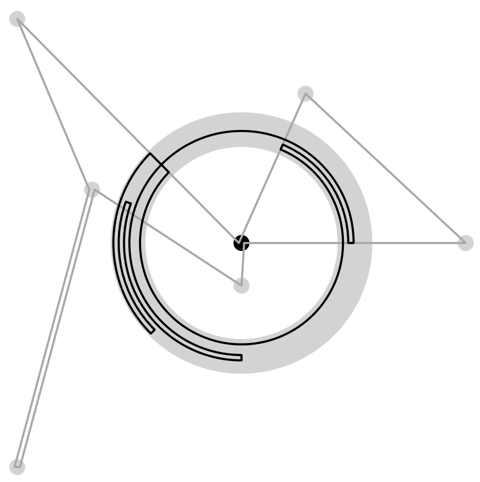
Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik I:
Algorithmen, Komplexität und wissensbasierte Systeme

Master Thesis

A Combinatorial Upper Bound on the Length of Twang Cascades



Leon Sering

Submitted on February 1, 2016

First reviewer: Prof. Dr. Alexander Wolff
Second reviewer: Prof. Dr. Christian Glaßer

Abstract

Damian et al. [DFOR10] have introduced an intuitive type of transformations between simple polygons on a finite set of points in the plane. Each transformation consists of a sequence of atomic modifications of two types, called *stretches* and *twangs*. They have proven that, for a given set of n points, the space of these simple polygons is connected by $O(n^2)$ such transformations.

In this thesis we solve an open question of Damian et al. concerning a combinatorial upper bound on the length of these sequences. To this end, we show that the length of a twang cascade is bounded by $n^3/2$. Furthermore, we present an algorithm that generates a random simple polygon with approximately uniform distribution. This algorithm is based on the above mentioned transformations and uses a Markov chain Monte Carlo sampling.

Zusammenfassung

Wir betrachten kreuzungsfreie Polygone, die eine endliche Menge von Punkten in der Ebene abdecken. Damian et al. [DFOR10] beschreiben eine intuitive Transformationsart solcher Polygone. Jede dieser Transformationen besteht aus einer Sequenz elementarer Modifikationen, welche sich in zwei Klassen unterteilen lassen: *Stretches* und *Twangs*. Sie konnten zeigen, dass für eine vorgegebene Menge von n Punkten, der Raum der kreuzungsfreien Polygone durch $O(n^2)$ solcher Transformationen verbunden ist.

In dieser Arbeit lösen wir ein offenes Problem von Damian et al., welches die kombinatorische obere Schranke der Länge solcher Sequenzen betrifft. Wir beweisen, dass eine Twang-Kaskade nicht länger als $n^3/2$ sein kann. Des Weiteren geben wir einen Algorithmus an, der zufällige kreuzungsfreie Polygone generiert, wobei die Verteilung auf dem Polygon-Raum annäherungsweise gleichmäßig ist. Der Algorithmus basiert auf der oben genannten Transformation und verwendet ein Markov-Ketten-Monte-Carlo-Stichprobenverfahren.

Contents

1 Introduction

This thesis studies simple polygons on a fixed set S of n points in the plane. For ease of presentation, we assume S to be in general position, that is no three points lie on a line. For the general case some minor modification of the proofs may be necessary for the general case. A *polygon* on S is a cycle of straight line segments, each connecting two points of S such that every point of S is visited exactly once. A *simple polygon* is a polygon without crossings. Let $\mathcal{P}_S$ be the set of all simple polygons on S. We write *polygon* when we consider simple polygons throughout this thesis.

1.1 Counting and Sampling Polygons

We present two challenging open problems in the context of polygons on a fixed point set:

1. The Counting Problem:

   There is no efficient algorithm known to count the number of polygons on a given point set S. By *efficient* we mean an algorithm that runs in polynomial time. On the one hand there exist some simple subclasses of point sets where the counting is trivial. For example, if S is in convex position, that is, every point lies on the boundary of the convex hull of S, then there exists only one polygon. On the other hand the number of polygons can be exponential concerning the number of given points. There are point sets where the number of polygons lies in $\Theta(4.6^n)$ [GNT00].

