation strategy for G_{12} followed the approximation path for each edge, except when following an edge in a positive cone. Since our new strategy follows the approximation path there as well, our new routing ratio is only 3 times the one for the half-Θ_6-graph. Note that this is still sufficient to charge the additional $4 \cdot |sa|$ travelled to the transition from case $\mathcal{B}$ to $\mathcal{C}$, which has $3 \cdot 2 \cdot |as|$ of otherwise unused potential. Since G_9 is a subgraph of G_{12}, this strategy works on G_{12} as well. □

4.8 CONCLUSIONS

We presented a competitive deterministic 1-local 0-memory routing algorithm on the half-Θ_6-graph. We also presented matching lower bounds on the routing ratio for any deterministic k-local 0-memory algorithm, showing that our algorithm is optimal. Since any triangulation can be embedded as a half-Θ_6-graph using Schnyder's embedding [24], this shows that any triangulation has an embedding that admits a competitive routing algorithm. An interesting open problem here is whether this approach can be extended to other theta-graphs. In particular, we recently extended the proof for the spanning ratio of the half-Θ_6-graph to theta-graphs with $4k + 2$ cones, for integer $k > 0$ [7]. It would be interesting to see if it is possible to find optimal routing algorithms for these graphs as well.

We further extended our routing algorithm to work on versions of the half-Θ_6-graph with bounded maximum degree. As far as we know, these are the first competitive routing algorithms on bounded-degree plane graphs. There are several problems here that are still open. For example, while we found a matching lower bound for negative routing in the regular half-Θ_6-graph, we do not have one for the version with bounded degree. Can we find this, or is it possible to improve the routing algorithm further?

Bonichon et al. [5] also introduced a version of the half-Θ_6-graph with maximum degree 6. This graph differs from G_{12} and G_9 in that it is not a subgraph of the half-Θ_6-graph: to maintain the spanning ratio while removing even more edges, they add certain shortcut edges that were not part of the original half-Θ_6-graph. It would be interesting to see if our routing algorithms could be extended to work on this graph. This would most likely require locally detecting shortcut edges, and finding a way to route 'around' the newly removed edges.

BIBLIOGRAPHY

[1] Patrizio Angelini, Fabrizio Frati, and Luca Grilli. An algorithm to construct greedy drawings of triangulations. *Journal of Graph Algorithms and Applications*, 14(1):19–51, 2010.

[2] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.

[3] Luis Barba, Prosenjit Bose, Mirela Damian, Rolf Fagerberg, Wah Loon Keng, Joseph O'Rourke, André van Renssen, Perouz Taslakian, Sander Verdonschot, and Ge Xia. New and improved spanning ratios for Yao graphs. *Journal of Computational Geometry*, 6(2):19–53, 2015. Special issue for SoCG 2014.

[4] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between theta-graphs, Delaunay triangulations, and orthogonal surfaces. In *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010)*, pages 266–278, 2010.

[5] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. Plane spanners of maximum degree six. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP 2010 (1))*, pages 19–30, 2010.

[6] Prosenjit Bose, Andrej Brodnik, Svante Carlsson, Erik D. Demaine, Rudolf Fleischer, Alejandro López-Ortiz, Pat Morin, and J. Ian Munro. Online routing in convex subdivisions. *International Journal of Computational Geometry & Applications*, 12(4):283–295, 2002.

[7] Prosenjit Bose, Jean-Lou De Carufel, Pat Morin, André van Renssen, and Sander Verdonschot. Optimal bounds on Theta-graphs: More is not always better. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012)*, pages 305–310, 2012.

[8] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SIAM Journal on Computing*. Forthcoming.

[9] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive routing in the half-θ_6-graph. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1319–1328, 2012.

[10] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive routing on a bounded-degree plane spanner. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012)*, pages 299–304, 2012.

[11] Prosenjit Bose and Pat Morin. Online routing in triangulations. *SIAM Journal on Computing*, 33(4):937–951, 2004.

[12] Prosenjit Bose, André van Renssen, and Sander Verdonschot. On the spanning ratio of Theta-graphs. In *Proceedings of the 13th Algorithms and Data Structures Symposium (WADS 2013)*, pages 182–194, 2013.

[13] L. Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[14] Raghavan Dhandapani. Greedy drawings of triangulations. *Discrete & Computational Geometry*, 43(2):375–392, 2010.

[15] Michael B. Dillencourt. Realizability of Delaunay triangulations. *Information Processing Letters*, 33(6):283–287, 1990.

[16] Michael T. Goodrich and Darren Strash. Succinct greedy geometric routing in the Euclidean plane. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, pages 781–791, 2009.

[17] Xin He and Huaming Zhang. Schnyder greedy routing algorithm. In *Proceedings of the 7th annual conference on Theory and Applications of Models of Computation (TAMC 2010)*, pages 271–283, 2010.

[18] Xin He and Huaming Zhang. On succinct convex greedy drawing of 3-connected plane graphs. In *Proceedings of the 22nd annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1477–1486, 2011.

[19] J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7(1):13–28, 1992.

[20] Tom Leighton and Ankur Moitra. Some results on greedy embeddings in metric spaces. *Discrete & Computational Geometry*, 44(3):686–705, 2010.

[21] Sudip Misra, Isaac Woungang, and Subhas Chandra Misra. *Guide to wireless sensor networks*. Springer, 2009.

[22] Christos H. Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. *Theoretical Computer Science*, 344(1):3–14, 2005.

[23] Harald Räcke. Survey on oblivious routing strategies. In *Proceedings of the 5th Conference on Computability in Europe (CiE 2009)*, pages 419–429, 2009.

[24] Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the 1st annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1990)*, pages 138–148, 1990.

Part II

FLIPS IN TRIANGULATIONS

5

A HISTORY OF FLIPS IN COMBINATORIAL TRIANGULATIONS

Given two combinatorial triangulations, how many edge flips are necessary and sufficient to convert one into the other? This question has occupied researchers for over 75 years. We provide a comprehensive survey, including full proofs, of previous attempts to answer it, before presenting our own contribution in Chapter 6.

This chapter was first published as an invited chapter in the proceedings of the XIV Spanish Meeting on Computational Geometry (EGC 2011) [3], and contains joint work with Prosenjit Bose.

5.1 INTRODUCTION

A *triangulation* is a simple planar graph that is *maximal*, which means that adding any other edge would make the graph non-planar. This implies that every face is a triangle (a cycle of length 3). In any triangulation, an edge $e = (a, b)$ is adjacent to two faces: abc and abd. An *edge flip* consists of deleting the edge e from the triangulation and adding the other diagonal of the resulting quadrilateral (in this case (c, d)) to the graph, so that it remains a triangulation. Figure 5.1 shows an example of an edge flip. An edge e is not flippable if (c, d) is already an edge of the triangulation. If the vertices have fixed coordinates in the plane and edges are drawn as straight-line segments between their endpoints, the restriction that the new edge may not introduce any crossings is usually added. This is commonly referred to as the *geometric* setting. However, we focus on the problem in the *combinatorial* setting, where we are only given a combinatorial embedding of the graph (the clockwise order of edges around each vertex). Even in this setting, not all edges in a triangulation are flippable. Gao et al. [5] showed that in every n-vertex triangulation at least $n - 2$ edges are always flippable and that there exist some triangulations where at most $n - 2$ edges are flippable. If the triangulation has

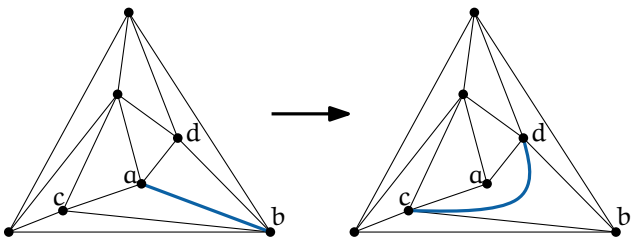


Figure 5.1: An example triangulation before and after flipping edge (a, b).

minimum degree at least 4, they showed that there are at least $2n + 3$ flippable edges and the bound is tight in certain cases.

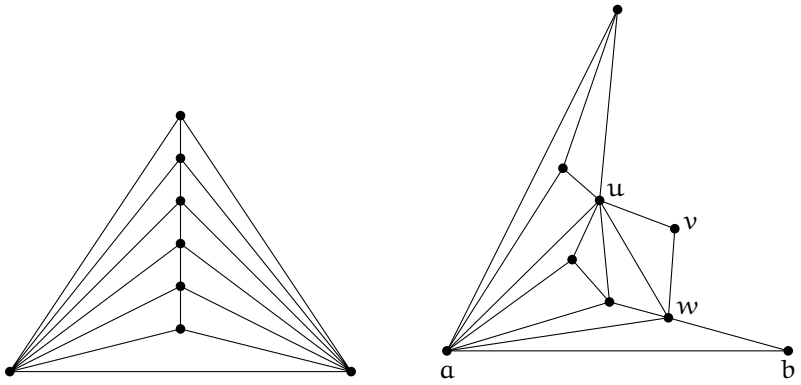
Note that by flipping an edge e, we transform one triangulation into another. This gives rise to the following question: Can any n-vertex triangulation be transformed into any other n-vertex triangulation through a finite sequence of flips? This question was first addressed by Wagner [11] in 1936, who answered it in the affirmative. Although it is well known that the number of n-vertex triangulations is exponential in n, Wagner's inductive proof gives rise to an algorithm that can achieve this transformation using at most $2n^2$ edge flips. The key element of Wagner's proof is that he circumvents the issue of graph isomorphism by showing how to convert any given triangulation into a fixed *canonical* triangulation that can be easily recognized. The downside of this approach is that one may use many more flips than necessary to convert one triangulation into another. In fact, it is possible that two triangulations are one edge flip away from each other, but Wagner's approach uses a quadratic number of flips to convert one into the other.

The notion of two triangulations being "close" to each other in terms of number of flips can be expressed through a *flip graph*. The *flip graph* has a vertex for each distinct n-vertex triangulation and an edge between two vertices if their corresponding triangulations differ by a single flip. Two triangulations are considered distinct if they are not isomorphic. Questions about the flip operation can be viewed as questions on the flip graph. Asking whether any n-vertex triangulation can be converted into any other via flips is asking whether the flip graph is connected. Asking for the smallest number of flips required to convert one triangulation into another is asking for the shortest path in the flip graph between the two vertices representing the given triangulations. The maximum, minimum and average degree in the flip graph almost correspond to the maximum, minimum and average number of flippable edges, with the caveat that different edges might result in isomorphic triangulations when flipped. One can also ask what the chromatic number of the flip graph is, whether it is Hamiltonian, etc. Many of these questions have been addressed in the literature. The survey by Bose and Hurtado gives a good overview of the field [2]. In this chapter, we focus mainly on attempts to determine the diameter of the flip graph. In other words, how many edge flips are sufficient and sometimes necessary to transform a given triangulation into any other? Sections 5.2, 5.3, and 5.4 detail the techniques used to provide upper bounds for this question, while Section 5.5 presents a lower bound. Our own contributions to this question are presented in the next chapter.

5.2 WAGNER'S BOUND

In 1936, Wagner [11] first addressed the problem of determining whether one can convert a given triangulation into another via edge flips. Although his paper is entitled "Remarks on the four-colour problem", it contains a proof that every planar graph has a straight-line embedding, defines the edge flip operation (or diagonal transformation, as Wagner calls it) and shows that any two triangulations can be transformed into each other by a finite series of edge flips before finishing with a result on the number of valid colourings of a graph.

To prove that any pair of triangulations can be transformed into each other via flips, Wagner first introduces the *canonical triangulation*, which is the unique triangulation with two dominant vertices (see Figure 5.2a). We will denote the canonical triangulation on n vertices by $\triangle_n$.



(a) (b)

Figure 5.2: (a) The canonical triangulation on 8 vertices. (b) A face uwv such that u and w are neighbours of a, while v is not. Flipping the edge (u, w) brings us closer to the canonical triangulation.

LEMMA 5.1 (Wagner [11], Theorem 4). *Any triangulation on n vertices can be transformed into $\triangle_n$ by a sequence of at most $n^2 - 7n + 12$ flips.*

Proof. To transform a given triangulation into the canonical one, we fix an outer face and pick two of its vertices, say a and b, to become the dominant vertices in the canonical triangulation. If a is not adjacent to all other vertices, there exists a face uwv such that u and w are neighbours of a, while v is not. This situation is illustrated in Figure 5.2b. We flip the edge (u, w).

In his original proof, Wagner argues that this gives a finite sequence of flips that increases the degree of a by one. He simply states that this sequence is finite and does not argue why (u, w) is flippable in the first place. We provide these additional arguments below.

We consider two cases:

- auw is a face. In this case, the flip will result in the edge (a, v), increasing the degree of a by one. This flip is valid, as v was not adjacent to a before the flip.
- auw is not a face. In this case the flip is also valid, since auw forms a triangle that separates v from the vertices inside. The flip does not increase the degree of a, but it does increase the degree of v and since the number of vertices is finite, the degree of v cannot increase indefinitely. Therefore, we must eventually arrive to the first case, where we increase the degree of a by one.

Since the same strategy can be used to increase the degree of b as long as it is not dominant, this gives us a sequence of flips that transforms any triangulation into the canonical one. Every vertex of a triangulation has degree at least 3, so the degree of a and b needs to increase by at most $n-4$. Since we might need to increase the degree of v from 2 until it is adjacent to all but one of the neighbours of a or b, the total flip sequence has length at most

$$2\sum_{i=3}^{n-2}(i-2) \quad = \quad n^2-7n+12. \qquad \square$$

By using the canonical triangulation as an intermediate form, the main result follows.

THEOREM 5.2 (Wagner [11], Theorem 4). *Any pair of triangulations T_1 and T_2 on n vertices can be transformed into each other by a sequence of at most $2n^2-14n+24$ flips.*

Proof. By Lemma 5.1, we have two sequences of flips, S_1 and S_2, that transform T_1 and T_2 into the canonical triangulation, respectively. Since a flip can be reversed, we can use S_1, followed by the reverse of S_2 to transform T_1 into T_2. Since both S_1 and S_2 have length at most $n^2-7n+12$, the total sequence uses at most $2n^2-14n+24$ flips. $\square$

A simpler and more precise proof that also gives a quadratic upper bound was given by Negami and Nakamoto [9].

LEMMA 5.3 (Negami and Nakamoto [9], Theorem 1). *Any triangulation on n vertices can be transformed into $\triangle_n$ by a sequence of $O(n^2)$ flips.*

Proof. Let abc be the outer face. Suppose we wish to make both a and b dominant. Instead of showing that a sequence of flips can always increase the degree of a or b, we will show that it is always possible to find one flip that decreases the degree of c. Once c has degree 3, the

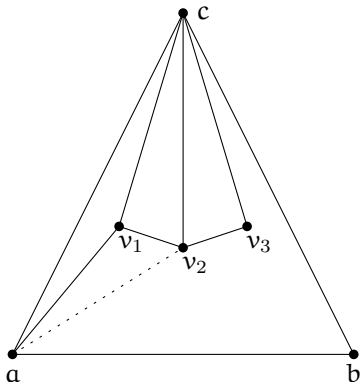


Figure 5.3: The exterior triangle abc with the first three neighbours of c in counter-clockwise order. Depending on the presence of edge (a, v_2), either (c, v_1) or (c, v_2) is flipped.

same argument can be used to find a flip that decreases the degree of c's neighbour inside the triangle until it has degree 4, and so on.

To determine which edge to flip, let $a, v_1, v_2, \ldots, b$ be the neighbours of c in counter-clockwise order. This situation is illustrated in Figure 5.3. If a and v_2 are not adjacent, we can flip (c, v_1) into (a, v_2), reducing the degree of c. If a and v_2 are adjacent, av_2c forms a cycle that separates v_1 and v_3, so we can flip (c, v_2) to reduce c's degree. We continue this until c has degree 3, at which point we apply the same argument to reduce the degree of c's remaining neighbour inside the triangle until it has degree 4. Then we continue with the neighbour of v_1 inside the triangle av_1b, and so on, until all vertices except for a and b have degree 3 or 4, at which point we have obtained the canonical triangulation. □

5.3 KOMURO'S BOUND

Since Wagner's result, it remained an open problem whether the diameter of the flip graph was indeed quadratic in the number of vertices. Komuro [7] showed that in fact the diameter was linear by proving a linear upper and lower bound. We present the argument for the upper bound in this section and discuss the lower bound in Section 5.5.

Komuro used Wagner's approach of converting a given triangulation into the canonical triangulation. Given an arbitrary triangulation, the key is to bound the number of flips needed to make two vertices, say a and b, dominant. If there always exists one edge flip that increases the degree of a or b by 1, then at most $2n - 8$ flips are sufficient since dominant vertices have degree $n - 1$ and all vertices in a triangulation have degree at least 3. However, this is not always the case. Figure 5.4 shows a triangulation where no single flip increases the degree of a or b. Komuro used the following function to bound

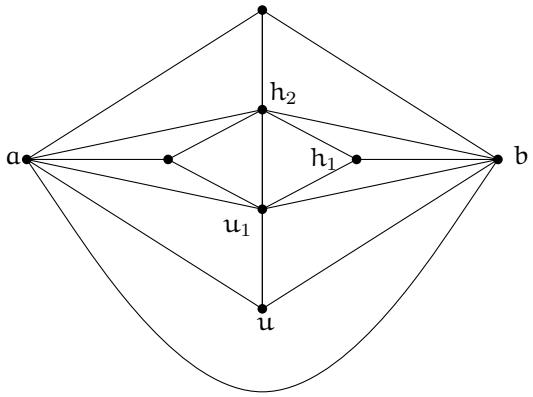


Figure 5.4: No single edge can be flipped to increase the degree of a or b.

the number of flips: $d_G(a,b) = 3\deg(a) + \deg(b)$. He showed that there always exists either one edge flip where $d_G(a,b)$ goes up by at least 1 or two edge flips where $d_G(a,b)$ goes up by at least 2. The cleverness of the function is that in some cases, two edge flips increase the degree of a by 1 but decrease the degree of b by 1. However, since the function increases by 2, it still increases by at least 1 per flip. Since $d_G(a,b) \leqslant 4n-4$, we have that $4n-4-d_G(a,b)$ is an upper bound on the number of flips required to make a and b dominant.

LEMMA 5.4 (Komuro [7], Lemma 2). *Let* G *be a triangulation on* n *vertices and let* a, b *be any pair of adjacent vertices of* G. *Then* G *can be transformed into the canonical triangulation* $\triangle_n$ *with* a *and* b *as dominant vertices with at most* $4n-4-(3\deg(a)+\deg(b))$ *edge flips.*

Proof. In a triangulation, every vertex must have degree at least 3. Let uab be a face adjacent to ab. We consider two cases: $\deg(u) = 3$ and $\deg(u) > 3$. We begin with the latter. Since $\deg(u) \geqslant 4$, let a, b, w_1, w_2 be four consecutive neighbours of u in counter-clockwise order. If b is not adjacent to w_2, then flipping edge (u, w_1) increases $\deg(b)$ by 1 and thus $d_G(a,b)$ by 1. If b is adjacent to w_2, then ubw_2 is a separating triangle (a cycle of length 3 whose removal disconnects the graph) that separates a from w_1. Therefore, flipping edge (u,b) decreases $\deg(b)$ by 1 and increases $\deg(a)$ by 1. Thus, with one flip $d_G(a,b)$ increases by 2.

Now consider the case when $\deg(u) = 3$. Let u_1 be the unique vertex adjacent to u, a, and b. We now have 3 cases to consider: $\deg(u_1) = 3$, $\deg(u_1) \geqslant 5$, or $\deg(u_1) = 4$. If $\deg(u_1) = 3$, then the graph is isomorphic to K_4, which is $\triangle_4$. If $\deg(u_1) \geqslant 5$, let a, u, b, h_1, and h_2 be five consecutive neighbours of u_1 in counter-clockwise order. If b is not adjacent to h_2, then flipping the edge (u_1, h_1) increases $\deg(b)$ by 1 and thus $d_G(a,b)$ by 1. If b is adjacent to h_2, then u_1bh_2 is a separating triangle that separates u and a from h_1 (see Figure 5.4). Therefore, flipping edges (u_1, b) and (u_1, u) decreases $\deg(b)$ by 1 and increases $\deg(a)$ by 1. Thus, with two flips $d_G(a,b)$ increases by 2.

Finally, if $\deg(u_1) = 4$, then there is unique vertex u_2 adjacent to a, u_1, and b. If $\deg(u_2) = 3$, the graph is isomorphic to $\triangle_5$. If $\deg(u_2) \geqslant 5$ we apply the same argument as when $\deg(u_1) \geqslant 5$. If $\deg(u_2) = 4$, we obtain another unique vertex u_3. This process ends with u_{n-3}, at which point a and b are dominant.

Since $d_G(a, b)$ increases by at least 1 for one flip and at least 2 for two flips, we note that the total number of flips does not exceed $d_{\triangle_n}(a, b) - d_G(a, b) = 4n - 4 - (3\deg(a) + \deg(b))$ as required. □

Using this lemma, Komuro proved the following theorem.