   There exists a simple exponential-time brute-force algorithm that counts all polygons: Create all $n!$ permutations of the n points and check whether this induces a crossing-free cycle or not. Check for duplications and save all unique polygons in a list. In the end return the length of the list.

2. The Sampling Problem:

   Given a point set S as input. The task is to generate a random polygon on S with *uniform distribution*, that is, if $r := |\mathcal{P}_S|$ is the number of polygons on S, we want to choose one with a probability of $\frac{1}{r}$. No efficient algorithm is known for this problem.

   The brute-force algorithm mentioned above works here as well: First, generate all polygons on S and afterwards choose one of them uniformly at random.

Intuitively, the counting problem seems to be the harder one, which is underlined by the following connection between the two problems: Zhu et al. [ZSSM96] state an intuitive way to generate a random polygon. It is an incremental approach that builds the polygon edge by edge. In order to achieve uniform distribution, we need to use the right probabilities in each step. Algorithm 1 generates a uniformly distributed random polygon and terminates after a finite number of steps. Note that the symbol () in line 1 stands for the empty sequence, which is contained in every polygon.

Algorithm 1: Extending Algorithm for Random Polygons

```
   Input: Point Set $S = \{s_1, \dots, s_n\}$.
   Output: Random Polygon $P$.
 1 Set $X_0 = ()$
 2 for $i = 1$ to $n$ do
 3     $p_1, \dots, p_n = 0$
 4     for $s_j \notin X_i$ do
 5         if $(X_i, s_j)$ is crossing-free then
 6             Let $N_{X_i}$ be the number of polygons on $S$ that contain $X_i$.
 7             Let $N_{(X_i,s_j)}$ be the number of polygons on $S$ that contain $(X_i, s_j)$.
 8             $p_j = N_{(X_i,s_j)}/N_{X_i}$
 9     Let $s = s_j$ with probability $p_j$.
10     $X_{i+1} = (X_i, s)$
11 return $X_n$
```

The probability that Algorithm 1 generates a specific polygon $P = (t_1, \dots, t_n)$, with $\{t_1, \dots, t_n\} = S$ is

$$\mathcal{P}(X_n = P) = \frac{N_{(t_1)}}{N_{()}} \cdot \frac{N_{(t_1,t_2)}}{N_{(t_1)}} \cdots \frac{N_P}{N_{(t_1,\dots,t_{n-1})}} = \frac{N_P}{N_{()}} = \frac{1}{r}$$

In order to compute the right probabilities in lines 6 and 7, we need to solve a more general counting problem in each step: Given a polygonal sequence X of some subset of S, we need to count the number of polygons on S that extend X. Such a counting algorithm would also solve the simple counting problem by choosing $X = ()$. Hence, there is no efficient method known to calculate this number.

In this thesis, we focus on the sampling problem. Aside from being of theoretical interest, a solution to the sampling problem would help to evaluate and verify the time complexities of algorithms operating on arbitrary simple polygons. Since there can be exponentially many of them, the only time-efficient option for larger scales is to test the algorithm on random polygons using an efficient random generator.

There are many different approaches to generate simple polygons. On the one hand, there are efficient generators for specific subsets of simple polygons such as x-monotone polygons [ZSSM96] or star-shaped polygons [AH98]. On the other hand, there are algorithms that produce all possible polygons with positive probability but they are not generated uniformly at random. For example, the *2-Opt-Moves* algorithm [AH98] produces a random permutation of S and then removes all self-intersections step by step.

1.2 Random Walk Approach

We focus on the uniform case and we will approximate it by a so called *Markov chain Monte Carlo sampling.* In order to do this, we define a class of transformations, each turning one polygon into another, such that for every polygon the number of applicable transformations is bounded polynomially. The algorithm starts with an arbitrary polygon, chooses an available transformation at random and applies it in order to produce a new polygon. We repeatedly execute transformations at random until we stop at some point in time. The last polygon is the output.

In other words, we simulate a *random walk* on the transformation graph induced by the defined class of transformations: The *transformation graph* of a transformation type T is the directed graph $\mathcal{G}_T$ whose vertices consist of all polygons on S and where two vertices are connected by an directed edge whenever there is a transformation of type T that turns one polygon into the other. In order to reach every polygon with positive probability, our transformation class must meet the following two conditions:

1. *Connectivity*: Every possible polygon must be reachable from every other polygon by a sequence of transformations. In other words, $\mathcal{G}_T$ must be connected.

2. *Aperiodicity*: The greatest common divisor of all cycles in $\mathcal{G}_T$ is 1. For example, $\mathcal{G}_T$ must not be bipartite.

In the notation of Markov chains, we say that a random walk is *irreducible* if the transition graph is connected and it is *ergodic* if it meets both of these two conditions.

Aperiodicity can be guaranteed by simply including the identity as transformation. That means the polygon can be left unmodified by some positive probability at every step. We get a so called *lazy random walk.* Graph-theoretically this can be modeled by adding a loop on every vertex of $\mathcal{G}_T$. Therefore, aperiodicity is not the challenge, but to find a transformation that can be efficiently computed and that also connects all polygons on S is an unsolved task so far.

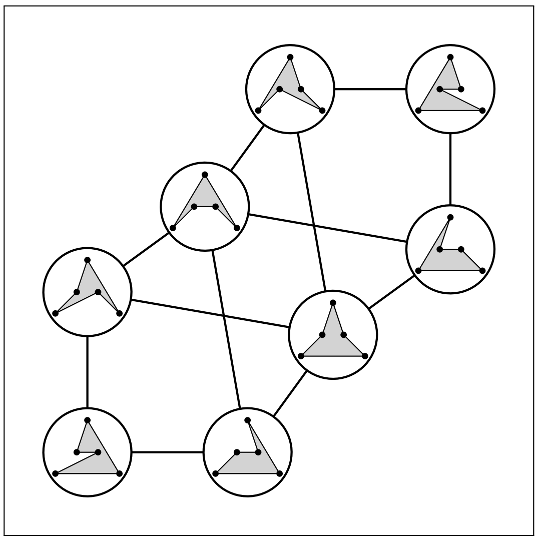
A random walk with an ergodic transformation converges to its *stationary distribution*, which has a positive probability for each polygon. Furthermore, it is possible to define the transition probabilities in a way such that the stationary distribution is uniform. The *convergence rate* of the random walk is of great interest as well. It highly depends on the connectivity and the diameter of the transformation graph. Depending on the convergence rate, we can determine the *mixing time*, in other words when to stop our algorithm to guarantee a desired probability for each polygon.

The book *Markov Chains and Mixing Times* [LPW09] gives a good introduction to this subject and provides all needed techniques and results that we will use in Chapter 4.

1.3 Local Transformations: The Flip

An important class of local transformations on polygons is called *flips.* Flips have the advantage of being reversible, that is, the corresponding transformation graphs, in this case called *flip graphs*, are undirected.

The simplest flip is the 2-flip. It removes two edges of the polygon and adds two other in order to get a new polygon. Figure 1.1a shows a complete 2-flip-graph for a set of five points. Unfortunately, there are point sets where the flip graph is not connected [Hou96], which is shown by the existence of an isolated vertex. In other words there exists a polygon that is not 2-flippable at all. See Figure 1.1b for such a polygon.


(a) An example of a connected 2-flip-graph.

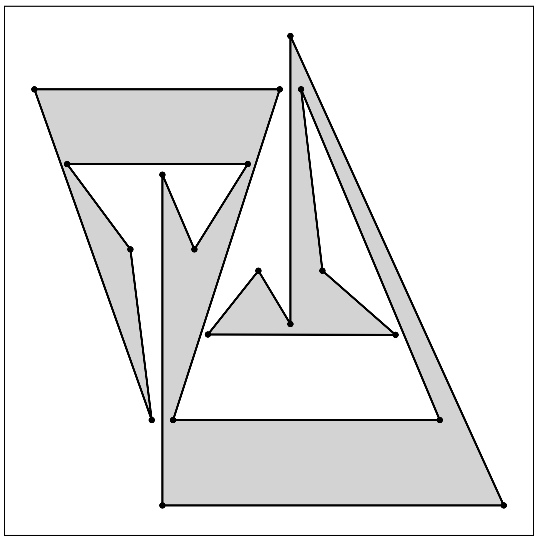

(b) A non-2-flippable polygon leading to an isolated vertex in the 2-flip-graph.