THEOREM 5.5 (Komuro [7], Theorem 1). *Any two triangulations with n vertices can be transformed into each other by at most $8n - 54$ edge flips if $n \geqslant 13$ and at most $8n - 48$ edge flips if $n \geqslant 7$.*

Proof. Given a triangulation G on $7 \leqslant n \leqslant 12$ vertices, one can prove by contradiction that either G is one flip from $\triangle_n$ or there exists an edge (a, b) where both vertices have degree at least 5, implying that $d_G(a, b) \geqslant 20$. This gives an upper bound of $4n - 24$ to convert G to $\triangle_n$, which gives an upper bound of $8n - 48$ to convert any triangulation to any other via the canonical triangulation. Moreover, for $n \geqslant 13$, either G is one flip from canonical or there exists an edge (a, b) where a has degree at least 6 and b has degree at least 5. This means that $d_G(a, b) \geqslant 23$. The result follows. □

5.4 MORI ET AL.'S BOUND

In 2001, Mori, Nakamoto and Ota [8] improved the bound by Komuro to $6n - 30$. They used a two-step approach by finding a short path to a strongly connected kernel, which consists of all Hamiltonian triangulations. An n-vertex triangulation is Hamiltonian if it contains a Hamiltonian cycle, i.e. a cycle of length n. The general idea of the proof is to find a fast way to make any triangulation Hamiltonian and then use the Hamiltonian cycle to decompose the graph into two outerplanar graphs. These have the following nice property.

LEMMA 5.6 (Mori et al. [8], Lemma 8 and Proposition 9). *Any vertex v in a maximal outerplanar graph on n vertices can be made dominant by $n - 1 - \deg(v)$ flips.*

Proof. If v is not dominant, there is a triangle vxy where (x, y) is not an edge of the outer face. Then we can flip (x, y) into (v, z), where z is the other vertex of the quadrilateral formed by the two triangles that share (x, y). This flip must be legal, since if (v, z) was already an edge, the graph would have K_4 as a subgraph, which is impossible for outerplanar graphs. Since each such flip increases the degree of v by one, $n - 1 - \deg(v)$ flips are both necessary and sufficient. □

With this property, Mori et al. showed that it is possible to quickly transform any Hamiltonian triangulation into the canonical form by decomposing it along the Hamiltonian cycle into two outerplanar graphs. Interestingly, this exact approach was already used 10 years earlier by Sleator, Tarjan and Thurston [10] to prove a $\Theta(n \log n)$ bound on the diameter of the flip graph when the vertices are labelled. Note that this was even before the linear bound by Komuro that was discussed in the previous section. However, they did not state their result in terms of unlabelled triangulations and it seems that both Komuro and Mori et al. were unaware of this earlier work.

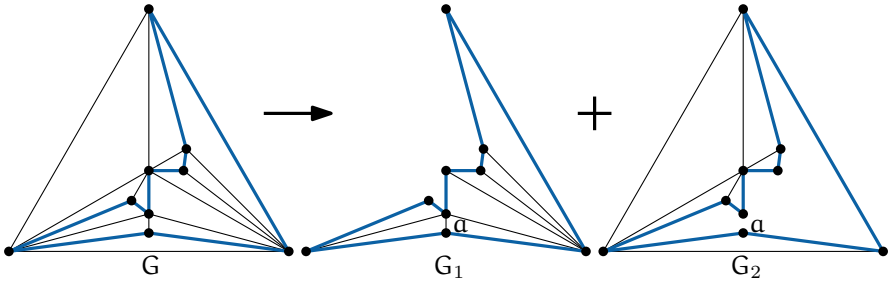


Figure 5.5: The decomposition of a Hamiltonian graph G into two outerplanar graphs G_1 and G_2. The vertex a has degree 2 in G_2.

THEOREM 5.7 (Mori et al. [8], Proposition 9). *Any Hamiltonian triangulation on n vertices can be transformed into $\triangle_n$ by at most $2n-10$ flips, preserving the existence of Hamiltonian cycles.*

Proof. Given a Hamiltonian triangulation G with Hamiltonian cycle C, we can decompose it into two outerplanar graphs G_1 and G_2, such that each contains C and all edges on one side of C. This is illustrated in Figure 5.5. Let a be a vertex of degree 2 in G_2. We are going to make a dominant in G_1. Since G is 3-connected and a has no additional neighbours in G_2, the degree of a in G_1 is at least 3. Thus by Lemma 5.6, we can make a dominant by at most $n-4$ flips. Each of these flips is valid, as a is not connected to anyone in G_2, except for its neighbours on C.

Now consider the subgraph $G_2' = G_2 \setminus \{a\}$. Since a has degree 2 in G_2, G_2' is still outerplanar, so by applying Lemma 5.6 again we can make a vertex of G_2' dominant as well, which gives us the canonical triangulation. Since G_2' has $n-1$ vertices and it always has a vertex of degree at least 4 (provided that $n \geqslant 6$), we need at most $n-6$ flips for this. Since we did not flip any of the edges on C, the theorem follows. □

This shows that the Hamiltonian triangulations are closely connected, so all we need to figure out is how we can quickly make a triangulation Hamiltonian. Here, we turn to an old result by Whitney [12] that shows that all 4-connected triangulations are Hamiltonian. Since a triangulation is 4-connected if and only if it does not

have any separating triangles (cycles of length 3 whose removal disconnects the graph; see Lemma 6.15 for a proof), by removing all separating triangles from a triangulation, we make it 4-connected and therefore Hamiltonian. Fortunately, separating triangles are easy to remove using flips, as the following lemmas show.

LEMMA 5.8 (Mori et al. [8], Lemma 11). *In a triangulation with $n \geqslant 6$ vertices, flipping any edge of a separating triangle $D = abc$ will remove that separating triangle. This never introduces a new separating triangle, provided that the selected edge belongs to multiple separating triangles or none of the edges of D belong to multiple separating triangles.*

Proof. Since D is separating and the newly created edge connects a vertex on the inside to a vertex on the outside, the flip is always legal. Since the flip removes an edge of D, it is no longer a separating triangle. Now suppose that we flipped (a, b) to a new edge (x, y) and introduced a new separating triangle D'. Then D' must be xyc. But since $n \geqslant 6$ and our construction so far uses only 5 vertices, one of the faces ayc, byc, axc, or bcx must be a separating triangle as well. This means that either (a, c) or (b, c) is an edge that belongs to multiple separating triangles, while (a, b) only belongs to D, which contradicts the choice of (a, b). □

LEMMA 5.9 (Mori et al. [8], Lemma 11). *Any triangulation on n vertices can be made 4-connected by at most $n - 4$ flips.*

Proof. We will show that a triangulation can have at most $n - 4$ separating triangles, the result follows by Lemma 5.8. The proof is by induction on n. For the base case, let $n = 4$. Then our graph must be K_4, which has no separating triangles as required. For the induction we can assume that our graph G has a separating triangle T which partitions G into two components G_1 and G_2. By induction, G_1 and G_2 have at most $n_1 - 4$ and $n_2 - 4$ separating triangles, where n_1 and n_2 are the number of vertices in G_1 and G_2, respectively, including the vertices of T. Therefore G can have at most $n_1 - 4 + n_2 - 4 + 1 = (n_1 + n_2 - 3) - 4 = n - 4$ separating triangles. □

Now we can prove the main result.

THEOREM 5.10 (Mori et al. [8], Theorem 4). *Any two triangulations on n vertices can be transformed into each other by at most $6n - 30$ flips.*

Proof. The connection between Lemma 5.9 and Theorem 5.7 is an old proof by Whitney [12] that any 4-connected triangulation is Hamiltonian. Therefore we can transform any triangulation into the canonical form by at most $n - 4 + 2n - 10 = 3n - 14$ flips. By looking carefully at the proof of Theorem 5.7, we see that if the graph is 4-connected, the first vertex (vertex a) is guaranteed to have degree at least 4, which

brings the bound down to $3n - 15$ flips to the canonical triangulation and $6n - 30$ flips between any pair of triangulations. □

Note that, although finding a Hamiltonian cycle is NP-hard in general [6], there exists a linear-time algorithm by Asano et al. for finding a Hamiltonian cycle in any 4-connected triangulation [1]. Thus, the assumption in the proof that the Hamiltonian cycle is given is not a practical concern when implementing the resulting algorithm.

5.5 LOWER BOUNDS

In addition to the upper bound described in Section 5.3, Komuro [7] also gave a lower bound on the diameter of the flip graph, based on the maximum degree of the vertices in the graph.

THEOREM 5.11 (Komuro [7], Theorem 5). *Let* G *be a triangulation on* n *vertices. Then at least* $2n - 2\Delta(G) - 3$ *flips are needed to transform* G *into the canonical triangulation, where* $\Delta(G)$ *denotes the maximum degree of* G.

Proof. Let a and b be the two vertices of degree $n - 1$ in the canonical triangulation. Each flip increases the degree in G of either a or b by at most one. The only possible exception is the flip that creates the edge (a, b), which increases the degree of both vertices by one. Since the initial degree of a and b is at most $\Delta(G)$, we need at least $2(n - 1 - \Delta(G)) - 1 = 2n - 2\Delta(G) - 3$ flips. □

Since there are triangulations that have maximum degree 6, this gives a lower bound of $2n - 15$ flips. It is interesting that one of the triangulations in the lower bound is the canonical form. This implies that either the lower bound is very far off, or the canonical triangulation is a bad choice of intermediate triangulation. It also means that as long as we use this canonical form, the best we can hope for is an upper bound of $4n - 30$ flips. Komuro also gave a lower bound on the number of flips required to transform between any pair of triangulations, again based on the degrees of the vertices.

THEOREM 5.12 (Komuro [7], Theorem 4). *Let* G *and* G' *be triangulations on* n *vertices. Let* $v_1, \ldots, v_n$ *and* $v'_1, \ldots, v'_n$ *be the vertices of* G *and* G'*, respectively, ordered by increasing degree. Then at least* $\frac{1}{4}D(G, G')$ *flips are needed to transform* G *into* G'*, where* $D(G, G') = \sum_{i=1}^{n} |\deg(v_i) - \deg(v'_i)|$.

Proof. Let σ be a mapping between the vertices of G and G' and suppose we transform G into G' using flips, such that $v_i \in G$ becomes $v'_{\sigma(i)} \in G'$. Since every flip changes the degree of a vertex by one, we need at least $|\deg(v_i) - \deg(v'_{\sigma(i)})|$ flips to obtain the correct degree for $v'_{\sigma(i)}$. However, each flip affects the degrees of 4 vertices, giving a bound of $\frac{1}{4}\sum_{i=1}^{n} |\deg(v_i) - \deg(v'_{\sigma(i)})|$ flips. Our actual lower bound

is the minimum of this bound over all mappings σ. Mapping every vertex to a vertex with the same rank when ordered by degree (i.e. $\sigma(i) = i$) achieves this minimum. □

This was the best known lower bound for almost twenty years, but in a recent pre-print, Frati [4] presented an improved lower bound, based on the notion of common edges.

THEOREM 5.13 (Frati [4], Lemma 1). *Let* G *and* G' *be triangulations on* n *vertices. Let* σ *be the bijection between vertices of* G *and* G' *that maximizes the number of common edges, and let* $c(\sigma)$ *be that number. Then any flip sequence that transforms* G *into* G' *has length at least* $3n - 6 - c(\sigma)$.

Proof. At the end of the flip sequence, the two graphs will be isomorphic, so they will have all $3n - 6$ edges in common. Since each flip can introduce at most one new common edge, the lower bound follows. □

Frati then constructs a graph G_{LB} that shares at most $\frac{2n}{3}$ edges with the canonical form, regardless of the bijection used. The graph consists of an arbitrary triangulation on $\frac{n}{3} + 2$ vertices with maximum degree six, with a degree-three vertex inserted in each face. The vertices of the original triangulation are colored blue, while the inserted vertices are colored red. Note that the red vertices form an independent set.

THEOREM 5.14 (Frati [4], Theorem 1). *The diameter of the flip graph is at least* $\frac{7n}{3} - 34$.

Proof. Consider any bijection between the vertices of G_{LB} and $\triangle_n$. Since the blue vertices had degree at most six before the red vertices were inserted, the maximum degree of G_{LB} is twelve. Thus, at most 24 of the edges incident to the dominant vertices of $\triangle_n$ can be common. Now consider the chain of edges not incident to the dominant vertices. Since no two red vertices in G_{LB} are adjacent, every edge on this chain must have one endpoint mapped to a blue vertex. But since there are only $\frac{n}{3} + 2$ blue vertices in G_{LB}, no more than $\frac{2n}{3} + 4$ edges on the chain can be common. Thus, the maximum number of common edges between G_{LB} and $\triangle_n$ is $\frac{2n}{3} + 28$, which by Theorem 5.13 gives a lower bound of $3n - 6 - (\frac{2n}{3} + 28) = \frac{7n}{3} - 34$ flips. □

BIBLIOGRAPHY

[1] Takao Asano, Shunji Kikuchi, and Nobuji Saito. A linear algorithm for finding Hamiltonian cycles in 4-connected maximal planar graphs. *Discrete Applied Mathematics*, 7(1):1–15, 1984.

[2] Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry: Theory and Applications*, 42(1):60–80, 2009.

[3] Prosenjit Bose and Sander Verdonschot. A history of flips in combinatorial triangulations. In *Proceedings of the XIV Spanish Meeting on Computational Geometry (EGC 2011)*, volume 7579 of *Lecture Notes in Computer Science*, pages 29–44. 2012.

[4] Fabrizio Frati. A lower bound on the diameter of the flip graph. *ArXiv e-prints*, 2015. arXiv:1508.03473 [cs.CG].

[5] Zhicheng Gao, Jorge Urrutia, and Jianyu Wang. Diagonal flips in labelled planar triangulations. *Graphs and Combinatorics*, 17(4):647–657, 2001.

[6] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. 1972.

[7] Hideo Komuro. The diagonal flips of triangulations on the sphere. *Yokohama Mathematical Journal*, 44(2):115–122, 1997.

[8] Ryuichi Mori, Atsuhiro Nakamoto, and Katsuhiro Ota. Diagonal flips in Hamiltonian triangulations on the sphere. *Graphs and Combinatorics*, 19(3):413–418, 2003.

[9] Seiya Negami and Atsuhiro Nakamoto. Diagonal transformations of graphs on closed surfaces. *Science Reports of the Yokohama National University. Section I. Mathematics, Physics, Chemistry*, (40):71–97, 1993.

[10] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Short encodings of evolving structures. *SIAM Journal on Discrete Mathematics*, 5(3):428–450, 1992.

[11] Klaus Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[12] Hassler Whitney. A theorem on graphs. *Annals of Mathematics, Second Series*, 32(2):378–390, 1931.

6

MAKING TRIANGULATIONS 4-CONNECTED USING FLIPS

In this chapter, we show that any combinatorial triangulation on n vertices can be transformed into a 4-connected one using at most $\lfloor (3n-9)/5 \rfloor$ edge flips. We also give an example of an infinite family of triangulations that requires this many flips to be made 4-connected, showing that our bound is tight. In addition, for $n \geqslant 19$, we improve the upper bound on the number of flips required to transform any 4-connected triangulation into the canonical triangulation (the triangulation with two dominant vertices), matching the known lower bound of $2n-15$. Our results imply a new upper bound on the diameter of the flip graph of $5.2n-33.6$, improving on the previous best known bound of $6n-30$.

This chapter was first published in the proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2011) [4], and was subsequently invited and accepted to a special issue of Computational Geometry: Theory and Applications [5]. It contains joint work with Prosenjit Bose, Dana Jansens, André van Renssen and Maria Saumell.

6.1 INTRODUCTION

As reviewed in Chapter 5, a lot of research has gone into the following question: "Given two combinatorial triangulations, how can we transform one into the other using edge flips?" The best known algorithm, developed independently by Sleator et al. [11] and Mori et al. [9], consists of two steps. In the first step, the given triangulation is transformed into a 4-connected one, using at most $n-4$ flips. Since a 4-connected triangulation is always Hamiltonian (an old result by Whitney [12]; in fact, the cycle can even be found quickly [2]), the resulting Hamiltonian triangulation is then transformed into the canonical one by at most $2n-11$ flips, using a decomposition into two outerplanar graphs that share a Hamiltonian cycle as their respective outer faces. Thus $6n-30$ flips are sufficient to transform any triangulation into any other. The algorithm and analysis are described in more detail in Section 5.4.

The upper bound on the number of flips used to make the triangulation 4-connected arises from the fact that any separating triangle can be removed by flipping one of its edges, and that a triangulation can have at most $n-4$ separating triangles. However, this analysis does not take advantage of the fact that if two separating triangles

share an edge, a single flip can remove both of them. In this chapter, we combine this observation with an edge charging scheme to show that any triangulation can be made 4-connected using at most $\lfloor (3n-9)/5 \rfloor$ flips.

The problem of making triangulations 4-connected has also been studied in the setting where many edges may be flipped simultaneously, provided none of them are part of the same triangle. Bose et al. [3] showed that any triangulation can be made 4-connected by one such simultaneous flip and that $O(\log n)$ simultaneous flips are sufficient and sometimes necessary to transform between two given triangulations.

The remainder of this chapter is organized as follows. In Section 6.2, we prove the new upper bound on the number of flips to make a triangulation 4-connected, thereby improving the first step of the construction by Mori et al. For $n \geqslant 19$, we also improve the bound on the second step of their algorithm to match the lower bound by Komuro [8]. This results in a new upper bound on the diameter of the flip graph of $5.2n - 33.6$. We then show in Section 6.3 that, when n is a multiple of 5, there are triangulations that require $(3n-10)/5 = \lfloor (3n-9)/5 \rfloor$ flips to be made 4-connected, showing that our bound is tight. Section 6.5 contains proofs for various technical lemmas that are used in the proof of the upper bound.

After completion of this chapter, Cardinal et al. [6] further improved the upper bound on the diameter of the flip graph to $5n - 23$ by proving that a triangulation can be directly transformed into a Hamiltonian one using at most $n/2$ flips. This allowed them to construct an arc drawing (a plane drawing with all vertices on a line and edges represented by a connected sequence of semi-circles centred on the line) for any planar graph, in which all edges are drawn as a single semi-circle, except for $n/2$ edges that are drawn as a sequence of two semicircles. In addition, they showed that there always exists a single simultaneous flip of fewer than $2n/3$ edges that makes a triangulation 4-connected, and that this bound is tight up to an additive constant.

6.2 UPPER BOUND

In this section we prove an upper bound on the number of flips to make any given triangulation 4-connected. Specifically, we show that $\lfloor (3n-9)/5 \rfloor$ flips always suffice. The proof references several technical lemmas whose proofs can be found in Section 6.5. We also prove that any 4-connected triangulation can be transformed into the canonical form using a worst-case optimal number of $2n-15$ flips. We start by providing more precise definitions of relevant concepts.

DEFINITIONS Our input consists of a triangulation T, along with a combinatorial embedding specifying the clockwise order of edges around each vertex of T. In addition, one of the faces of T is marked as the *outer face*. If an edge of the outer face is flipped, one of the two new faces is designated as the new outer face. A *separating triangle* D is a cycle in T of length three whose removal splits T into two (non-empty) connected components. We call the component that contains vertices of the outer face the *exterior* of D, and the other component the *interior* of D. A vertex in the interior of D is said to be *inside* D and likewise, a vertex in the exterior of D is said to be *outside* D. An edge is inside a separating triangle if one or both endpoints are inside.

A separating triangle A *contains* another separating triangle B if and only if the interior of B is a subgraph of the interior of A with a strictly smaller vertex set. If A contains B, A is the *containing* triangle. A separating triangle that is contained by the largest number of separating triangles in T is called *deepest*. Since containment is transitive, a deepest separating triangle cannot contain any separating triangles, as these would have a higher number of containing triangles.

ALGORITHM We use the same general strategy as the earlier algorithms - flip an edge of a separating triangle until there are none left. This strategy is guaranteed to terminate by the following Lemma (see Section 5.4, Lemma 5.8 for a proof).

LEMMA 6.1 (Mori et al. [9], Lemma 11). *In a triangulation on* $n \geqslant 6$ *vertices, flipping any edge of a separating triangle* D *will remove that separating triangle. This never introduces a new separating triangle, provided that the selected edge belongs to multiple separating triangles or none of the edges of* D *belong to multiple separating triangles.*

Since a triangulation is 4-connected if and only if it does not have any separating triangles (see Lemma 6.15), this strategy transforms any triangulation into one that is 4-connected. With this in mind, our algorithm works as follows. The reasoning behind some of the choices will become clear during the analysis.

ALGORITHM 1 (Make 4-connected)

- Find a deepest separating triangle D, preferring ones that do not use an edge of the outer face.
  - If D does not share any edge with other separating triangles, flip an edge of D that is not on the outer face.
  - If D shares exactly one edge with another separating triangle, flip this edge.
  - If D shares multiple edges with other separating triangles, flip one of the shared edges that is not shared with a containing triangle (such an edge always exists in this case).
- Repeat until T is 4-connected.

ANALYSIS Fundamentally, our analysis relies on counting edges. We separate the edges into two categories: edges that are part of some separating triangle, and edges that are not. We call the latter *free edges*. If there were no free edges, a triangulation would have sufficiently many edges for each of the maximum $n-4$ separating triangles (from Lemma 5.9) to be edge-disjoint. And since we need to flip at least one edge of every separating triangle in order to make the triangulation 4-connected, we would require $n-4$ flips. Thus, to get a better upper bound, we need to show that this situation is impossible. The following lemma does so, by showing that the presence of a separating triangle forces some other edges to be free.

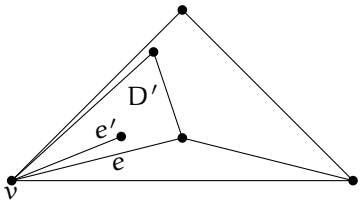


Figure 6.1: Every vertex of a separating triangle is incident to a free edge inside the triangle.

LEMMA 6.2. *In a triangulation, every vertex v of a separating triangle D is incident to at least one free edge inside D.*

Proof. Consider one of the edges of D that is incident to v. Since D is separating, its interior cannot be empty and since D is part of a triangulation, there is a triangular face inside D that uses this edge. Let e be the other edge of this face that is incident to v (see Figure 6.1).

The remainder of the proof is by induction on the number of separating triangles contained in D. For the base case, assume that D does not contain any other separating triangles. Then e must be a free edge and we are done.

For the induction step, there are two further cases. If e does not belong to a separating triangle, we are again done, so assume that e belongs to a separating triangle D'. Since D' is itself a separating triangle contained in D and containment is transitive, the number of separating triangles contained in D' must be strictly smaller than the number contained in D. Since v is also a vertex of D', our induction hypothesis tells us that there is a free edge incident to v inside D'. Since D' is contained in D, this edge is also inside D. □

This immediately gives us a better bound on the maximum number of edge-disjoint separating triangles.

COROLLARY 6.3. *If all separating triangles are edge-disjoint, a triangulation on n vertices can contain at most $(3n-10)/5$ separating triangles.*

Proof. We prove this by assigning five edges to each separating triangle without assigning any edge twice. First, each edge of a separating

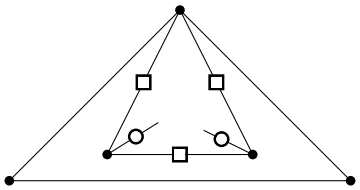


Figure 6.2: Each edge-disjoint separating triangle can be assigned five edges: the three edges of the triangle (squares) and two free edges incident to unshared vertices (circles).

triangle is assigned to that triangle. Since they are all edge-disjoint, this does not assign any edge twice. Next, we turn to the free edges identified by Lemma 6.2. We start with the topmost separating triangles, i.e. those who are not contained in any other separating triangle, and assign all three free edges (one per vertex) to them. Note that this assigns six edges to each of these triangles, instead of five – a fact we use later to tighten the bound.

Now consider a separating triangle D that is contained in some other separating triangles. We have to be careful to avoid free edges that have already been assigned to triangles that contain it. But since all separating triangles are edge-disjoint, D can share at most one vertex with a separating triangle that contains it (see Lemma 6.19). Thus, we can safely assign the two free edges that are incident to the unshared vertices to the triangle (see Figure 6.2).

Since we assigned five edges to each separating triangle, and a triangulation has exactly $3n-6$ edges, there can be at most $(3n-6)/5$ edge-disjoint separating triangles. However, recall that each topmost separating triangle was actually assigned six edges. And any edge that is not contained in a separating triangle has not been assigned at all. To prove a bound of $(3n-10)/5$ separating triangles, we need to find at least four edges between these two categories.

Consider the edges of the outer face. These edges are either free, or part of a topmost separating triangle. If all three edges of the outer face are free, then either there are no separating triangles, or there is at least one topmost separating triangle whose edge we can also use. In either case, we are done.

If only two edges of the outer face are free, there is a topmost separating triangle that uses the other edge, giving us three free edges already. But this triangle cannot use the vertex shared by the two free edges. Since this vertex has degree at least three, it is incident to either a free edge, or another topmost separating triangle, both of which give us four free edges.

Finally, if one or no edges of the outer face are free, we have three free edges between the edges of the outer face and the topmost separating triangles, so we just need to find one more. Consider a vertex v shared by two non-free edges of the outer face. Let D_1 and D_2 be the topmost separating triangles that use these edges. Since D_1 and D_2

cannot share an edge, there is at least one face adjacent to v that lies between D_1 and D_2. Consider the edge of that face opposite from v. If it is free, we are done. If it is not free, it must be used by another topmost separating triangle that does not use any edge of the outer face, also giving us a fourth free edge. Therefore a triangulation can contain no more than $(3n-10)/5$ separating triangles. □

Thus, if all separating triangles are edge-disjoint, we only need $(3n-10)/5$ flips to make a triangulation 4-connected. But what if some of the separating triangles do share edges? As it turns out, we can show a similar upper bound on the number of flips needed by the algorithm we presented earlier.

THEOREM 6.4. *A triangulation on $n \geqslant 6$ vertices can be made 4-connected using at most $\lfloor (3n-9)/5 \rfloor$ flips.*

Proof. We prove this using a charging scheme. We begin by placing a coin on every edge of the triangulation. Then we flip the edges indicated by the algorithm until no separating triangles remain, while paying five coins for every flip. The exact charging scheme will be described later. During this process, we maintain two invariants:

- Every edge of a separating triangle has a coin.
- Every vertex of a separating triangle has an incident free edge that is inside the triangle and has a coin.

These invariants have several nice properties. First, an edge can either be a free edge or belong to a separating triangle, but not both. So at any given time, only one invariant applies to an edge. Second, an edge only needs one coin to satisfy the invariants, even if it is on multiple separating triangles or is a free edge for multiple separating triangles. These two properties imply that the invariants hold initially, since by Lemma 6.2, every vertex of a separating triangle has an incident free edge.

We now show that these invariants are sufficient to guarantee that we can pay five coins for every flip. Consider the situation after we flip an edge that belongs to a deepest separating triangle D and satisfies the criteria of Lemma 6.1, but before we remove any coins. Since flipping the edge has removed D and no new separating triangles are introduced, both invariants still hold. We proceed by identifying four types of edges whose coins we can now remove to pay for this flip without upsetting the invariants.

TYPE 1 (■): The flipped edge e. By Lemma 6.1, e cannot belong to any separating triangle after the flip, so the first invariant still holds if we remove e's coin. Before the flip, e was not a free edge, so the second invariant was satisfied even without e's coin. Since the flip did not introduce any new separating triangles, this is still the case.

TYPE 2 (□): A non-flipped edge e of D that is not shared with any other separating triangle. By Lemma 6.1, the flip removed D and did not introduce any new separating triangles. Therefore e cannot belong to any separating triangle, so the first invariant still holds if we remove e's coin. By the same argument as for the previous type, e is also not required to have a coin to satisfy the second invariant.

TYPE 3 (○): A free edge e of a vertex of D that is not shared with any containing separating triangle. Since e did not belong to any separating triangle and the flip did not introduce any new ones, e is not required to have a coin to satisfy the first invariant. Further, since the flip removed D and D was deepest, e is not incident to a vertex of another separating triangle that contains it. Therefore it is no longer required to have a coin to satisfy the second invariant.

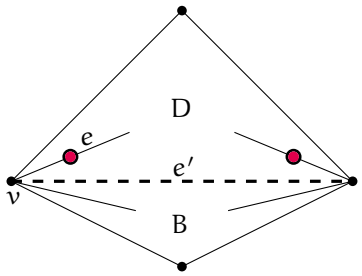


Figure 6.3: Two type 4 edges.

TYPE 4 (●): A free edge e incident to a vertex v of D, where v is an endpoint of an edge e' of D that is shared with a non-containing separating triangle B, provided that we flip e' (illustrated in Figure 6.3). Any separating triangle that contains D but not B must share e' (Lemma 6.21) and is therefore removed by the flip.
So every separating triangle after the flip that contains D also contains B. In particular, this also holds for containing triangles that share v. Since the second invariant requires only one free edge with a coin for each vertex of a separating triangle, we can safely charge the one inside D, as long as we do not charge the free edge in B.

To decide which edges we charge for each flip, we distinguish five cases, based on the number of edges D shares with other separating triangles and whether any of these triangles contain D. These cases are illustrated in Figures 6.4, 6.5, and 6.6.

CASE 1. D does not share any edges with other separating triangles (Figure 6.4a). In this case, we flip any of D's edges. By the first invariant, each edge of D has a coin. These edges all fall into Types 1 and 2, so we use their coins to pay for the flip. Further, D can share at most one vertex with a containing triangle (Lemma 6.19), so we charge two free edges, each incident to one of the other two vertices (Type 3).

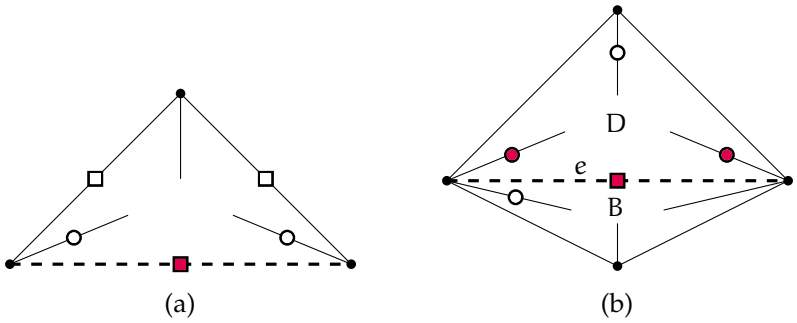


Figure 6.4: The edges that are charged if (a) the deepest separating triangle does not share any edges with other separating triangles, or (b) the deepest separating triangle only shares edges with non-containing separating triangles. The flipped edge is dashed and the charged edges are marked with filled boxes (Type 1), empty boxes (Type 2), empty disks (Type 3) or filled disks (Type 4).

CASE 2. D does not share any edge with a containing triangle, but shares one or more edges with non-containing separating triangles (Figure 6.4b). In this case, we flip one of the shared edges e. We charge e (Type 1) and two free edges inside D that are incident to the vertices of e (Type 4). This leaves us with two more coins that we need to charge.

Let B be the non-containing separating triangle that shares e with D. We first show that B must have the same depth as D. There can be no separating triangles that contain D but not B, as any such triangle would have to share e (Lemma 6.21) and D does not share any edge with a containing triangle. Therefore any triangle that contains D must contain B as well. Since D is contained in the maximal number of separating triangles, this holds for B as well. This means that B cannot contain any separating triangles and to satisfy the second invariant we only need to concern ourselves with triangles that contain both B and D.

Now consider the number of vertices of the quadrilateral formed by B and D that can be shared with containing triangles. Since D does not share an edge with a containing triangle, it can share at most one vertex with a containing triangle (Lemma 6.19). Now suppose that B shares an edge with a containing triangle. Then one of the vertices of this edge is part of D as well. Since the other two vertices of the quadrilateral are both part of D, they cannot be shared with containing triangles. On the other hand, if B does not share an edge with a containing triangle, it too can share at most one vertex with containing triangles. Thus, in both cases, at most two vertices of the quadrilateral can be shared with containing triangles, which means that there are at least two vertices that are not shared. For each of these vertices, if it is the vertex of D that is not shared with B, we charge the free edge in D, otherwise we charge the free edge in B (both Type 3).

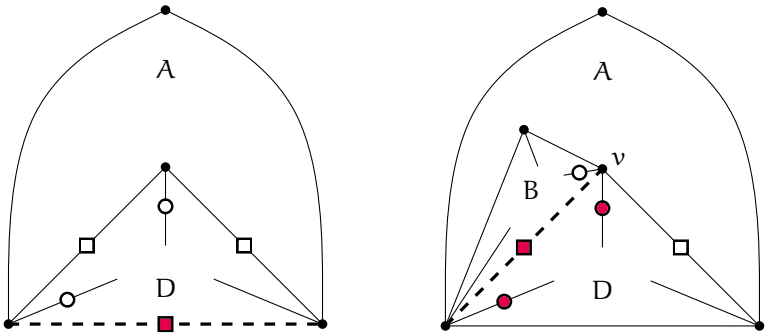


Figure 6.5: The edges that are charged if the deepest separating triangle shares an edge with a containing triangle, and zero (a) or one (b) edges with non-containing separating triangles.

CASE 3. D shares an edge with a containing triangle A and does not share the other edges with any separating triangle (Figure 6.5a). In this case, we flip the shared edge and charge all of D's edges, since one is the flipped edge (Type 1) and the others are not shared (Type 2). The vertex of D that is not shared with A cannot be shared with any containing triangle (Lemma 6.20), so we charge a free edge incident to this vertex (Type 3).

Further, if A shares an edge with a containing triangle, it either shares the flipped edge, which means that the containing triangle is removed by the flip, or it shares another edge, in which case the vertex that is not an endpoint of this edge cannot be shared with any containing triangle. If A does not share an edge with a containing triangle, it can share at most one vertex with a containing triangle (Lemma 6.19). In both cases, one of the vertices of the flipped edge is not shared with any containing triangle (Type 3), so we charge a free edge incident to it.

CASE 4. D shares an edge with a containing triangle A and exactly one other edge with a non-containing separating triangle B (Figure 6.5b). In this case, we flip the edge that is shared with B. Let v be the vertex of D that is not shared with A. We charge the flipped edge (Type 1), the unshared edge of D (Type 2) and two free edges inside D that are incident to the vertices of the flipped edge (Type 4). We charge the last coin from a free edge in B that is incident to v. We can charge it, since v cannot be shared with a triangle that contains D (Lemma 6.20) and every separating triangle that contains B but not D must share the flipped edge as well (Lemma 6.21) and is therefore removed by the flip.

All that is left is to argue that there can be no separating triangle contained in B that requires the coin on this free edge to satisfy the second invariant. Every separating triangle that contains D but not B must share the flipped edge (Lemma 6.21). Since D already shares

another edge with a containing triangle and it cannot share two edges with containing triangles (Lemma 6.18), all separating triangles that contain D must also contain B. Since D is deepest, B must be deepest as well and therefore cannot contain any separating triangles.

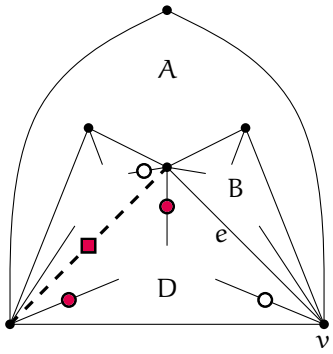


Figure 6.6: The edges that are charged if the deepest separating triangle shares an edge with a containing triangle and both other edges with non-containing separating triangles.

CASE 5. D shares one edge with a containing triangle A and the other two with non-containing separating triangles (Figure 6.6). In this case we also flip the edge shared with one of the non-containing triangles. The charged edges are identical to the previous case, except that there is no unshared edge any more. Instead, we charge the last free edge in D.

Before we argue why we are allowed to charge it, we need to give some names. Let e be the edge of D that is not shared with A and is not flipped. Let B be a non-containing triangle that shares e with D and let v be the vertex that is shared by A, B and D. Now, any separating triangle that shares v and contains D must contain B as well. If it did not, it would have to share e with D, but D already shares an edge with a containing triangle and cannot share more than one (Lemma 6.18). Since the second invariant requires only a single free edge with a coin for each vertex of a separating triangle, it is enough that v still has an incident free edge with a coin in B.

This shows that we can charge 5 coins for every flip while maintaining the invariants, but we still need to show that after performing these flips we have indeed removed all separating triangles. So suppose that our graph contains separating triangles. Since each separating triangle is contained in a certain number of other separating triangles (which can be zero), there is at least one deepest separating triangle D. Since D shares at most one edge with containing separating triangles (Lemma 6.18), one of the cases above must apply. This gives us an edge of D to flip and five edges to charge, each of which is guaranteed by the invariants to have a coin. Therefore the process stops only after all separating triangles have been removed.

Finally, since we pay 5 coins per flip and there are $3n-6$ edges, by initially placing a coin on each edge, we flip at most $\lfloor (3n-6)/5 \rfloor$ edges. Now consider the edges of the outer face. We show that these still have a coin at the end of the algorithm. By definition, these edges are not inside any separating triangle and since we only charge free edges inside separating triangles, they can only ever be charged as Type 1 or 2. Thus, if an edge of the outer face gets charged, it was part of the deepest separating triangle D that was removed by the flip. Since an edge of the outer face cannot be shared with a non-containing separating triangle and it cannot be contained by any separating triangle, it can only be charged in Case 1 or 3. In Case 1, we charge only two of the free edges inside D, since there could be a containing separating triangle that shares just a vertex. However, this is not possible if D uses an edge of the outer face (Lemma 6.22), so we can charge this free edge instead of the edge of the outer face. Since we flip one of the edges that is not on the outer face, after the flip, all edges of the outer face still have their coins. In Case 3 we can charge this remaining free edge for the same reasons. However, since in this case we actually flip the edge of the outer face, we are not done yet. The outer face after the flip consists of the flipped edge, one edge of the current outer face and a current interior edge. Charging the extra free edge guarantees that the flipped edge can retain its coin, but we need to ensure that the current interior edge has a coin as well. Let A be the deepest of the separating triangles that contain D. Since it, too, uses an edge of the outer face, A can only be contained in triangles that share this edge (Lemma 6.22). It also cannot contain any separating triangles other than D, as these would be deepest as well and we prefer to remove separating triangles that do not use an edge of the outer face. Therefore there can be no other separating triangle that uses the free edge incident to the vertex of A that is not on the outer face and we can move this coin to the new edge of the outer face. Since this is the only case in which an edge of the outer face is flipped, this shows that the edges of the outer face retain their coins during the entire process. Therefore we actually only need $3n-9$ coins, resulting in a maximum of $\lfloor (3n-9)/5 \rfloor$ flips. □

TRANSFORMING HAMILTONIAN TRIANGULATIONS Now that we improved the bound on the number of flips needed during the first step of the algorithm by Mori et al. [9], we can turn our attention to the second step. This step consists of transforming the obtained 4-connected triangulation into the canonical form. Mori et al. showed that this can be done using at most $2n-11$ flips. We improve this slightly to $2n-15$ flips, matching the lower bound by Komuro [8] (see Theorem 5.11). We first need to prove a few more lemmas.

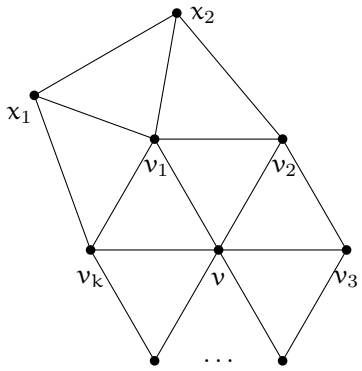


Figure 6.7: The neighbourhood of a vertex with degree at least 6 in a 4-connected triangulation.

LEMMA 6.5. *In a 4-connected triangulation on $n \geqslant 13$ vertices, every vertex of degree at least 6 either has a neighbour of degree at least 6, or it can be connected to a vertex of degree at least 5 by a single flip.*

Proof. Let v be a vertex of degree at least 6. Komuro [8] showed that either the graph consists of a cycle of length $n-2$ with v and one other vertex connected to every vertex on the cycle, or v has a neighbour with degree at least 5. In the first case, there is a vertex of high degree that can be connected to v by a single flip, so assume that this is not the case. Let v_1 be a neighbour of v with degree at least 5 and let $v_2, \ldots, v_k$ be the other neighbours of v, in clockwise order from v_1. Suppose that none of these neighbours have degree at least 6. Since the graph is 4-connected, this means that each has degree 4 or 5 and v_1 has degree exactly 5. Furthermore, no edge can connect two non-consecutive neighbours of v, as this would create a separating triangle. Let x_1 and x_2 be the neighbours of v_1 that are not adjacent to v, in clockwise order (see Figure 6.7). We distinguish two cases, based on the degree of v_2:

If v_2 has degree 4, x_2 must be connected to v_3. Both x_1 and x_2 can be connected to v with a single flip, so if either has degree at least 5, we are done. The only way to keep their degree at 4 is to connect both x_1 and v_k to v_3. But this would give v_3 degree at least 6, which is a contradiction. Therefore either x_1 or x_2 must have degree at least 5.

If v_2 has degree 5, let x_3 be its new neighbour. Again, if one of x_1, x_2 or x_3 has degree at least 5, we are done. Since x_2 already has degree 4, the only way to keep its degree below 5 is to connect x_1 and x_3 by an edge. But then both x_1 and x_3 have degree 4 and the only way to keep one at degree 4 is to create an edge to the other. Therefore at least one of x_1, x_2 or x_3 must have degree at least 5. □

In 1931, Whitney [12] showed that any 4-connected triangulation has a Hamiltonian cycle. The main ingredient of his proof is the following lemma:

LEMMA 6.6 (Whitney [12]). *Consider a cycle C in a 4-connected triangulation, along with two distinct vertices a and b on C. These vertices split C into two paths C_1 and C_2 with a and b as endpoints. Consider all edges on one side of the cycle, say the inside. If no vertex on C_1 (resp. C_2) is connected to another vertex on C_1 (resp. C_2) by an edge inside C, we can find a path from a to b that passes through each vertex on and inside C exactly once and uses only edges of C and inside C.*

We use this to prove the following lemma:

LEMMA 6.7. *For every edge (u, v) in a 4-connected triangulation, there is a Hamiltonian cycle that uses (u, v) such that all non-cycle edges incident to u are on one side of the cycle and all non-cycle edges incident to v are on the other side.*

Proof. Let x and y be the other vertices of the faces that have (u, v) as an edge. Let $v, x, u_1, \ldots, u_k, y$ be the neighbours of u in counterclockwise order and let $y, v_1, \ldots, v_m, x, u$ be the neighbours of v (see Figure 6.8). Note that all the u_i and v_i are distinct vertices, as a vertex other than x or y that is adjacent to both u and v would form a separating triangle. This means that $x, u_1, \ldots, u_k, y, v_1, \ldots, v_m, x$ forms a cycle. Moreover, no two non-consecutive neighbours of u can be connected by an edge, since this would create a separating triangle as well. Since this holds for the neighbours of v as well, x and y split the cycle into two parts that satisfy the conditions of Lemma 6.6. If we call the side of the cycle that does not contain (u, v) the inside, this means that we can find a path from x to y that passes through each vertex on and inside the cycle exactly once and uses only edges of and inside the cycle. This path can be completed to a Hamiltonian cycle that satisfies the conditions by adding the edges (y, u), (u, v) and (v, x). □

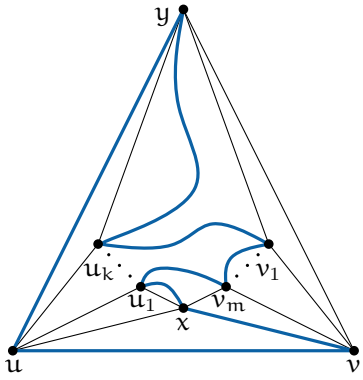


Figure 6.8: A possible Hamiltonian cycle that uses (u, v) and has all non-cycle edges incident to u on one side of the cycle and all non-cycle edges incident to v on the other.