Figure 1.1: The 2-flip is the simplest transformation.

For the 3-flip (removing three edges and adding three new ones) it is still unknown whether the corresponding flip graph is always connected. In fact, for any number $k \geq 3$ the question whether the flip graph of the k-flip is always connected has not been answered yet. Hernando et al. [HHH02] give a good overview of different types of flips and of subclasses of polygons that are connected by them.

Since flips are not known to be sufficient for our requirements, we consider another type of transformation that guarantees connectivity of the transformation graph. Unfortunately, our transformation is not local and, therefore, it is more complicated and takes longer to compute.

2 Stretches and Twangs

For a given set S of points in the plain, Damian, Flatland, Robin, O'Rourke and Ramaswami [DFOR10] imagine the polygon with vertex set S as an elastic band that is attached to the points in S at its vertices. They introduce a new transformation, called *forward move*, based on the idea of deforming this elastic band. It consists of two different types of atomic operations, namely *stretches* and *twangs*. We will construct a transformation graph $\mathcal{G}_S$ with transformations based on these operations.

2.1 Polygonal Wrap

Neither stretches nor twangs are guaranteed to create a new polygon because they do not preserve simplicity. They produce an object called *polygonal wrap*, which does not have any proper crossings but can be self-touching.

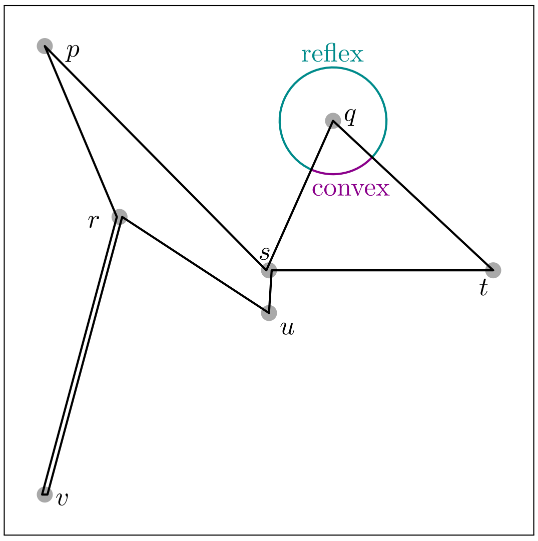


Figure 2.1: The polygonal wrap $(p, r, v, r, u, s, t, q, s)$ with multiple contacts at r and s and a hairpin at v. The reflex and the convex side of the visit (s, q, t) are shown.

Definition 2.1. A *polygonal wrap* W of length $m \geq n$ on S is a cyclic polygonal chain $(w_0, w_1, w_2, \ldots, w_m = w_0)$ such that

(W1) The wrap only bends at points of S, i.e., $w_1, w_2, \ldots, w_m \in S$.

(W2) Every point in S is visited at least once, i.e., for every $p \in S$, there is an i with $w_i = p$.

(W3) The wrap does not contain any proper crossings, i.e., there exists an arbitrarily small perturbation of the vertices of W that makes the cyclic polygonal chain non-self-intersecting.

If a point is visited more than once, we call it a point *in multiple contact* and a subsequence (a, b, a) is a *needle-pin* at b. Furthermore, we say that a line segment *ab does not properly cross W* if there is an arbitrarily small perturbation of W and ab such that W and ab do not intersect. For a visit (a, b, c), we call the cone in the minor arc the *convex side* and the complement the *reflex side*.