THEOREM 6.8. *Any 4-connected triangulation* T *on* $n \geqslant 13$ *vertices can be transformed into the canonical triangulation using at most* $2n - \Delta(T) - 8$ *flips, where* $\Delta(T)$ *is the maximum degree among vertices of* T.

Proof. We use the same approach as used by Mori et al. [9] in the proof of Theorem 5.7, but instead of taking an arbitrary Hamiltonian cycle, we use the preceding lemmas to carefully construct a good cycle.

Let x be a vertex of maximal degree in T and suppose for now that x has a neighbour y with degree at least 6. We use the cycle given by Lemma 6.7 to decompose T into two outerplanar graphs T_1 and T_2, each sharing the cycle and having all edges on the inside and outside, respectively. Note that x is an ear in one of these, say T_2, while y is an ear in the other. Mori et al. showed that we can make any vertex v of an outerplanar graph dominant using at most $n - d_v - 1$ flips, where d_v is the degree of v. Therefore we can make x dominant in T_1 using at most $n - \Delta(T) - 1$ flips. These flips are allowed because x does not have any incident edges in T_2. Then we can make y dominant in T_2 using at most $n - d_y - 1 \leqslant n - 7$ flips. Thus we can transform T into the canonical triangulation using at most $2n - \Delta(T) - 8$ flips.

Since any triangulation on $n \geqslant 13$ vertices has a vertex of degree at least 6, if x does not have a neighbour with degree at least 6, Lemma 6.5 tells us that there is a vertex v with degree at least 5 that can be connected to x by a single flip. We perform this flip and use v in the place of y. Since x now has degree $\Delta(T) + 1$, we can make it dominant using at most $n - \Delta(T) - 2$ flips. Similarly, v has degree at least 6 after the flip, so we can make it dominant using at most $n - 7$ flips. Including the initial flip, we again obtain the canonical triangulation using at most $2n - \Delta(T) - 8$ flips. □

Combining this result with Theorem 6.4 gives the following bound on the maximum flip distance between two triangulations.

COROLLARY 6.9. *Any two triangulations* T_1 *and* T_2 *can be transformed into each other using at most* $5.2n - 19.6 - \Delta(T_1) - \Delta(T_2)$ *flips, where* $\Delta(T)$ *is the maximum degree among vertices of* T.

Theorem 6.8 matches the worst-case lower bound of $2n - 15$ flips if the maximum degree is at least 7, but we need a stronger result if the maximum degree is 6.

LEMMA 6.10. *In a 4-connected triangulation on* $n \geqslant 19$ *vertices with maximum degree 6, there is always a pair of vertices of degree 6 that can be connected by a flip.*

Proof. Suppose that such a pair does not exist and consider the neighbourhood of a vertex v of degree 6. Each edge incident to v can be flipped, otherwise there would be an edge connecting two non-consecutive neighbours of v, forming a separating triangle. Thus there

are 6 pairs of vertices that can be connected by a flip and one vertex of each pair needs to have degree at most 5. To realize this, v needs to have at least 4 neighbours of degree at most 5. Similarly, a vertex of degree 5 needs at least 3 such neighbours and a vertex of degree 4 needs at least 2. Therefore each vertex of degree at most 5 can have at most 2 neighbours of degree 6.

Let n_d be the number of vertices of degree d and let k be the number of edges between vertices of degree 6 and vertices of degree at most 5. Every vertex of degree 6 needs at least 4 neighbours of degree at most 5, so $k \geqslant 4n_6$. But every vertex of degree at most 5 can have at most 2 neighbours of degree 6, so $k \leqslant 2(n_4 + n_5)$. Combining these inequalities, we get that $n_6 \leqslant (n_4 + n_5)/2$. Since a triangulation with maximum degree 6 can have at most 12 vertices of degree less than 6, it follows that $n = n_4 + n_5 + n_6 \leqslant 18$. Thus for $n \geqslant 19$, there is always a pair of vertices of degree 6 that can be connected by a flip. □

THEOREM 6.11. *Any 4-connected triangulation on $n \geqslant 19$ vertices with maximum degree 6 can be transformed into the canonical triangulation using at most $2n - 15$ flips.*

Proof. By Lemma 6.10, there is always a pair of vertices x and y of degree 6 that can be connected by a flip. We first perform the flip that connects x and y, giving both vertices degree 7. We then proceed similarly to the proof of Theorem 6.8. We make x dominant in one of the outerplanar graphs using $n - 8$ flips and we make y dominant in the other, also using $n - 8$ flips. Counting the initial flip, we obtain the canonical triangulation using at most $2n - 15$ flips. □

By combining this with Theorem 6.8, we get the following bound.

COROLLARY 6.12. *Any 4-connected triangulation T on $n \geqslant 19$ vertices can be transformed into the canonical triangulation using at most $\min\{2n - 15, 2n - \Delta(T) - 8\}$ flips, where $\Delta(T)$ is the maximum degree among vertices of T.*

Proof. This follows from Theorems 6.8 and 6.11, along with the observation that $2n - 15 < 2n - \Delta(T) - 8$ if $\Delta(T)$ is 6 and $2n - \Delta(T) - 8 \leqslant 2n - 15$ if $\Delta(T) \geqslant 7$. □

And finally, using our bound from Theorem 6.4 on the number of flips it takes to make triangulations 4-connected, we obtain an improved bound on the diameter of the flip graph.

COROLLARY 6.13. *The diameter of the flip graph of all triangulations on $n \geqslant 19$ vertices is at most $5.2n - 33.6$.*

Proof. By Theorem 6.4, any triangulation can be made 4-connected using at most $\lfloor (3n - 9)/5 \rfloor$ flips. By Corollary 6.12, we can transform the resulting graph into the canonical triangulation using at most

$2n - 15$ flips. Hence, we can transform any triangulation into any other using at most $2 \cdot \big((3n-9)/5 + 2n - 15\big) = 5.2n - 33.6$ flips. □

6.3 LOWER BOUND

In this section we present a lower bound on the number of flips required to remove all separating triangles from a triangulation. Specifically, we present a triangulation that has $(3n-10)/5$ edge-disjoint separating triangles. This matches our upper bound on the number of edge-disjoint separating triangles in a triangulation and shows that there indeed exist triangulations that require this many flips to make them 4-connected.

The triangulation that gives rise to the lower bound is constructed recursively and resembles the Sierpiński triangle [10]. The construction starts with an empty triangle. The recursive step consists of adding an inverted triangle in the interior and connecting each vertex of the new triangle to the two vertices of the opposing edge of the original triangle. This is recursively applied to the three new triangles that share an edge with the inserted triangle, but not to the inserted triangle itself (see Figure 6.9). After k iterations, instead of applying the recursive step again, we add a single vertex in the interior of each triangle we are recursing on and connect this vertex to each vertex of the triangle. We also add a single vertex in the exterior face so that the original triangle becomes separating. The resulting triangulation is called $\mathcal{T}_k$.

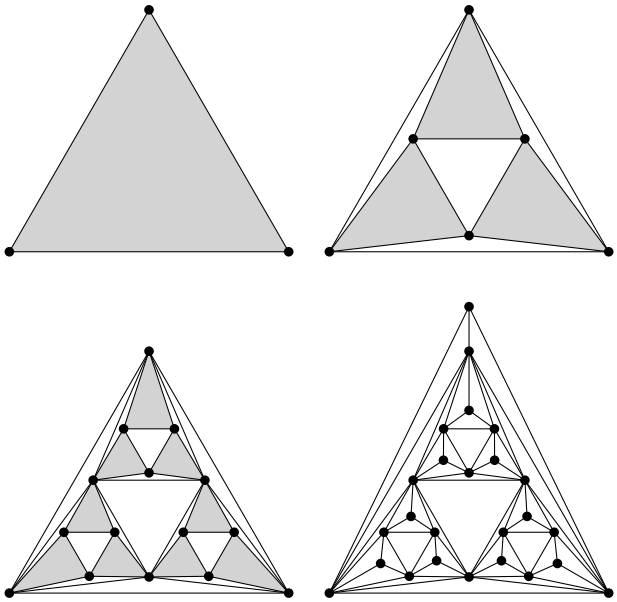


Figure 6.9: The stepwise construction of $\mathcal{T}_2$. The triangles used in the next recursive step are shaded.

THEOREM 6.14. *There are triangulations on n vertices that require $(3n-10)/5$ flips to make them 4-connected, where n is a multiple of 5.*

Proof. In the construction scheme presented above, each of the triangles we recurse on becomes a separating triangle that does not share any edges with the original triangle or the other triangles that we recurse on. Thus all these separating triangles are edge-disjoint. But how many of these triangles do we get? Let L_i be the number of triangles that we recurse on after i iterations of the construction, so $L_0 = 1$, $L_1 = 3$, etc. Now let V_i be the number of vertices of $\mathcal{T}_i$. We can see that $V_1 = 10$ and if we transform $\mathcal{T}_1$ into $\mathcal{T}_2$, we have to remove each of the interior vertices added in the final step and replace them with a configuration of 6 vertices. So to get $\mathcal{T}_2$, we add 5 vertices in each of the L_1 triangles. This is true in general, giving

$$V_i = V_{i-1} + 5L_{i-1} = 10 + 5\sum_{j=2}^{i} L_{j-1}. \tag{6.1}$$

Let S_i be the number of separating triangles of $\mathcal{T}_i$. We can see that $S_1 = 4$ and each recursive refinement of a separating triangle leaves it intact, while adding 3 new ones. Therefore

$$S_i = S_{i-1} + 3L_{i-1} = 4 + 3\sum_{j=2}^{i} L_{j-1}. \tag{6.2}$$

From Equation (6.1), we get that

$$\sum_{j=2}^{i} L_{j-1} = \frac{V_i - 10}{5}.$$

Substituting this into Equation (6.2) gives

$$S_i = 4 + 3 \cdot \frac{V_i - 10}{5} = \frac{3V_i - 10}{5}.$$

Since each flip removes only the separating triangle that the edge belongs to, we need $(3n-10)/5$ flips to make this triangulation 4-connected. Constructions for multiples of 5 between V_i and V_{i+1} can be obtained by recursing on a subset of the triangles in the final recursion step. □

Note that this triangulation achieves the upper bound on the number of edge-disjoint separating triangles from Corollary 6.3. It is natural to wonder whether this construction also leads to a better lower bound for the diameter of the flip graph in general. This is unfortunately not the case, as the resulting triangulation is Hamiltonian (see Figure 6.10). Thus, even though it is not 4-connected, we know that it can be transformed into the canonical triangulation by at most $2n - 11$ flips from the proof by Mori et al. [9].

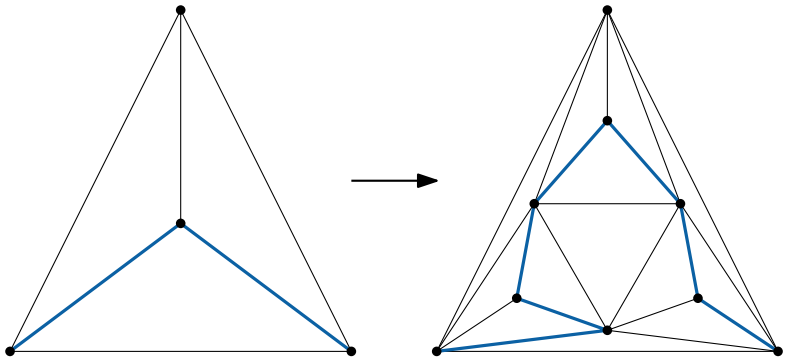


Figure 6.10: Updating the Hamilton cycle (bold) after a recursive step. The top vertex is visited at some other point on the initial cycle.

6.4 CONCLUSIONS AND OPEN PROBLEMS

We showed that any triangulation on n vertices can be made 4-connected using at most $\lfloor (3n-9)/5 \rfloor$ flips, while there are triangulations that require $(3n-10)/5 = \lfloor (3n-9)/5 \rfloor$ flips when n is a multiple of 5. This shows that our bound is tight for an infinite family of values for n, although a slight improvement to $\lfloor (3n-10)/5 \rfloor$ is still possible. We believe that this is the true bound. We also showed that any 4-connected triangulation on $n \geqslant 19$ vertices can be transformed into the canonical form using at most $2n-15$ flips. This matches the lower bound by Komuro [8] in the worst case where the graph has maximum degree 6 and results in a new upper bound of $5.2n-33.6$ on the diameter of the flip graph. It also means that both steps of the algorithm, when considered individually, are now tight in the worst case. Therefore, any further improvement must either merge the two steps in some fashion or employ a different technique.

Since 4-connectivity is not a necessary condition for Hamiltonicity, one possible approach is to show that a triangulation can be made Hamiltonian with fewer than $\lfloor (3n-9)/5 \rfloor$ flips. Indeed, Cardinal et al. [6] successfully used this approach to lower the bound further to $5n-23$, by showing that $n/2$ flips suffice to transform any triangulation into a Hamiltonian one. However, this still leaves a gap with the current best lower bound of $(n-8)/3$ flips, due to Aichholzer et al. [1].

Furthermore, all of the current algorithms use the same, single, canonical form. Surprisingly, the best known *lower bound* on the diameter of the flip graph (which was recently improved to $\frac{7n}{3}-34$ by Frati [7]) actually goes to the canonical form as well. This suggests that at least one of the two bounds still has significant room for improvement. So is there another canonical form that gives a better upper bound? Or can we get a better bound by using multiple canonical forms and picking the closest?

Another interesting problem is to minimize the number of flips to make a triangulation 4-connected. We showed that our technique is worst-case optimal, but there are cases where fewer flips would suffice. There is a natural formulation of the problem as an instance of 3-hitting set, where the subsets correspond to the edges of separating triangles and we need to pick a minimal set of edges such that we include at least one edge from every separating triangle. This gives a simple 3-approximation algorithm that picks an arbitrary separating triangle and flips all shared edges or an arbitrary edge if there are no shared edges. However, it is not clear whether the problem is even NP-hard, as not all instances of 3-hitting set can be encoded as separating triangles in a triangulation. Therefore it might be possible to compute the optimal sequence in polynomial time.

6.5 LEMMAS AND PROOFS

This section contains proofs for the technical lemmas used in the proof of Theorem 6.4.

LEMMA 6.15. *A triangulation is 4-connected if and only if it contains no separating triangles.*

Proof. The first direction is easy. If a triangulation has a separating triangle, by definition, removing three vertices is sufficient to disconnect the graph, which implies that it is not 4-connected. For the other direction, assume that we have a triangulation T that is not 4-connected. Since any triangulation is 3-connected, T must have a cut set of size three that separates the graph into components T_1 and T_2. Consider a vertex v in this cut set. This vertex must have neighbours in both T_1 and T_2. If not, the other two vertices would form a cut set of size two, which cannot exist as T is 3-connected.

Now look at the clockwise order of the neighbours of v, excluding the other vertices in the cut set. At some point, v has a neighbour v_1 in T_1, followed by a neighbour v_2 in T_2. If there were no other edges separating these, the edge (v_1, v_2) would be part of our triangulation, contradicting the fact that v is part of a cut set. Therefore these edges must be separated by an edge to a neighbour neither in T_1, nor T_2: a vertex of the cut set. The same argument holds for the transition from T_2 to T_1. Thus v is connected to both other vertices in the cut set. And since our choice of v was arbitrary, this same argument applies to them. Therefore this cut set must be a separating triangle. □

LEMMA 6.16. *If a separating triangle* A *contains a separating triangle* B*, then there is a vertex of* B *inside* A *and no vertex of* B *can lie outside* A*.*

Proof. Let z be a vertex in the interior of B and let y be a vertex of A that is not shared with B. Since the interior of B is a subgraph of the interior of A and y is not inside A, y must be outside B. Since

every triangulation is 3-connected, there is a path from z to y that stays inside A. This path connects the interior of B to the exterior, so there must be a vertex of B on the path and hence inside A.

Now suppose that there is another vertex of B outside A. Since all vertices of a triangle are connected by an edge, there is an edge between this vertex and the vertex of B inside A. This contradicts the fact that A is a separating triangle, so no such vertex can exist. □

LEMMA 6.17. *If a vertex x of a separating triangle B is inside a separating triangle A, then A contains B.*

Proof. Let y be a vertex of A that is not shared with B. There is a path from y to the outer face that stays in the exterior of A. There can be no vertex of B on this path, since this would create an edge between the interior and exterior of A. Therefore y is outside B.

Now suppose that A does not contain B. Then there is a vertex z inside B that is not inside A. There must be a path from z to x that stays inside B. Since x is inside A, there must be a vertex of A on this path. But since y is outside B, this would create an edge between the interior and exterior of B. Therefore A must contain B. □

LEMMA 6.18. *A separating triangle can share at most one edge with containing triangles.*

Proof. Suppose we have a separating triangle D that shares two of its edges with separating triangles that contain it. First of all, these triangles cannot be the same, since then they would be forced to share the third edge as well, which means that they are D. Since a triangle does not contain itself, this is a contradiction. So call one of these triangles A and call one of the triangles that shares the other edge B. Let x, y and z be the vertices of D, such that x is shared with A and B, y is shared only with A and z is shared only with B.

By Lemma 6.16, z must be inside A, while y must be inside B, since in both cases the other two vertices of D are shared and therefore not in the interior. But then by Lemma 6.17, A contains B and B contains A. This is a contradiction, since by transitivity it would imply that the interior of A is a subgraph of itself with a strictly smaller vertex set. □

LEMMA 6.19. *A separating triangle D that shares no edge with containing triangles can share at most one vertex with containing triangles.*

Proof. Suppose that D shares two of its vertices with containing triangles. First, both vertices cannot be shared with the same containing triangle, since then the edge between these two vertices would also be shared. Now let A be one of the containing triangles and let B be one of the containing triangles sharing the other vertex. By Lemma 6.16, there must be a vertex of D inside A. So then both vertices of D that are not shared with A must be inside A, otherwise there would be

an edge between the interior and the exterior of A. In particular, the vertex shared by B and D lies inside A, which by Lemma 6.17 means that A contains B. But the reverse is also true, so B contains A as well, which is a contradiction. □

LEMMA 6.20. *A separating triangle that shares an edge with a containing triangle cannot share the unshared vertex with another containing triangle.*

Proof. Suppose we have a separating triangle $D = (x, y, z)$ that shares an edge (x, y) with a containing triangle A and the other vertex z with another containing triangle B. By Lemma 6.16, at least one of x and y has to be inside B. Since these are vertices of A, by Lemma 6.17, B contains A. Similarly, z has to be inside A and since it is a vertex of B, A contains B. This is a contradiction. □

LEMMA 6.21. *Given two separating triangles* A *and* B *that share an edge* e*, any separating triangle that contains* A *but not* B *must use* e*.*

Proof. Suppose that we have a separating triangle D that contains A, but not B and that does not use one of the vertices v of e. By Lemma 6.16, v must be inside D. But then D would also contain B by Lemma 6.17, as v is a vertex of B as well. Therefore D must share both vertices of e and hence e itself. □

LEMMA 6.22. *A separating triangle* D *that uses an edge* e *of the outer face cannot be contained in a separating triangle that does not share* e*.*

Proof. Suppose D is contained in a separating triangle A. If A does not share e, by Lemma 6.16, at least one of the vertices of e must be inside A. But since e is part of the outer face, this is a contradiction. □

BIBLIOGRAPHY

[1] Oswin Aichholzer, Clemens Huemer, and Hannes Krasser. Triangulations without pointed spanning trees. *Computational Geometry: Theory and Applications*, 40(1):79–83, 2008.

[2] Takao Asano, Shunji Kikuchi, and Nobuji Saito. A linear algorithm for finding Hamiltonian cycles in 4-connected maximal planar graphs. *Discrete Applied Mathematics*, 7(1):1–15, 1984.

[3] Prosenjit Bose, Jurek Czyzowicz, Zhicheng Gao, Pat Morin, and David R. Wood. Simultaneous diagonal flips in plane triangulations. *Journal of Graph Theory*, 54(4):307–330, 2007.

[4] Prosenjit Bose, Dana Jansens, André van Renssen, Maria Saumell, and Sander Verdonschot. Making triangulations 4-connected using flips. In *Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2011)*, pages 241–247, 2011.

[5] Prosenjit Bose, Dana Jansens, André van Renssen, Maria Saumell, and Sander Verdonschot. Making triangulations 4-connected using flips. *Computational Geometry: Theory and Applications*, 47(2A):187–197, 2014. Special issue for CCCG 2011.

[6] Jean Cardinal, Michael Hoffmann, Vincent Kusters, Csaba D. Tóth, and Manuel Wettstein. Arc diagrams, flip distances, and Hamiltonian triangulations. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, pages 197–210, 2015.

[7] Fabrizio Frati. A lower bound on the diameter of the flip graph. *ArXiv e-prints*, 2015. arXiv:1508.03473 [cs.CG].

[8] Hideo Komuro. The diagonal flips of triangulations on the sphere. *Yokohama Mathematical Journal*, 44(2):115–122, 1997.

[9] Ryuichi Mori, Atsuhiro Nakamoto, and Katsuhiro Ota. Diagonal flips in Hamiltonian triangulations on the sphere. *Graphs and Combinatorics*, 19(3):413–418, 2003.

[10] Wacław Sierpiński. Sur une courbe dont tout point est un point de ramification. *Compte Rendus hebdomadaires des séance de l'Académie des Science de Paris*, 160:302–305, 1915.

[11] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Short encodings of evolving structures. *SIAM Journal on Discrete Mathematics*, 5(3):428–450, 1992.

[12] Hassler Whitney. A theorem on graphs. *Annals of Mathematics, Second Series*, 32(2):378–390, 1931.

7

EDGE-LABELLED FLIPS

The number of edge flips required to transform one triangulation into another has been studied extensively for unlabelled and vertex-labelled triangulations. In this chapter, we study this question for *edge-labelled triangulations*, in which every edge has a unique label that is carried over when the edge is flipped. Specifically, we prove that $O(n \log n)$ flips or $O(\log^2 n)$ simultaneous flips suffice to transform any combinatorial triangulation or triangulation of a convex n-gon into any other, and that $\Omega(n \log n)$ flips are sometimes required. For edge-labelled pseudo-triangulations, we also obtain a $\Theta(n \log n)$ bound, although the upper bound increases to $O(n^2)$ when we restrict ourselves to pointed pseudo-triangulations and exchanging flips.

The results on pseudo-triangulations have been accepted to the 27th Canadian Conference on Computational Geometry (CCCG 2015) [7]. This chapter is based on joint work with Prosenjit Bose, Anna Lubiw, and Vinayak Pathak.

7.1 INTRODUCTION

Flips have been studied in many different settings. While Wagner [22] originally studied them in the context of combinatorial triangulations (as surveyed in Chapter 5), interest in flips increased after Sleator, Tarjan and Thurston [19] showed a simple bijection between binary trees and triangulations of a convex polygon (subdivisions of the polygon into triangles using only diagonals), such that a flip in the triangulation corresponds to a rotation in the binary tree. This observation helped them in deriving a precise bound of $2n - 10$ flips on the diameter of the flip graph of an n-vertex convex polygon, and thereby on the maximal rotation distance between two binary trees on $n - 2$ nodes.

The same authors also studied flips in combinatorial triangulations with labelled vertices. Whereas in the unlabelled setting two triangulations are considered the same if they are isomorphic, here the isomorphism additionally needs to be consistent with the vertex labels. They proved an $O(n \log n)$ bound on the diameter of the flip graph in this setting [20]. Additionally, they presented a general framework for bounding the number of graphs reachable from an initial graph using simple transformations – including flips. Applying this framework to the vertex-labelled setting results in a matching $\Omega(n \log n)$ lower bound.

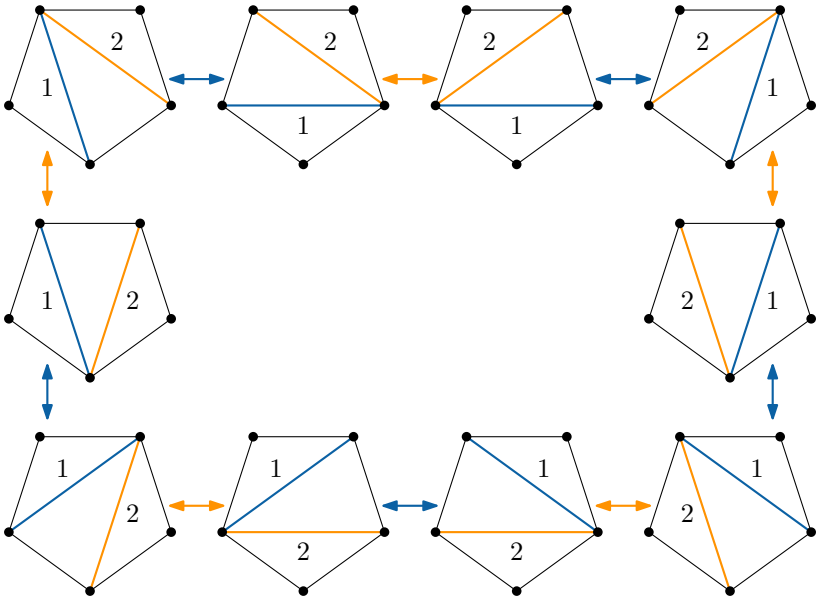


Figure 7.1: The flip graph of edge-labelled triangulations of a convex pentagon. The colour of the arrows corresponds to the colour of the flipped edge.

In this chapter, we consider a natural extension of the work by Sleator et al.: what happens when, instead of the vertices, the edges are labelled? A flip then reassigns the label of the flipped edge to the new edge, so that the set of labels does not change.

Edge-labelled flips in triangulations of a convex polygon have been studied independently by Araujo-Pardo et al. [4]. Their interest lies in the flip graph itself, which they call the *colorful associahedron* (see Figure 7.1 for an example). Although they establish that it is connected, their proof only gives a quadratic bound on the diameter. They then proceed to prove various structural properties that relate it to the flip graph in the unlabelled setting.

Edge labels have also been considered in different settings. Hernando et al. [12] investigated edge-labelled spanning trees of graphs. They showed that one can transform between any two edge-labelled spanning trees of a 2-connected graph by iteratively removing an edge and replacing it elsewhere with the same label while maintaining connectivity. The setting considered by Cano et al. [8] is different still; they transform between non-maximal plane graphs by 'rotating' edges around one of their endpoints. They prove that the corresponding edge rotation graph is connected, both in the labelled and unlabelled setting.

We start with the simplest setting, edge-labelled triangulations of an n-vertex convex polygon (Section 7.2), and show that this flip graph has a diameter of $\Theta(n \log n)$. We reuse Sleator et al.'s framework for the lower bound, but the proof for the upper bound is new. We then use this result to prove that the same bounds hold for edge-

labelled combinatorial triangulations (Section 7.3). As an aside, we consider what changes when we allow multiple edges to be flipped simultaneously, as long as they are not incident to the same triangle. In this setting, the upper bound reduces to $O(\log^2 n)$ both for convex polygons and combinatorial triangulations, but we no longer have a matching lower bound.

Finally, we consider edge-labelled pseudo-triangulations of point sets in the plane (Section 7.4). A *pseudo-triangulation* is a subdivision of the convex hull into *pseudo-triangles*: simple polygons with three convex interior angles. We first restrict ourselves to edge-labelled *pointed* pseudo-triangulations, which have the minimum number of edges. Here, we show that $O(n^2)$ *exchanging flips* (flips that replace one edge with another) suffice to transform between any two edge-labelled pointed pseudo-triangulations. If we additionally allow flips that only insert or remove an edge, we can transform any edge-labelled pseudo-triangulation into any other with $O(n \log c + h \log h)$ flips, where c is the number of convex layers of the point set and h is the number of points on the convex hull.

7.2 CONVEX POLYGONS

An *edge-labelled triangulation* of an n-vertex convex polygon is a triangulation of the polygon where each diagonal has a unique label in $\{1, \ldots, n-3\}$. In this section, we prove a tight $\Theta(n \log n)$ bound on the diameter of the flip graph of edge-labelled triangulations of a convex n-gon. We also show that the upper bound decreases to $O(\log^2 n)$ if we allow multiple edges, no two of which are incident on the same triangle, to be flipped simultaneously.

7.2.1 *Upper bound*

For the upper bound on the diameter of the flip graph, we show how to transform any edge-labelled triangulation into a canonical one in $O(n \log n)$ flips. Given two edge-labelled triangulations G_1 and G_2, the result then follows by composing this sequence for G_1 with the inverse sequence for G_2. First, consider the triangulation where all edges are incident to the vertex with the lowest y-coordinate. We call this configuration a *fan*. The canonical triangulation we use is a fan triangulation where the interior edges are labelled $1, \ldots, n-3$ in clockwise order around the bottom vertex (see Figure 7.2).

As we can transform any triangulation into a fan with $O(n)$ flips [10], the problem essentially reduces to sorting the labels of a fan. In this light, it is not surprising that our solution mimics a well-known sorting algorithm – quicksort – with a slight modification: instead of choosing a pivot at random, we always use the median. This guarantees that even in the worst case, we only use $O(n \log n)$

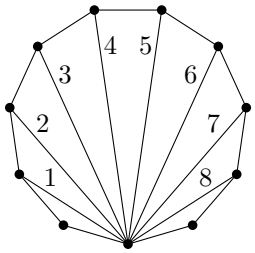


Figure 7.2: The canonical edge-labelled triangulation of a convex polygon with eleven vertices.

flips. We first show that we can use $2.5n$ flips to perform the 'partition' step of quicksort (ensuring that the first half of the edges of the fan are labelled with the first half of the labels and vice-versa). The flip sequence for this step is illustrated in Figure 7.3.

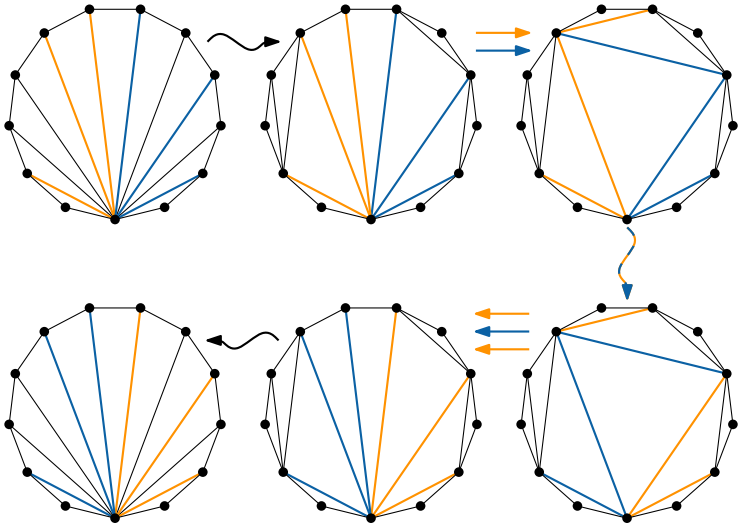


Figure 7.3: The flip sequence used in Lemma 7.1 to swap the labels of the high (lighter) and low (darker) edges.

LEMMA 7.1. *Let the diagonals of a fan triangulation of an n-vertex convex polygon be partitioned into three groups: low, neutral, and high, such that $|low| = |high|$ and all high edges occur to the left of any low edge. Then we can exchange the labels on the high and low edges with $2.5n$ flips, while leaving those on the neutral edges in place.*

Proof. We prove this by induction on n. In the base case ($n = 3$), there are no diagonals and we are done. So assume that $n > 3$ and that the lemma holds for any convex polygon with fewer than n vertices.

First, suppose that there is a neutral edge e. Flipping e makes it an ear of the current triangulation. Now the remaining diagonals are part of a fan triangulation of a convex polygon with $n - 1$ vertices that has e on the boundary. By induction, we can exchange the labels on the high and low edges in this polygon with $2.5(n - 1)$ flips.

Afterwards, we simply flip e back into place, giving a sequence of $2.5(n-1)+2 \leqslant 2.5n$ flips that successfully completes the swap.

Now suppose that there there are no neutral edges. Then the first half of the edges is high and the second half is low. Let e_1 and e_2 be the two edges in the middle, so that e_1 is high and e_2 is low. Then we can swap them with five flips, as shown in Figure 7.1. But note that two flips into this sequence, the two edges are out of the way – the remaining edges form a fan triangulation of a convex polygon with $n-2$ vertices. As we removed one low and one high edge, the two groups still have the same size in the smaller polygon. Thus, we can swap the labels of all other high and low edges with $2.5(n-2)$ flips by induction. Finally, we complete the swap of e_1 and e_2 with three more flips. This exchanges the labels of all high and low edges with a total of $2.5(n-2)+5 = 2.5n$ flips and concludes the proof. □

With this in place, we can sort all labels with $O(n \log n)$ flips by recursing on each half.

LEMMA 7.2. *Given an edge-labelled fan triangulation of a convex polygon with n vertices, we can sort the labels in ascending order around the bottom vertex with $O(n \log n)$ flips.*

Proof. The proof is by induction on n. In the base case $n = 3$ or $n = 4$, so the diagonals are sorted by default. Therefore assume that $n > 4$ and the lemma holds for all convex polygons with fewer than n vertices.

We identify two groups of edges. Let $m = \lfloor \frac{n-3}{2} \rfloor$ be the middle label. Then *high* edges have a label in $\{m+1, \dots, n-3\}$, but are among the m leftmost diagonals in the current fan. Conversely, *low* edges have a label in $\{1, \dots, m\}$, but are not among the m leftmost diagonals. To see that $|\text{high}| = |\text{low}|$, consider the m leftmost diagonals. By definition, these contain $m - |\text{low}|$ edges with a label in $\{1, \dots, m\}$. Thus, there must be $m - (m - |\text{low}|) = |\text{low}|$ edges among them with a label in $\{m+1, \dots, n-3\}$.

Therefore all conditions of Lemma 7.1 are satisfied, and we can swap the labels of the low and high edges with $2.5n$ flips. This ensures that the first m diagonals contain all labels from 1 through m, while the rightmost m or $m+1$ diagonals (depending on whether n is even or odd) contain all labels from $m+1$ through $n-3$. By induction, we can sort these two halves recursively, thereby sorting all labels. The total number of flips satisfies the recursion $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 2.5n$, which solves to $O(n \log n)$. □

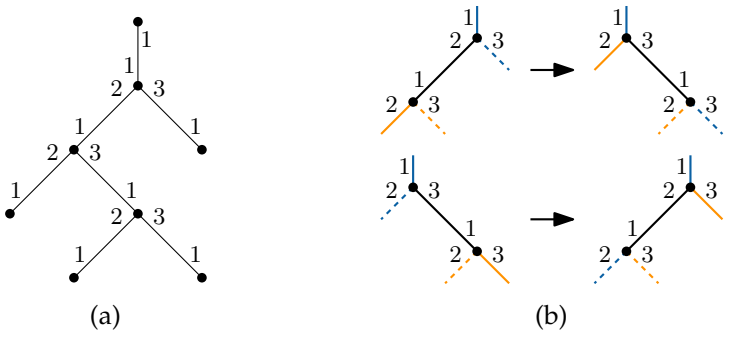
Since $O(n)$ flips suffice to transform any edge-labelled triangulation into a fan (by simply ignoring the labels), the upper bound on the diameter of the flip graph follows.

THEOREM 7.3. *Any edge-labelled triangulation of a convex polygon with n vertices can be transformed into any other by $O(n \log n)$ flips.*

7.2.2 *Lower bound*

The lower bound uses a slightly modified version of the $\Omega(n \log n)$ lower bound for the vertex-labelled setting by Sleator, Tarjan, and Thurston [20]. We first give an overview of their technique, before applying it to edge-labelled triangulations of a convex polygon.

Let a *tagged half-edge graph* be an undirected graph with maximum degree Δ, whose vertices have labels called *tags*, and whose edges are split into two *half-edges*. Each half-edge is incident to one endpoint, and labelled with an *edge-end label* in $\{1, \ldots, \Delta\}$, such that all edge-end labels incident on a vertex are distinct (see Figure 7.4a for an example). A *half-edge part* is a half-edge graph in which some half-edges do not have a twin. Note that tags are not restricted to integers: they could be tuples, or even arbitrary strings.



(a) (b)

Figure 7.4: (a) A half-edge graph representation of a rooted binary tree. (b) A graph grammar for rotations in binary trees. Correspondence between half-edges is indicated by a combination of colour and line style.

A *graph grammar* Γ is a sequence of production rules $\Gamma_i = (L, \to, \mathcal{T}, R)$, where L and R are half-edge parts with the same number of half-edges, $\to$ is a correspondence between the half-edges of L and R, and $\mathcal{T}$ is a function that computes the tags of vertices in R from those in L. A possible graph grammar for rotations in (unlabelled) binary trees is depicted in Figure 7.4b.

Sleator, Tarjan, and Thurston prove the following theorem.

THEOREM 7.4 (Sleator, Tarjan, and Thurston [20]). *Let G be a tagged half-edge graph of n vertices, Γ be a graph grammar, c be the number of vertices in left sides of Γ, and r be the maximum number of vertices in any right side of a production of Γ. Then $|R(G, \Gamma, m)| \leqslant (c+1)^{n+r \cdot m}$, where $R(G, \Gamma, m)$ is the set of graphs obtainable from G by derivations in Γ of length at most m.*

We cannot apply this theorem directly to triangulations of a convex polygon, as these do not have bounded degree. Instead, we turn to the dual graph. The *augmented dual graph* of a triangulation of a convex polygon is a tagged half-edge graph G with two sets of vertices:

triangle-vertices T corresponding to the triangles of the triangulation, and edge-vertices E_{CH} corresponding to the boundary edges. One edge-vertex is designated as the *root*.

Two triangle-vertices are connected by an edge if their triangles are adjacent. All edge-vertices are leaves, each connected to the triangle-vertex whose triangle is incident to their corresponding edge (see Figure 7.5). As every triangle has three edges, the maximum degree of G is three. The edge towards the root receives edge-end label 1. For a triangle-vertex, the other edge-end labels are assigned in counter-clockwise order, as in Figure 7.4a.

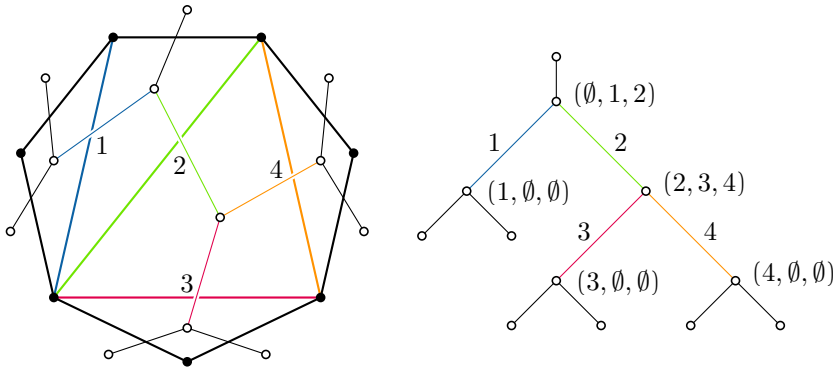


Figure 7.5: An edge-labelled triangulation of a convex polygon with its augmented dual graph. The edges-labels on the dual graph are shown to more clearly indicate the correspondence – they are actually labelled with edge-end labels as in Figure 7.4a.

This is where we deviate slightly from the original paper. Since Sleator, Tarjan, and Thurston were working in the vertex-labelled setting, they used the tags in the augmented dual graph to encode the labels of the vertices around the corresponding triangles. Instead, we use these tags to encode the edge-labels. Specifically, we tag each triangle-vertex with a triple containing the edge-label of each edge of its triangle, starting from the edge closest to the root, and proceeding in counter-clockwise order. Edges of the convex hull are assumed to have label $\emptyset$. Edge-vertices will not be involved in any of the production rules, so they do not need tags.

As flips in the triangulation correspond to rotations in the augmented dual graph [19], the graph grammar is identical to the graph grammar presented before. The only addition is the computation of new tags for the vertices on the right-hand side (see Figure 7.6). This grammar has four vertices in left sides, and a maximum of two vertices in any right side. Since a triangulation of an n-vertex convex polygon has $n-2$ triangles and n convex hull edges, the augmented dual graph has $2n-2$ vertices. Thus, Theorem 7.4 gives us the following.

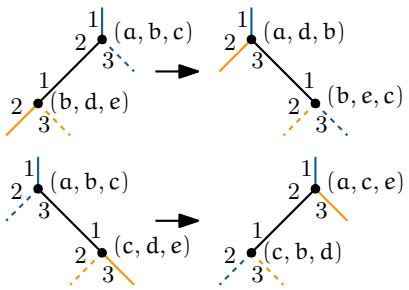


Figure 7.6: A graph grammar for rotations in augmented dual graphs, which correspond to flips in edge-labelled triangulations of a convex polygon.

LEMMA 7.5. *Given an edge-labelled triangulation* G *of an* n*-vertex convex polygon, the number of distinct edge-labelled triangulations reachable from* G *in* m *flips is at most* $5^{2n-2+2m}$.

This bound can be further refined to 3^{n-1+2m}, using the leader-follower and zero-elimination techniques from Sleator, Tarjan, and Thurton's paper [20]. However, the cruder bound already suffices to derive the correct asymptotic lower bound.

THEOREM 7.6. *There are pairs of edge-labelled triangulations of a convex polygon with* n *vertices such that transforming one into the other requires* $\Omega(n \log n)$ *flips.*

Proof. We first estimate the number of edge-labelled triangulations. An n-vertex convex polygon has $n-3$ diagonals, and in a fan triangulation, each sequence of labellings results in a new triangulation. Thus, there are at least $(n-3)!$ edge-labelled triangulations.

Let d be the diameter of the flip graph. Then, for every graph G, d flips suffice to reach all edge-labelled triangulations. But from Lemma 7.5, we know that a sequence of m flips can generate at most $5^{2n-2+2m}$ unique edge-labelled triangulations. This gives us the following bound.

$$\begin{aligned}
5^{2n-2+2d} &\geqslant (n-3)! \\
\log_5 5^{2n-2+2d} &\geqslant \log_5 (n-3)! \\
2n-2+2d &\geqslant \log_5 (n!/n^3) \\
2d &\geqslant \log_5 n! - \log_5 n^3 - 2n + 2 \\
2d &\geqslant \Omega(n \log n) - O(n) \\
d &\geqslant \Omega(n \log n)
\end{aligned}$$

□

Combining the upper bound from Theorem 7.3 with the lower bound from Theorem 7.6 gives us an asymptotically tight bound on the worst-case number of flips required to transform one edge-labelled triangulation of a convex polygon into another.

COROLLARY 7.7. *The flip graph of edge-labelled triangulations of a convex polygon with* n *vertices has diameter* $\Theta(n \log n)$.

7.2.3 *Simultaneous flips*

A *simultaneous flip* is a transformation that consists of one or more regular flips that are executed at the same time. For this definition to make sense, it is important that two of these flips do not interfere with each other. Therefore we add the restriction that in a single simultaneous flip, at most one edge of each triangle can be flipped. In other words, no two edges in the same simultaneous flip can be incident to the same triangle. Galtier et al. [11] showed that $O(\log n)$ simultaneous flips suffice to transform any triangulation of an n-vertex convex polygon into any other.

The bounds on simultaneous flips are clearly connected to those on regular flips, as each simultaneous flip can group only $O(n)$ regular flips. This means that the $\Omega(n \log n)$ lower bound from Theorem 7.6 also implies an $\Omega(\log n)$ lower bound for simultaneous flips. This does not apply to the upper bound, however. For example, it is not possible, in general, to simply perform each $c \cdot n$ flips simultaneously (for some constant c), as two such flips can easily be incident on the same triangle. The upper bounds do translate in the other direction: proving that $O(k)$ simultaneous flips suffice immediately implies that $O(kn)$ regular flips suffice, simply by performing each set of flips in sequence.

In this section, we show that any edge-labelled triangulation of a convex polygon with n vertices can be transformed into any other with $O(\log^2 n)$ simultaneous flips. We show that even the partition step of quicksort already requires $\Omega(\log n)$ simultaneous flips. We start by proving an analogue to Lemma 7.1, showing that this bound on the partition step is tight.

LEMMA 7.8. *Let the diagonals of a fan triangulation of an n-vertex convex polygon be partitioned into three groups: low, neutral, and high, such that $|low| = |high|$ and all high edges occur to the left of any low edge. Then we can exchange the labels on the high and low edges with $O(\log n)$ simultaneous flips, while leaving those on the neutral edges in place.*

Proof. We first flip each neutral diagonal, so that they are no longer incident to the bottom vertex. Since we can flip every second edge in a single simultaneous flip, $O(\log n)$ simultaneous flips suffice to flip all neutral diagonals. This leaves only the high and low diagonals, inside a smaller convex polygon formed by the original polygon and the neutral edges.

This reduces our problem to transforming a convex polygon where the first half of the diagonals is high and the second is low to one where these sets are reversed. We do this by transforming a third configuration, called an *alternating zig-zag*, into both, with $O(\log n)$ simultaneous flips. The result then follows by the reversibility of flips.

Let r be a ray from the bottom vertex that has $\lfloor n/2 \rfloor$ of the remaining vertices to its left. Let a_i be the vertex to the left of r at distance i

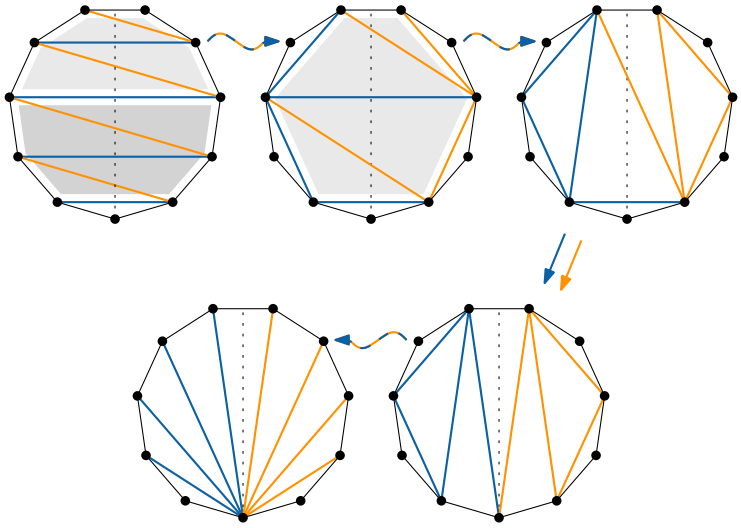


Figure 7.7: A sequence of simultaneous flips that transforms an alternating zig-zag into a fan partition. We can choose which group of edges ends up to the left of the ray. The hexagons involved in the first two steps are lightly shaded.

from the bottom vertex (along the boundary of the polygon), and let b_i be the analogous vertex to the right of r. The alternating zig-zag contains the edges a_1b_1, b_1a_2, a_2b_2, etc. Each edge a_ib_i is low, and each edge b_ia_{i+1} is high (see Figure 7.7).

To transform the alternating zig-zag into the fan triangulation with all low edges on the left, we first partition it into hexagons. Each hexagon is formed by the edges a_ib_i and $a_{i+2}b_{i+2}$, along with the boundary edges between them. In this way, each hexagon contains three diagonals: two high diagonals b_ia_{i+1} and $b_{i+1}a_{i+2}$, and one low diagonal $a_{i+1}b_{i+1}$. We now use a constant number of simultaneous flips to transform the triangulation within each hexagon to have the low diagonal at a_ia_{i+2}, and the high diagonals at b_ib_{i+2} and b_ia_{i+2}. Since the hexagons are separated by low diagonals, none of these simultaneous flips interfere with each other.

Now consider the edges that intersect r in the resulting triangulation. These are the low edges separating the hexagons, and one high diagonal inside each hexagon. These edges form another alternating zig-zag, half the size of the first one. Therefore we can repeat this procedure until less than three edges intersect r, halving the size each time. This requires $O(\log n)$ simultaneous flips. Once few edges intersect r, a constant number of simultaneous flips suffice to properly partition these, giving a triangulation where all low edges are to the left of r. Similarly, by consistently moving the low edge in each hexagon to the right of r, we obtain a sequence of $O(\log n)$ simultaneous flips that constructs a triangulation where all low edges are to the right of r.

Now that we have a triangulation where all low edges are on one side of r and all high edges are on the other, all that is left is to change this into a fan triangulation. Since each group forms a triangulation of a smaller convex polygon, we can use the result by Galtier et al. [11] to transform these two parts into a fan triangulation with $O(\log n)$ simultaneous flips. This completes the proof. □

As in the non-simultaneous upper bound, this allows us to simulate a deterministic version of quicksort (that always picks the median as pivot) to sort the labels of a fan, giving the following upper bound.

THEOREM 7.9. *Any edge-labelled triangulation of a convex polygon with n vertices can be transformed into any other by $O(\log^2 n)$ simultaneous flips.*

Proof. We first ignore the edge labels and transform both triangulations into a fan triangulation with $O(\log n)$ simultaneous flips [11]. Next, we show how to sort the labels of a fan. The result follows by the reversibility of flips.

To sort, we partition the edges into high, neutral, and low edges as in the proof of Lemma 7.2, and use Lemma 7.8 to exchange the edges with low labels with those with high labels with $O(\log n)$ flips. Now we can sort each half recursively, combining the simultaneous flips in each part into one larger simultaneous flip. Thus, the total number of simultaneous flips is given by the recurrence $T(n) = T(\lfloor n/2 \rfloor) + O(\log n)$, which comes out to $O(\log^2 n)$ flips.

Of course, for the combined simultaneous flip to be valid, we must ensure that no two flips use an edge of the same triangle. We do this by finding the edge with the median label and placing it on the right edge. This is a special case of Lemma 7.8, with one high and low edge, and all others neutral, so we can do it with $O(\log n)$ simultaneous flips. By recursing to the left and right of this fixed edge, we can guarantee that the simultaneous flips on each side do not interfere, proving the theorem. □

Unfortunately, we do not have a matching lower bound in this case. In fact, the best asymptotic lower bound we have is that $\Omega(\log n)$ simultaneous flips are sometimes required – identical to the unlabelled setting! This bound is easily derived from Theorem 7.6, or directly, by observing that there are triangulations with constant maximum degree, and every simultaneous flip can at most double the degree of a vertex. What we *can* prove is that this lower bound holds already for the partition step. This means that at least the result of Lemma 7.8 is best possible.

THEOREM 7.10. *Let the first half of the diagonals of a fan triangulation of an n-vertex convex polygon be* high, *and the rest* low. *Then any sequence of simultaneous flips that exchanges the labels on the high and low edges must have length $\Omega(\log n)$.*

Proof. Let r be a ray from the bottom vertex that separates the high edges from the low ones, and let $\mathcal{F}$ be a sequence of simultaneous flips that exchanges the high and low edges. Consider an arbitrary label ℓ. Before executing $\mathcal{F}$, ℓ is on one side of r and afterwards it is on the other side. But since an edge that lies completely on one side of r cannot intersect an edge that lies completely on the other side, and a flip always transforms an edge into another that intersects it, we cannot flip directly from the first edge to the second. In particular, at some point during the flip sequence, the edge with label ℓ must intersect r. This holds for all labels.

Now let X_i be the number of distinct labels that have intersected r after i simultaneous flips. Clearly $X_0 = 0$, and by the argument above $X_{|\mathcal{F}|} = n-3$ (the total number of labels). Consider a flip that creates a new edge intersecting r. This flip takes place in a quadrilateral that itself intersects r. Therefore two of its boundary edges must intersect r as well. Since a boundary edge can be shared by at most two quadrilaterals, and each quadrilateral has two boundary edges that intersect r, a simultaneous flip that creates k new crossing must already have had at least k crossings. In other words, $X_{i+1} \leqslant 2X_i$. Since $X_{|\mathcal{F}|} = n-3$, this implies that $|\mathcal{F}| \geqslant \log_2(n-3) = \Omega(\log n)$, proving the lemma. □

7.3 COMBINATORIAL TRIANGULATIONS

In this section, we show that any edge-labelled combinatorial triangulation can be transformed into any other with $O(n \log n)$ flips, and that this bound is tight. Note that we consider two edge-labelled triangulations to be equivalent if they have an isomorphism that preserves the edge labels.

7.3.1 *Upper bound*

For the upper bound, we use a canonical triangulation much like the one used by Sleator, Tarjan, and Thurston [20] 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 7.8). For our canonical labelling, we separate the labels into three groups. We call labels $1, \ldots, n-2$ group $\mathcal{S}$ and we place them on the spine edges, starting with the edge on the outer face and continuing in clockwise order around v_{in}. The next $n-2$ labels make up group $\mathcal{C}_{in}$ and are placed on the edges incident to v_{in} in clockwise order, starting with the edge incident to the vertex shared by the edges with labels 1 and 2. Finally, group $\mathcal{C}_{out}$ consists of the last $n-2$ labels, which we place on the edges incident to v_{out} in clockwise order, starting with the edge that shares a vertex with the edge labelled $2n-4$.

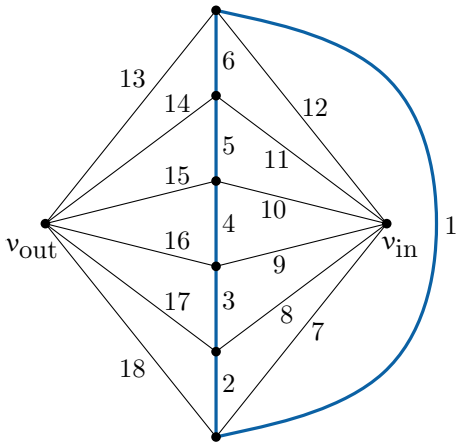


Figure 7.8: The canonical edge-labelled combinatorial triangulation on 8 vertices. The spine is indicated in bold.

THEOREM 7.11. *Any edge-labelled combinatorial triangulation with n vertices can be transformed into any other by $O(n \log n)$ flips.*

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

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 [20] and results in the correct graph, although the labels may be in arbitrary positions. To fix the labels, we first get the groups to have the correct set of labels, that is, all labels in group S are on the spine, etc., before we rearrange the labels within each group.

We use two main tools for this. The first is a *swap* that interchanges one spine edge with an incident non-spine edge in seven flips, using the flip sequence depicted in Figure 7.9. Our second tool is a *scramble* algorithm that reorders all labels incident to v_{in} or v_{out} 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 7.8) and then apply the algorithm from Theorem 7.3 to the outerplanar graph induced by the spine plus v_{in} (or v_{out}), observing that no flip will create a duplicate edge since the omitted edges are all incident to v_{out} (resp. v_{in}). Note that this method cannot alter the labels on the two non-spine edges that lie on the exterior face of the outerplanar graph (labelled 7 and 12 in Figure 7.8), 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.

To get the labels of group S on the spine, we partner every edge incident to v_{in} that has a label in S with an edge on the spine that has a label in C_{in} or C_{out}. A scramble at v_{in} makes each such edge incident to its partner, and then swaps exchange partners. By doing the same

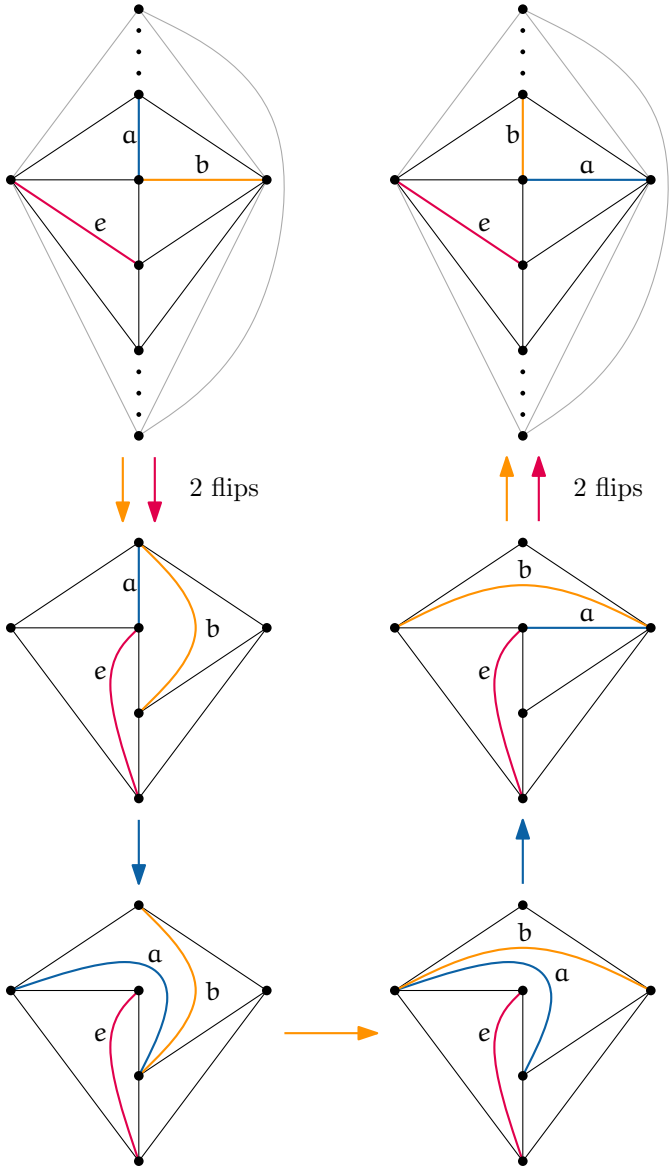


Figure 7.9: A sequence of seven flips that swaps two edges a and b that are consecutive around a vertex on the spine. Although edge e ends up at the same place as at the start of the sequence, it essentially acts as a catalyst here. If we did not flip it, we would not be able to flip edge a after edge b, as that would create a duplicate edge.

at v_{out}, all labels of $\mathcal{S}$ are placed on the spine. Next we partner every edge incident to v_{in} that has a label in $\mathcal{C}_{out}$ with an edge incident to v_{out} that has a label in $\mathcal{C}_{in}$. A scramble at v_{in} makes partners incident, and three swaps per pair then exchange partners.

This ensures that each edge's label is in the correct group, but the order of the labels within each group may still be incorrect. Rearranging the labels in $\mathcal{C}_{in}$ and $\mathcal{C}_{out}$ is straightforward, as we can simply scramble at v_{in} and v_{out}, leaving only the labels on the spine out of order. We then 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 scramble does not affect the order of labels on the spine, we can simply exchange the edges once more to obtain the canonical triangulation. □

7.3.2 *Lower bound*

The proof for the lower bound for combinatorial triangulations is very similar to the lower bound for triangulations of a convex polygon, described in Section 7.2.2. We again construct a graph grammar, which describes transformations on the dual graph that correspond to flips.

As our primary graph is a combinatorial triangulation, each vertex of the dual graph corresponds to a triangle and has degree three. As such, there is no distinction between internal nodes and leaves, and no root. This means that we need to adapt our definitions slightly. Without a root, the placement of the edge-end labels is less constrained. We only require that they occur in counter-clockwise order around each vertex. The order of labels in each tag can now follow the placement of the edge-end labels: the first label belongs to the primary edge corresponding to the dual edge with edge-end label 1, and so on.

Finally, we need a few more production rule to deal with all possible rotations of the edge-end labels around the two triangle-vertices involved in the flip. The full collection of rules is shown in Figure 7.10.

As the dual graph of an n-vertex combinatorial triangulation has $2n-4$ vertices, Theorem 7.4 gives us the following bound.

LEMMA 7.12. *Given an n-vertex edge-labelled combinatorial triangulation G, the number of distinct edge-labelled triangulations reachable from G in m flips is at most $13^{2n-4+2m}$.*

Again, Sleator, Tarjan, and Thurston [20] show that this bound can be significantly reduced (to $3^{2n-4}8^m$), but the simple bound suffices for our purposes.

THEOREM 7.13. *There are pairs of edge-labelled combinatorial triangulations with n vertices such that transforming one into the other requires $\Omega(n \log n)$ flips.*

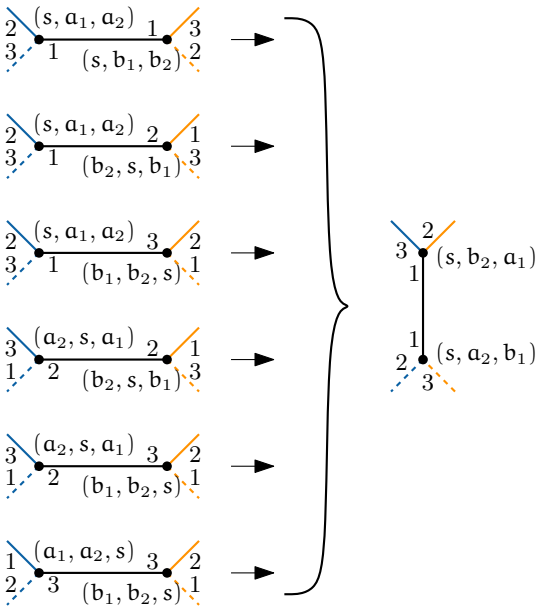


Figure 7.10: A graph grammar that corresponds to flips in edge-labelled combinatorial triangulations. The right-hand side of all productions is the same.

Proof. If we fix the labelling of the spine edges in the canonical triangulation from the proof of Theorem 7.11, any relabelling of the remaining edges is unique. Thus, there are at least $(2n-6)!$ distinct edge-labelled combinatorial triangulations. Combined with Lemma 7.12, this implies that $13^{2n-4+2d} \geqslant (2n-6)!$, where d is the diameter of the flip graph. We derive the following.

$$\begin{aligned}
13^{2n-4+2d} &\geqslant (2n-6)! \\
13^{2n-4+2d} &\geqslant n! && \text{(for } n \geqslant 5\text{)} \\
\log_{13} 13^{2n-4+2d} &\geqslant \log_{13} n! \\
2n-4+2d &\geqslant \log_{13} n! \\
2d &\geqslant \log_{13} n! - 2n + 4 \\
2d &\geqslant \Omega(n \log n) - O(n) \\
d &\geqslant \Omega(n \log n)
\end{aligned}$$

□

Combining the upper and lower bound from Theorem 7.11 and 7.13 yields the following.

COROLLARY 7.14. *The flip graph of edge-labelled combinatorial triangulations with n vertices has diameter $\Theta(n \log n)$.*

7.3.3 *Simultaneous flips*

Recall that, in a triangulation of a convex polygon, a simultaneous flip is a set of flips that are executed in parallel, such that no two flipped edges share a triangle. In a combinatorial triangulation, we

have the additional requirement that the resulting graph may not contain duplicate edges.

Simultaneous flips in combinatorial triangulations were first studied by Bose et al. [6]. They showed a tight $\Theta(\log n)$ bound on the diameter of the flip graph. As part of their proof, they showed that every combinatorial triangulation can be made 4-connected with a single simultaneous flip. Recently, Cardinal et al. [9] proved that it is possible to find such a simultaneous flip that consists of fewer than $2n/3$ individual flips. They used this result to obtain arc drawings of planar graphs in which only $2n/3$ edges are represented by multiple arcs.

In this section, we show that, just as in the non-simultaneous setting, we obtain the same bounds for edge-labelled convex polygons and edge-labelled combinatorial triangulations. That is, we can transform any edge-labelled combinatorial triangulation into any other with $O(\log^2 n)$ simultaneous flips, and $\Omega(\log n)$ simultaneous flips are sometimes necessary. The lower bound holds already in the unlabelled setting, if one vertex has linear degree in the first triangulation, while every vertex has constant degree in the second. We now prove the upper bound.

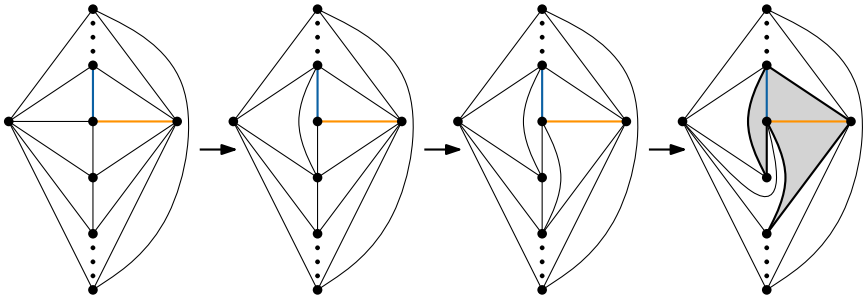


Figure 7.11: A sequence of three flips that creates a pentagon (shaded) in which the two highlighted edges can be swapped. All new edges and all diagonals of the pentagon are incident to one of the four spine vertices shown.

THEOREM 7.15. *Any edge-labelled combinatorial triangulation with n vertices can be transformed into any other by $O(\log^2 n)$ simultaneous flips.*

Proof. We closely follow the strategy of the proof of Theorem 7.11. We first transform the given triangulations into the canonical one with $O(\log n)$ simultaneous flips, using the result of Bose et al. [6]. This reduces the problem to sorting the edge labels on the canonical triangulation. In the non-simultaneous setting, we did this by reordering the labels on the edges incident to v_{in} or v_{out} (called *scrambling*), and swapping a subset of spine edges with incident non-spine edges. Thus, the theorem follows if we can show how to perform these operations with $O(\log^2 n)$ simultaneous flips.

Since the sequence of flips from Figure 7.9, which swaps a single spine edge with an incident non-spine edge, only involves a constant number of triangles, it is tempting to think we can simply perform many of these swaps simultaneously. Unfortunately, this is not the case, since the sequence creates the edge $(v_{\text{out}}, v_{\text{in}})$. This means that trying to perform this sequence simultaneously in different locations would create a duplicate edge. Therefore we use a slightly longer sequence that creates a pentagon containing the edges to be swapped (illustrated in Figure 7.11), performs the swap inside this pentagon, and restores the canonical triangulation, using a total of eleven flips. The crucial property of this sequence is that it only creates edges incident to four spine vertices near the edge to be swapped. Thus, we can perform any number of swaps simultaneously without creating duplicate edges, as long as each swap is at distance four or more from the others. This means that, given a set of spine edges to swap, we can divide them into four rounds such that the edges to be swapped in each round are at distance four or more, and perform the swaps in each round simultaneously. Thus, we can swap any subset of spine edges with $O(1)$ simultaneous flips.

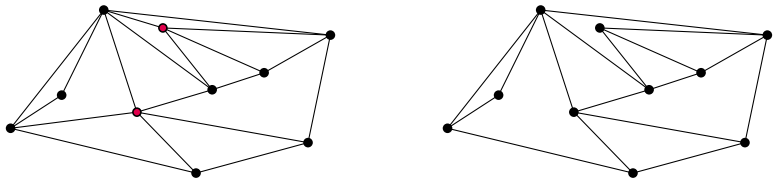
To scramble the edges incident on v_{out}, we first flip to create $(v_{\text{out}}, v_{\text{in}})$ and then apply the algorithm from Theorem 7.9 to the outerplanar graph induced by the edges incident to v_{out}. This uses $O(\log^2 n)$ simultaneous flips to rearrange all labels, except for those on the two outermost edges that are part of the boundary. In the non-simultaneous setting, we fixed this by swapping these labels along the spine, but this would take too many flips here. Instead, if the labels that need to be on the outermost edges are in the interior, we use Theorem 7.9 to place these labels on the interior edges closest to the outermost edges. Then, we can exchange them with the labels on the outermost edges with only three swaps. This ensures that the outermost edges have the correct labels, so a second application of Theorem 7.9 can place the remaining labels in the right order. If the label for one of the outermost edges is not in the interior and not already in place, it must be on the other outermost edge. In this case, we can first exchange it with the label on a nearby interior edge with a constant number of swaps. The entire sequence requires $O(\log^2 n)$ simultaneous flips.

Since these operations use $O(1)$ and $O(\log^2 n)$ simultaneous flips, and we can sort the labels with a constant number of applications, the theorem follows. □

7.4 PSEUDO-TRIANGULATIONS

A *pseudo-triangle* is a simple polygon with three convex interior angles, called *corners*, that are connected by reflex chains. Given a set P of n points in the plane, a *pseudo-triangulation* of P is a subdivision of

its convex hull into pseudo-triangles, using all points of P as vertices (see Figure 7.12a). A pseudo-triangulation is *pointed* if all vertices are incident to a reflex angle in some face (including the outer face; see Figure 7.12b for an example). Pseudo-triangulations find applications in areas such as kinetic data structures [14] and rigidity theory [21]. More information on pseudo-triangulations can be found in a survey by Rote, Santos, and Streinu [18].

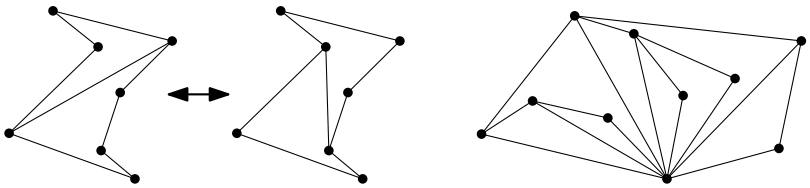


(a) (b)

Figure 7.12: (a) A pseudo-triangulation with two non-pointed vertices. (b) A pointed pseudo-triangulation.

Since a regular triangle is also a pseudo-triangle, pseudo-triangulations generalize triangulations (subdivisions of the convex hull into triangles). In a triangulation, a flip is a local transformation that removes one edge, leaving an empty quadrilateral, and inserts the other diagonal of that quadrilateral. Note that this is only possible if the quadrilateral is convex. Lawson [15] showed that any triangulation with n vertices can be transformed into any other with $O(n^2)$ flips, and Hurtado, Noy, and Urrutia [13] gave a matching $\Omega(n^2)$ lower bound.

Pointed pseudo-triangulations support a similar type of flip, but before we can introduce this, we need to generalize the concept of pseudo-triangles to *pseudo-k-gons*: weakly simple polygons with k convex interior angles. A diagonal of a pseudo-k-gon is called a *bitangent* if the pseudo-k-gon remains pointed after insertion of the diagonal. In a pointed pseudo-triangulation, *flipping* an edge removes the edge, leaving a pseudo-quadrilateral, and inserts the unique other bitangent of the pseudo-quadrilateral (see Figure 7.13a). In contrast with triangulations, all internal edges of a pointed pseudo-triangulation are flippable. Bereg [5] showed that $O(n \log n)$ flips suffice to transform any pseudo-triangulation into any other.



(a) (b)

Figure 7.13: (a) A flip in a pseudo-quadrilateral. (b) A left-shelling pseudo-triangulation.

Aichholzer et al. [2] showed that the same result holds for all pseudo-triangulations (including triangulations) if we allow two more types of flips: *insertion* and *deletion* flips. As the name implies, these either insert or delete one edge, provided that the result is still a pseudo-triangulation. To disambiguate, they call the other flips *exchanging* flips. In a later paper, this bound was refined to $O(n \log c)$ [1], where c is the number of convex layers of the point set.

In this section, we investigate flips in *edge-labelled pseudo-triangulations*: pseudo-triangulations where each internal edge has a unique label in $\{1, \ldots, 3n - 3 - 2h\}$, where h is the number of vertices on the convex hull ($3n - 3 - 2h$ is the number of internal edges in a triangulation). In the case of an exchanging flip, the new edge receives the label of the old edge. For a deletion flip, the edge and its label are simply removed, and for an insertion flip, the new edge receives an unused label from the set of all possible labels.

Our results are the following: using only exchanging flips, we show that $O(n^2)$ flips suffice to transform any edge-labelled pointed pseudo-triangulation into any other with the same set of labels. By using insertion, deletion and exchanging flips, we can transform any edge-labelled pseudo-triangulation into any other with $O(n \log c + h \log h)$ flips.

Before we can start the proof, we need a few more definitions. Given a set of points in the plane, let v_0 be the point with the lowest y-coordinate, and let $v_1, \ldots, v_n$ be the other points in clockwise order around v_0. The *left-shelling* pseudo-triangulation is the union of the convex hulls of $v_0, \ldots, v_i$, for all $2 \leqslant i \leqslant n$ (see Figure 7.13b). Thus, every vertex after v_1 is associated with two edges: a *bottom* edge connecting it to v_0 and a *top* edge that is tangent to the convex hull of the earlier vertices. The *right-shelling* pseudo-triangulation is similar, with the vertices added in counter-clockwise order instead.

7.4.1 *Pointed pseudo-triangulations*

In this section, we show that every edge-labelled pointed pseudo-triangulation can be transformed into any other with the same set of labels by $O(n^2)$ exchanging flips. We do this by showing how to transform a given edge-labelled pointed pseudo-triangulation into a *canonical* one. The result then follows by the reversibility of flips. As canonical pseudo-triangulation, we use the left-shelling pseudo-triangulation, with the bottom edges labelled in clockwise order around v_0, followed by the internal top edges in the same order (based on their associated vertex).

Since we can transform any pointed pseudo-triangulation into the left-shelling pseudo-triangulation with $O(n \log n)$ flips [5], the main part of the proof lies in reordering the labels of a left-shelling pseudo-triangulation. We use two tools for this, called a *sweep* and a *shuffle*,

that are implemented by a sequence of flips. A sweep interchanges the labels of some internal top edges with their respective bottom edges, while a shuffle permutes the labels on all bottom edges.

LEMMA 7.16. *We can transform any left-shelling pseudo-triangulation into the canonical one with* $O(1)$ *shuffle and sweep operations.*

Proof. In the canonical pseudo-triangulation, we call the labels assigned to bottom edges *low*, and the labels assigned to top edges *high*. In the first step, we use a shuffle to line up every bottom edge with a high label with a top edge with a low label. Then we exchange these pairs of labels with a sweep. Now all bottom edges have low labels and all top edges have high labels, so all that is left is to sort the labels. We can sort the low labels with a second shuffle. To sort the high labels, we sweep them to the bottom edges, shuffle to sort them there, then sweep them back. □

The remainder of this section describes how to perform a sweep and a shuffle with flips.

LEMMA 7.17. *We can interchange the labels of the edges incident to an internal vertex* v *of degree two with three exchanging flips.*

Proof. Consider what happens when we remove v. Deleting one of its edges leaves a pseudo-quadrilateral. Removing the second edge then either merges two corners into one, or removes one corner, leaving a pseudo-triangle T. There are three bitangents that connect v to T, each corresponding to the geodesic between v and a corner of T. Any choice of two of these bitangents results in a pointed pseudo-triangulation. When one of them is flipped, the only new edge that can be inserted so that the result is still a pointed pseudo-triangulation is the bitangent that was not there before the flip. Thus, we can interchange the labels with three flips (see Figure 7.14). □

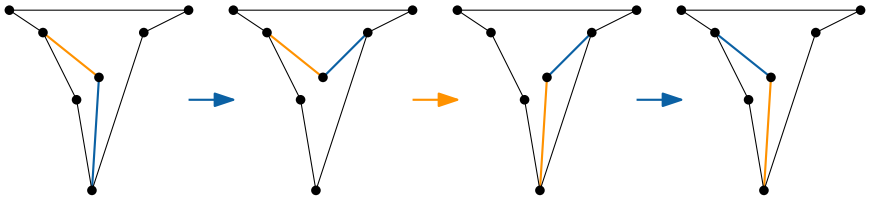


Figure 7.14: Interchanging the labels of the edges incident to a vertex of degree two.

LEMMA 7.18 (Sweep). *In the left-shelling pseudo-triangulation, we can interchange the labels of any number of internal top edges and their corresponding bottom edges with* $O(n)$ *exchanging flips.*

Proof. Let S be the set of vertices whose internal top edge should have its label swapped with the corresponding bottom edge. Consider a ray L from v_0 that starts at the positive x-axis and sweeps through the point set to the negative x-axis. We will maintain the following invariant: the graph induced by the vertices to the left of L is their left-shelling pseudo-triangulation and the graph induced by the vertices to the right of L is their right-shelling pseudo-triangulation (both groups include v_0). Furthermore, the labels of the top edges of the vertices in S to the right of L have been interchanged with their respective bottom edges. This invariant is satisfied at the start.

Suppose that L is about to pass a vertex v_k. If v_k is on the convex hull, its top edge is not internal and no action is required for the invariant to hold after passing v_k. So assume that v_k is not on the convex hull and consider its incident edges. It is currently part of the left-shelling pseudo-triangulation of points to the left of L, where it is the last vertex. Thus, v_k is connected to v_0 and to one vertex to its left. It is not connected to any vertex to its right, since there are $2n-3$ edges in total, and the left- and right-shelling pseudo-triangulations to each side of L contribute $2(k+1)-3+2(n-k)-3 = 2n-4$ edges. So the only edge that crosses L is an edge of the convex hull. Therefore v_k has degree two, which means that we can use Lemma 7.17 to swap the labels of its top and bottom edge with three flips if $v_k \in S$.

Furthermore, the sides of the pseudo-triangle that remains if we were to remove v_k, form part of the convex hull of the points to either side of L. Thus, flipping the top edge of v_k results in the tangent from v_k to the convex hull of the points to the right of L – exactly the edge needed to add v_k to their right-shelling pseudo-triangulation. Therefore we only need $O(1)$ flips to maintain the invariant when passing v_k.

At the end, we have constructed the right-shelling pseudo-triangulation and swapped the desired edges. An analogous transformation without any swapping can transform the graph back into the left-shelling pseudo-triangulation with $O(n)$ flips in total. □

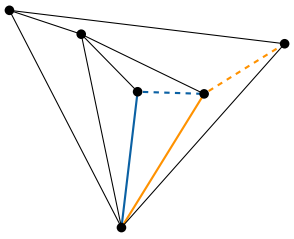


Figure 7.15: A pseudo-pentagon with four bitangents. It is impossible to swap the two diagonals without flipping an edge of the pseudo-pentagon, as they just flip back and forth between the solid bitangents and the dotted ones, regardless of the position of the other diagonal.

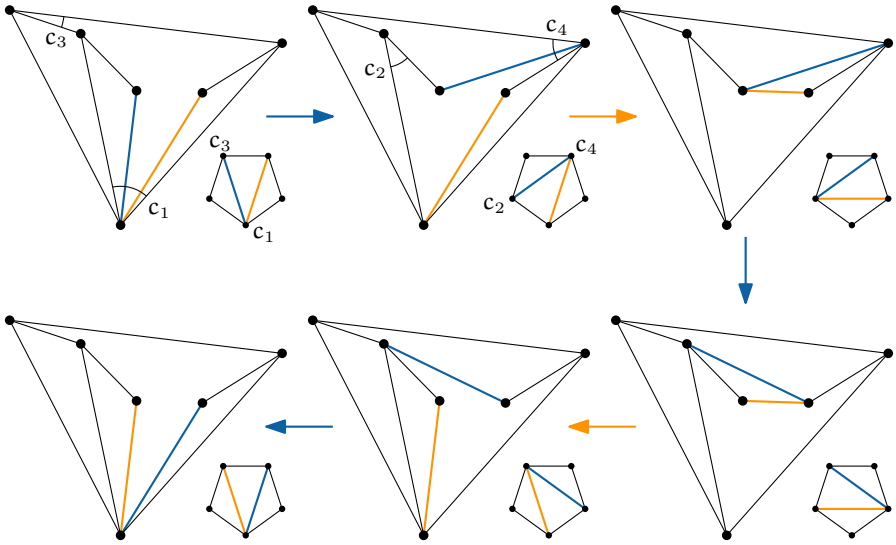


Figure 7.16: Interchanging the labels of two bitangents of a pseudo-pentagon with five bitangents. An edge in the pentagon corresponds to a geodesic between two corners of the pseudo-pentagon.

LEMMA 7.19. *In the left-shelling pseudo-triangulation, we can interchange the labels of two consecutive bottom edges with* $O(1)$ *exchanging flips.*

Proof. When we remove the two consecutive bottom edges (say a and b), we are left with a pseudo-pentagon X. A pseudo-pentagon can have up to five bitangents, as each bitangent corresponds to a geodesic between two corners. If X has exactly five bitangents, this correspondence is a bijection. This implies that the bitangents of X can be swapped just like diagonals of a convex pentagon (see Figure 7.16). On the other hand, if X has only four bitangents, it is impossible to swap a and b without flipping an edge of X (see Figure 7.15).

Fortunately, we can always transform X into a pseudo-pentagon with five bitangents. If the pseudo-triangle to the right of b is a triangle, X already has five bitangents (see Lemma 7.24 in Section 7.4.1.1). Otherwise, the top endpoint of b is an internal vertex of degree two and we can flip its top edge to obtain a new pseudo-pentagon that does have five bitangents (see Lemma 7.25 in Section 7.4.1.1). After swapping the labels of a and b, we can flip this top edge back. Thus, in either case we can interchange the labels of a and b with $O(1)$ flips. □

We can use Lemma 7.19 to reorder the labels of the bottom edges with insertion or bubble sort, as these algorithms only swap adjacent values.

COROLLARY 7.20 (Shuffle). *In the left-shelling pseudo-triangulation, we can reorder the labels of all bottom edges with* $O(n^2)$ *exchanging flips.*

Combining this with Lemmas 7.16 and 7.18, and the fact that we can transform any pointed pseudo-triangulation into the left-shelling one with $O(n \log n)$ flips [5], gives the main result.

THEOREM 7.21. *We can transform any edge-labelled pointed pseudo-triangulation with n vertices into any other with $O(n^2)$ exchanging flips.*

The following lower bound follows from the $\Omega(n \log n)$ lower bound on the flip distance between edge-labelled triangulations of a convex polygon (Theorem 7.6).

THEOREM 7.22. *There are pairs of edge-labelled pointed pseudo-triangulations with n vertices that require $\Omega(n \log n)$ exchanging flips to transform one into the other.*

7.4.1.1 *Deferred proofs*

This section contains a few technical lemmas that were omitted from the previous section.

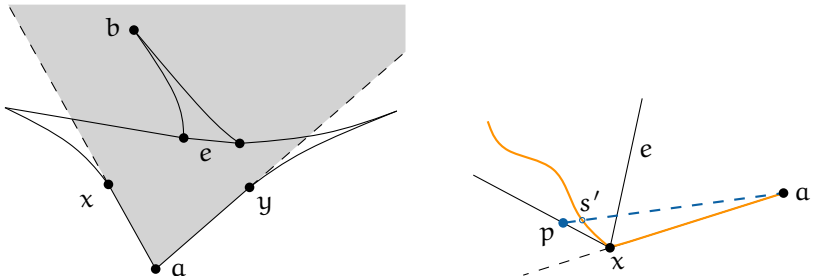


Figure 7.17: (a) A corner of a pseudo-triangle and an edge such that the entire pseudo-triangle on the other side of the edge lies inside the corner's wedge. (b) If a can see a point past x, then the geodesic does not contain x.

LEMMA 7.23. *Let a be a corner of a pseudo-triangle with neighbours x and y, and let e be an edge on the chain opposite a. If all vertices of the other pseudo-triangle containing e lie in the wedge formed by extending the edges ax and ay into half-lines (see Figure 7.17a), then flipping e will result in an edge incident on a.*

Proof. Let T be the pseudo-triangle on the other side of e, and let b be the corner of T opposite e. Then flipping e inserts the geodesic between a and b. This geodesic must intersect e in a point s and then follow the shortest path from s to a. If s lies strictly inside the wedge, nothing can block as, thus the new edge will contain as and be incident on a.

Now, if all of e lies strictly inside the wedge, our result follows. But suppose that e has x as an endpoint and the geodesic between a and b intersects e in x. As a can see x and all of T lies inside the wedge,

there is an $\varepsilon > 0$ such that a can see the point X on the boundary of T at distance ε from x (see Figure 7.17b). The line segment ap intersects the geodesic at a point s'. By the triangle inequality, $s'a$ is shorter than following the geodesic from s' via x to a. But then this would give a shorter path between a and b, by following the geodesic to s' and then cutting directly to a. As the geodesic is the shortest path by definition, this is impossible. Thus, the geodesic cannot intersect e at x and the new edge must be incident to a. □

LEMMA 7.24. *Let a and b be two consecutive internal bottom edges in the left-shelling pseudo-triangulation, such that the pseudo-triangle to the right of b is a triangle. Then the pseudo-pentagon X formed by removing a and b has five bitangents.*

Proof. Let $c_0, \ldots, c_4$ be the corners of X in counter-clockwise order around the boundary. By Lemma 7.23, flipping b results in an edge b' that intersects b and is incident on c_1. This edge is part of the geodesic between c_1 and c_3, and as such it is tangent to the convex chain $v_0, v_a, \ldots, c_3$, where v_a is the top endpoint of a (v_a could be c_3). Therefore it is also the tangent from c_1 to the convex hull of $\{v_0, \ldots, v_a\}$. This means that the newly created pseudo-triangle with c_1 as corner and a on the opposite pseudo-edge also meets the conditions of Lemma 7.23. Thus, flipping a results in another edge, a', also incident on c_1. As b separates c_1 from all vertices in $\{v_0, \ldots, v_a\}$, a' must also intersect b. This gives us four bitangents, of which two are incident on v_0 (a and b), and two on c_1 (a' and b'). Finally, flipping a before flipping b results in a bitangent that is not incident on v_0 (as v_0 is a corner and cannot be on the new geodesic), nor on c_1 (as b separates a from c_1). Thus, X has five bitangents. □

LEMMA 7.25. *Let a and b be two consecutive internal bottom edges in the left-shelling pseudo-triangulation, such that the pseudo-triangle to the right of b is not a triangle. Then the pseudo-pentagon X formed by flipping the corresponding top edge of b and removing a and b has five bitangents.*

Proof. Let v_a and v_b be the top endpoints of a and b. By Lemma 7.23 and since b had degree two, flipping the top edge of b results in the edge $v_b c_1$. We get three bitangents for free: a, b, and b' – the old top edge of b and the result of flipping b.

X consists of a reflex chain C that is part of the convex hull of the points to the left of a, followed by three successive tangents to C, v_a, or v_b. Since C lies completely to the left of a, it cannot significantly alter any of the geodesics or bitangents inside the polygon, so we can reduce it to a single edge. Now, X consists either of a triangle with two internal vertices, or a convex quadrilateral with one internal vertex.

If X is a triangle with two internal vertices, the internal vertices are v_a and v_b. Let its exterior vertices be v_0, x, and y. Then there

are seven possible bitangents: $a = v_0v_a, b = v_0v_b, xv_a, xv_b, yv_a, yv_b$, and v_av_b. We know that xv_a and yv_b are edges, so there are five possible bitangents left. As all vertices involved are either corners or have degree one in X, the only condition for an edge to be a bitangent is that it does not cross the boundary of X. Since the exterior boundary is a triangle, this reduces to it not crossing xv_a and yv_b. Two line segments incident to the same vertex cannot cross. Thus, xv_b, yv_a, and v_av_b cannot cross xv_a and yv_b, and X has five bitangents.

If X's convex hull has four vertices, the internal vertex is v_b (otherwise the pseudo-triangle to the right of b would be a triangle). Let its exterior vertices be v_0, x, v_a, and y. Then there are six possible bitangents: $a = v_0v_a, b = v_0v_b, xy, xv_b, yv_b$, and v_av_b, of which one (yv_b) is an edge of X. Since a and b are guaranteed to be bitangents, and xy, xv_b, and v_av_b all share an endpoint with yv_b, the arguments from the previous case apply and we again have five bitangents. □

7.4.2 *General pseudo-triangulations*

In this section, we extend our results for edge-labelled pointed pseudo-triangulations to all edge-labelled pseudo-triangulations. Since not all pseudo-triangulations have the same number of edges, we need to allow flips that change the number of edges. In particular, we allow a single edge to be deleted or inserted, provided that the result is still a pseudo-triangulation.

Since we are dealing with edge-labelled pseudo-triangulations, we need to determine what happens to the edge labels. It is useful to first review the properties we would like these flips to have. First, a flip should be a local operation – it should affect only one edge. Second, a labelled edge should be flippable if and only if the edge is flippable in the unlabelled setting. This allows us to re-use the existing results on flips in pseudo-triangulations. Third, flips should be reversible. Like most proofs about flips, our proof in the previous section crucially relies on the reversibility of flips.

With these properties in mind, the edge-deletion flip is rather straightforward – the labelled edge is removed, and other edges are not affected. Since the edge-insertion flip needs to be the inverse of this, it should insert the edge and assign it a *free label* – an unused label in $\{1, \ldots, 3n - 3 - 2h\}$, where h is the number of vertices on the convex hull ($3n - 3 - 2h$ is the number of internal edges in a triangulation).

With the definitions out of the way, we can turn our attention to the number of flips required to transform any edge-labelled pseudo-triangulations into any other. In this section, we show that by using insertion and deletion flips, we can shuffle (permute the labels on bottom edges) with $O(n + h \log h)$ flips. Combined with the unlabelled bound of $O(n \log c)$ flips by Aichholzer et al. [1], this brings

the total number of flips down to $O(n \log c + h \log h)$. Note that, by Theorem 7.3, this holds for a set of points in convex position ($h = n$). In the remainder of this section we assume that $h < n$. As before, we first build a collection of simple tools that help prove the main result.

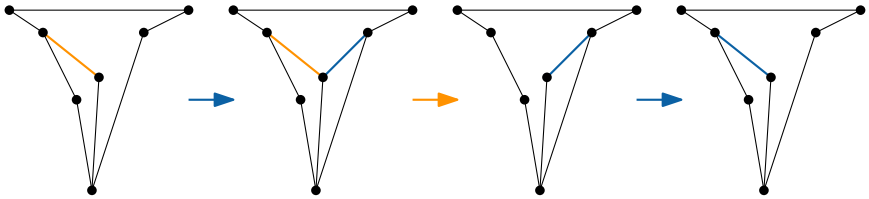


Figure 7.18: Interchanging the label of an edge incident to a vertex of degree two with a free label.

LEMMA 7.26. *With $O(1)$ flips, we can interchange the label of an edge incident to an internal vertex of degree two with a free label.*

Proof. Let v be a vertex of degree two and let e be an edge incident to v. Since v has degree two, its removal leaves an empty pseudo-triangle T. There are three bitangents that connect v to T, one for each corner. Thus, we can insert the third bitangent f with the desired free label, making v non-pointed (see Figure 7.18). Flipping e now removes it and frees its label. Finally, flipping f moves it into e's starting position, completing the exchange. □

This implies that, using an arbitrary free label as placeholder, we can swap any two edges incident to internal degree-two vertices – no matter where they are in the pseudo-triangulation.

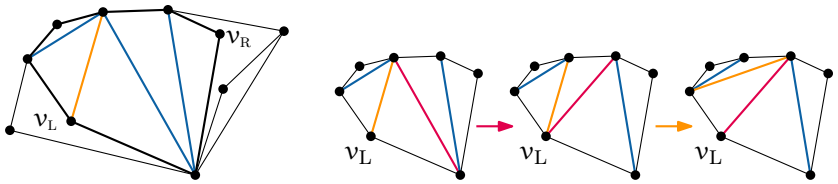
COROLLARY 7.27. *We can interchange the labels of two edges, each incident to some internal vertex of degree two, with $O(1)$ flips.*

Recall that during a sweep (Lemma 7.18), each internal vertex has degree two at some point. Since the number of free labels for a pointed pseudo-triangulation is equal to the number of internal vertices, this means that we can use Lemma 7.26 to swap every label on a bottom edge incident to an internal vertex with a free label by performing a single sweep. Afterwards, a second sweep can replace these labels on the bottom edges in any desired order. Thus, permuting the labels on bottom edges incident to internal vertices can be done with $O(n)$ flips. Therefore, the difficulty in permuting the labels on all bottom edges lies in bottom edges that are not incident to an internal vertex, that is, chords of the convex hull. If there are few such chords, a similar strategy (free them all and replace them in the desired order) might work. Unfortunately, the number of free labels can be far less than the number of chords.

We now consider operations on maximal groups of consecutive chords, which we call *fans*. As the vertices of a fan are in convex

position, fans behave in many ways like triangulations of a convex polygon, which can be rearranged with $O(n \log n)$ flips (Theorem 7.3). The problem now becomes getting the right set of labels on the edges of a fan.

Consider the internal vertices directly to the left (v_L) and right (v_R) of a fan F, supposing both exist. Vertex v_L has degree two and forms part of the reflex chain of the first pseudo-triangle to the left of F. Thus, flipping v_L's top edge connects it to the leftmost vertex of F (excluding v_0). Vertex v_R is already connected to the rightmost vertex of F, so we just ensure that it has degree two. To do this, we flip all incident edges from vertices further to the right, from the bottom to the top. Now the diagonals of F form a triangulation of a convex polygon whose boundary consists of v_0, v_L, the top endpoints of the chords, and v_R (see Figure 7.19a). It is possible that there is no internal vertex to one side of F. In that case, there is only one vertex on that side of F, which is part of the convex hull, and we can simply use that vertex in place of v_L or v_R without flipping any of its edges. Since there is at least one internal vertex by assumption, either v_L or v_R is an internal vertex. This vertex is called the *index* of F. If a vertex is the index of two fans, it is called a *shared index*.



(a) (b)

Figure 7.19: (a) An indexed fan. (b) Shifting the index (v_L) from the yellow edge to the red edge.

A triangulated fan is called an *indexed fan* if there is one edge incident to the index, the *indexed edge*, and the remaining edges are incident to one of the neighbours of the index on the boundary. Initially, all diagonals of F are incident to v_0, so we transform it into an indexed fan by flipping the diagonal of F closest to the index. Next, we investigate several operations on indexed fans that help us move labels between fans.

LEMMA 7.28 (Shift). *In an indexed fan, we can shift the indexed edge to the next diagonal with* $O(1)$ *flips.*

Proof. Suppose that v_L is the index (the proof for v_R is analogous). Let e be the current indexed edge, and f be the leftmost diagonal incident to v_0. Then flipping f followed by e makes f the only edge incident to the index and e incident to the neighbour of the index (see Figure 7.19b). Since flips are reversible, we can shift the index the other way too. □

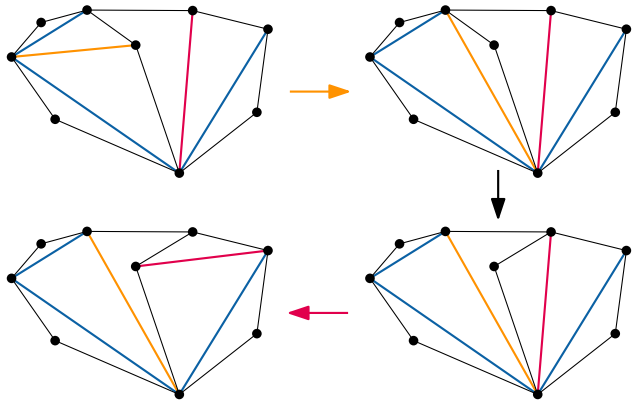


Figure 7.20: Changing which side a shared index indexes.

LEMMA 7.29. *We can switch which fan a shared index currently indexes with* $O(1)$ *flips.*

Proof. Flipping the current indexed edge "parks" it by connecting it to the two neighbours of the index, and reduces the degree of the index to two (see Figure 7.20). Now, flipping the top edge of the index connects it to the other fan, where we parked the previously indexed edge. Flipping that edge connects it to the index again. □

LEMMA 7.30. *In a pointed pseudo-triangulation, we can always decrease the degree of a vertex* v *of degree three by flipping one of the edges incident to its reflex angle.*

Proof. Consider the geodesic from v to the opposite corner c of the pseudo-triangle v is pointed in. The line supporting the part of the geodesic when it reaches v splits the edges incident to v into two groups. As there are three edges, one of these groups must contain multiple edges. Flipping the edge incident to its reflex angle in the group with multiple edges results in a geodesic to c. If this geodesic passed through v, it would insert the missing edges along the geodesic from v to c (otherwise we could find a shorter path). But inserting this geodesic would make v non-pointed. Thus, v cannot be on this geodesic. Therefore the new edge is not incident to v and the flip reduces the degree of v. □

Since the index always has degree three, this allows us to extend the results from Lemmas 7.26 and 7.17 regarding vertices of degree two to indexed edges.

COROLLARY 7.31. *In an indexed fan, we can interchange the label of the indexed edge with a free label in* $O(1)$ *flips.*

COROLLARY 7.32. *Given two indexed fans, we can interchange the labels of the two indexed edges with* $O(1)$ *flips.*

Now we have enough tools to shuffle the bottom edges.

LEMMA 7.33 (Shuffle). *In the left-shelling pseudo-triangulation, we can reorder the labels of all bottom edges with* $O(n + h \log h)$ *flips, where* h *is the number of vertices on the convex hull.*

Proof. In the initial pseudo-triangulation, let B and $\mathcal{F}$ be the sets of labels on bottom edges and free labels, respectively. Let F_i be the set of labels on the i-th fan (in some fixed order), and let $\overline{F}$ be the set of labels on non-fan bottom edges. Let F'_i and $\overline{F}'$ be these same sets in the target pseudo-triangulation. As we are only rearranging the bottom labels, we have that $B = F_1 \cup \ldots \cup F_k \cup \overline{F} = F'_1 \cup \ldots \cup F'_k \cup \overline{F}'$, where k is the number of fans.

We say that a label ℓ *belongs to* fan i if $\ell \in F'_i$. At a high level, the reordering proceeds in four stages. In stage one, we free all labels in $\overline{F}$. In stage two, we place each label from $B \setminus \overline{F}'$ in the fan it belongs to, leaving the labels in $\overline{F}'$ free. Then, in stage three, we correct the order of the labels within each fan. Finally, we place the labels in $\overline{F}'$ correctly.

Since each internal vertex contributes exactly one top edge, one bottom edge, and one free label, we have that $|\overline{F}| = |\mathcal{F}|$. To free all labels in $\overline{F}$, we perform a sweep (see Lemma 7.18). As every internal vertex has degree two at some point during the sweep, we can exchange the label on its bottom edge with a free label at that point, using Lemma 7.26. This requires $O(n)$ flips. The labels in $\mathcal{F}$ remain on the bottom edges incident to internal vertices throughout stage two and three, as placeholders.

To begin stage two, we index all fans with $O(n)$ flips and shift these indices to the first 'foreign' edge: the first edge whose label does not belong to the current fan. If no such edge exists, we can ignore this fan for the remainder of stage two, as it already has the right set of labels. Now suppose that there is a fan F_i whose indexed edge e is foreign: $\ell_e \notin F'_i$. Then either $\ell_e \in F'_j$ for some $j \neq i$, or $\ell_e \in \overline{F}'$. In the first case, we exchange ℓ_e with the label on the indexed edge of F_j, and shift the index of F_j to the next foreign edge. In the second case, we exchange ℓ_e with a free label in $B \setminus \overline{F}'$. If this label belongs to F_i, we shift its index to the next foreign edge. In either case, we increased the number of correctly placed labels by at least one. Thus $n - 1$ repetitions suffice to place all labels in the fan they belong to, wrapping up stage two. Since we perform a linear number of swaps and shifts, and each takes a constant number of flips, the total number of flips required for stage two is $O(n)$.

For stage three, we note that each indexed fan corresponds to a triangulation of a convex polygon. As such, we can rearrange the labelled diagonals of a fan F_i into their desired final position with $O(|F_i| \log |F_i|)$ flips (Theorem 7.3). Thus, if we let h be the number of

vertices on the convex hull, the total number of flips for this step is bounded by

$$\sum_i O(|F_i| \log |F_i|) \leqslant \sum_i O(|F_i| \log h) = O(h \log h).$$

For stage four, we first return to a left-shelling pseudo-triangulation by un-indexing each fan, using $O(n)$ flips. After stage two, the labels in $\bar{F}'$ are all free, so all that is left is to place these on the correct bottom edges, which we can do with a final sweep. Thus, we can reorder all bottom labels with $O(n + h \log h)$. □

This leads to the following bound.

THEOREM 7.34. *We can transform any edge-labelled pseudo-triangulation with n vertices into any other with $O(n \log c + h \log h)$ flips, where c is the number of convex layers and h is the number of vertices on the convex hull.*

Proof. Using the technique by Aichholzer et al. [1], we first transform the pseudo-triangulation into the left-shelling pseudo-triangulation T with $O(n \log c)$ flips. Our canonical pseudo-triangulation contains the labels $\{1, \ldots, 2n - h - 3\}$, but it is possible for T to contain a different set of labels. Since all labels are drawn from $\{1, \ldots, 3n - 2h - 3\}$, at most $n - h$ labels differ. This is exactly the number of internal vertices. Thus, we can use $O(n + h \log h)$ flips to shuffle (Lemma 7.33) all non-canonical labels on fan edges to bottom edges incident to an internal vertex. Once there, we use a sweep (Lemma 7.18) to ensure that every internal vertex has degree two at some point, at which time we replace its incident non-canonical labels with canonical ones with a constant number of flips (Lemma 7.26). Once our left-shelling pseudo-triangulation has the correct set of labels, we use a constant number of shuffles and sweeps to sort the labels (Lemma 7.16). Since we can shuffle and sweep with $O(n + h \log h)$ and $O(n)$ flips, respectively, the total number of flips reduces to $O(n \log c + n + h \log h) = O(n \log c + h \log h)$. □

The correspondence between triangulations of a convex polygon and pseudo-triangulations gives us the following lower bound.

THEOREM 7.35. *There are pairs of edge-labelled pseudo-triangulations with n vertices such that any sequence of flips that transforms one into the other has length $\Omega(n \log n)$.*

7.5 CONCLUSIONS AND OPEN PROBLEMS

We initiated the study of the diameter of the flip graph of edge-labelled triangulations in various settings. For edge-labelled triangulations of a convex polygon, we presented matching upper and lower bounds of $\Theta(n \log n)$. Allowing simultaneous flips brings the upper

bound down to $O(\log^2 n)$, while our best lower bound in that setting is $\Omega(\log n)$. For edge-labelled combinatorial triangulations, we obtained the same tight $\Theta(n \log n)$ bound on the diameter of the flip graph, as well as the $O(\log^2 n)$ and $\Omega(\log n)$ upper- and lower bounds for simultaneous flips.

We also studied the diameter of the flip graph of edge-labelled pseudo-triangulations. Here, we showed that $O(n^2)$ exchanging flips suffice to transform any edge-labelled pointed pseudo-triangulation into any other, while $\Omega(n \log n)$ flips are sometimes necessary. By allowing insertion and deletion flips in addition to exchanging flips, we obtained a tight $\Theta(n \log n)$ bound on the shortest flip sequence between any two edge-labelled pseudo-triangulations, even non-pointed ones.

There is still a lot of room for future work. The most obvious set of open problems is closing the gaps between the $O(\log^2 n)$ and $\Omega(\log n)$ bounds in the simultaneous setting, and the $O(n^2)$ and $\Omega(n \log n)$ bounds for pointed pseudo-triangulations. The most likely solution for the latter is to prove a result similar to Lemma 7.1 in this setting. This is easy when the vertices in the subsequence form a convex or concave chain, but handling alternations will be trickier.

The next set of open problems is to study what happens to the edge-labelled flip graph in non-convex polygons or point sets. Since it is possible for edges to be unflippable in these settings, we can never change the label of such edges with flips alone, resulting in a disconnected flip graph. In fact, each edge has a fixed set of other edges that it can be transformed into via flips. We call this set the *orbit* of the edge. This gives rise to the following conjecture.

CONJECTURE 1 (Orbit Conjecture). *Given two edge-labelled triangulations,* T *and* T'*, we can transform* T *into* T' *if and only if for every label* ℓ*, the edge with label* ℓ *in* T' *is in the orbit of the edge with label* ℓ *in* T*.*

For convex polygons, the Orbit Conjecture is implied by Theorem 7.3, since the orbit of each edge contains all other edges. In addition to convex polygons, the Orbit Conjecture also holds for polygons with a single reflex chain [17]. For general polygons, it is fairly easy to show that the condition is necessary, but we have not been able to show that it is also sufficient. Since triangulations of a set of points in the plane face the same difficulties, we believe that the Orbit Conjecture holds for that setting as well.

The final set of open problems relates to the computational hardness of finding the shortest sequence of flips that transforms one edge-labelled triangulation into the other. For unlabelled convex polygons, this question has been open for over 30 years [10]. Recently, the problem was shown to be NP-hard for triangulations of point sets [16] and simple polygons [3], but the case of convex polygons remains open.

We see two ways in which edge-labelled triangulations can help here. First, if we lift the restriction that all edge-labels have to be

unique, the problem directly generalizes the unlabelled setting. Thus, an NP-hardness proof in this setting might generate techniques that can be reused for the unlabelled setting.

Second, note that every flip sequence in the unlabelled setting defines a bijection between edges of the initial and final triangulations. We can view the flip distance problem in the unlabelled setting as consisting of two steps: find the best bijection, and find a minimum flip sequence that realizes this bijection. The edge-labelled version of the problem is just the second step in this chain. Thus, settling the complexity of the edge-labelled version could give insight into which part of the unlabelled problem generates the complexity.

BIBLIOGRAPHY

[1] Oswin Aichholzer, Franz Aurenhammer, Clemens Huemer, and Hannes Krasser. Transforming spanning trees and pseudo-triangulations. *Information Processing Letters*, 97(1):19–22, 2006.

[2] Oswin Aichholzer, Franz Aurenhammer, Hannes Krasser, and Peter Brass. Pseudotriangulations from surfaces and a novel type of edge flip. *SIAM Journal on Computing*, 32(6):1621–1653, 2003.

[3] Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015.

[4] Gabriela Araujo-Pardo, Isabel Hubard, Deborah Oliveros, and Egon Schulte. Colorful associahedra and cyclohedra. *Journal of Combinatorial Theory, Series A*, 129:122–141, 2015.

[5] Sergey Bereg. Transforming pseudo-triangulations. *Information Processing Letters*, 90(3):141–145, 2004.

[6] Prosenjit Bose, Jurek Czyzowicz, Zhicheng Gao, Pat Morin, and David R. Wood. Simultaneous diagonal flips in plane triangulations. *Journal of Graph Theory*, 54(4):307–330, 2007.

[7] Prosenjit Bose and Sander Verdonschot. Flips in edge-labelled pseudo-triangulations. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG 2015)*, pages 63–69, 2015.

[8] Javier Cano, José-Miguel Díaz-Báñez, Clemens Huemer, and Jorge Urrutia. The edge rotation graph. *Graphs and Combinatorics*, 29(5):1207–1219, 2013.

[9] Jean Cardinal, Michael Hoffmann, Vincent Kusters, Csaba D. Tóth, and Manuel Wettstein. Arc diagrams, flip distances, and Hamiltonian triangulations. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, pages 197–210, 2015.

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

[11] Jerôme Galtier, Ferran Hurtado, Marc Noy, Stéphane Pérennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *International Journal of Computational Geometry & Applications*, 13(2):113–133, 2003.

[12] Carmen Hernando, Ferran Hurtado, Mercè Mora, and Eduardo Rivera-Campo. Grafos de árboles etiquetados y grafos de árboles geométricos etiquetados. In *Proceedings of the X Spanish Meeting on Computational Geometry (Encuentros de Geometra Computacional)*, pages 13–19, 2003.

[13] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[14] David Kirkpatrick, Jack Snoeyink, and Bettina Speckmann. Kinetic collision detection for simple polygons. *International Journal of Computational Geometry & Applications*, 12(1-2):3–27, 2002.

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

[16] Anna Lubiw and Vinayak Pathak. Flip-distance between two triangulations of a point-set is NP-complete. In *Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2012)*, pages 119–124, 2012.

[17] Vinayak Pathak. *Reconfiguring triangulations*. PhD thesis, University of Waterloo, 2014.

[18] Günter Rote, Francisco Santos, and Ileana Streinu. Pseudo-triangulations—a survey. In *Surveys on discrete and computational geometry*, volume 453 of *Contemporary Mathematics*, pages 343–410. 2008.

[19] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988.

[20] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Short encodings of evolving structures. *SIAM Journal on Discrete Mathematics*, 5(3):428–450, 1992.

[21] Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete & Computational Geometry*, 34(4):587–635, 2005.

[22] Klaus Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of September 2, 2015 (`classicthesis` version 4.1).