A polygonal wrap can be viewed as the sequence $(w_0, w_1, w_2, \dots, w_m)$ of vertices or as an undirected graph with vertex set S and an edge for every pair of adjacent vertices:

$$E = \{w_{i-1}w_i \colon i = 1, \dots, m\}$$

In both cases, we see the wrap as a one-dimensional object, and we do not make a difference between the inner and the outer face.

The small perturbation in property (W3) can be created by splitting points that are in multiple contact into one copy for each visit and move each copy slightly to the convex side of the corresponding visit. In order to clarify in which direction the chain continues, we draw this perturbation whenever we visualize a polygonal wrap. For an example, see vertices r, s and v in Figure 2.1.

In the following, we introduce two atomic operations that transform polygonal wraps with the same vertex set into each other.

2.2 Twang

A twang is an operation that transforms one polygonal wrap into another. Choose a point in multiple contact that is not a needle-pin, detach one contact from the point and let the band snap back. The snapping band does not cross other vertices but attaches to them instead; see Figure 2.2a.

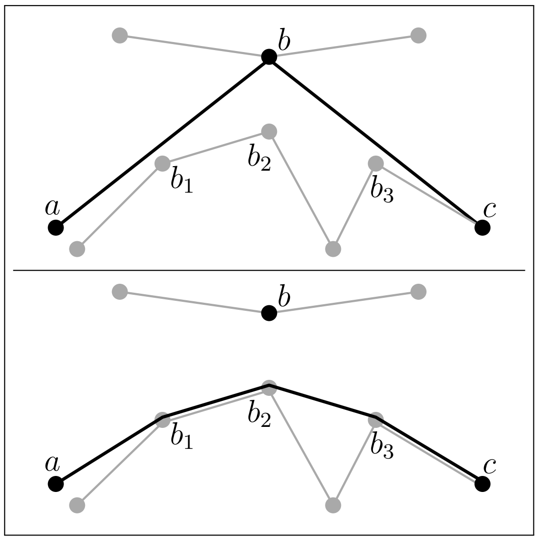
Definition 2.2. The operation $\mathrm{Tw}(abc)$ is defined for a subsequence (a, b, c) of a polygonal wrap W, whenever the following three conditions hold:

(T1) b is in multiple contact.

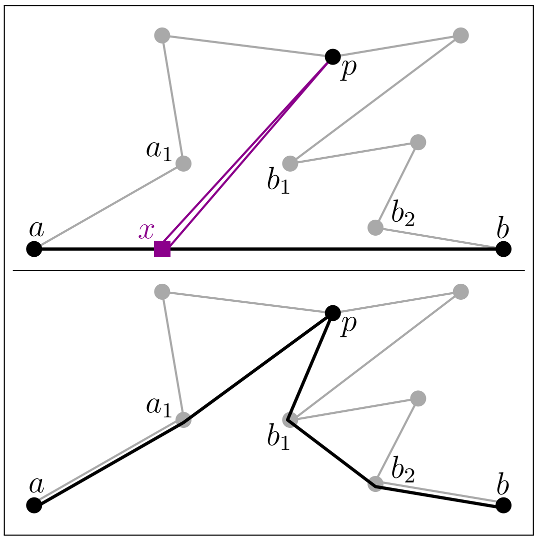
(T2) b is not a hairpin, i.e., $a \neq c$.

(T3) (a, b, c) does not contain any nested visits of b. In other words, whenever we perturb W slightly, the triangle $\triangle abc$ does not contain any other copies of b.

Then $\mathrm{Tw}(abc)$ replaces the subsequence (a, b, c) in W by $\mathrm{sp}(abc)$, where $\mathrm{sp}(abc)$ is the shortest path from a to c inside of $\triangle abc$ that does not properly cross W. Let W' be the twanged polygonal wrap. We call $\mathrm{Tw}(abc)$ a twang at b and sometimes write $W \to W'$.



(a) Twanging at b: (a, b, c) is replaced by $sp(abc) = (a, b_1, b_2, b_3, c)$. This creates a hairpin at c as well as multiple contacts at b_1, b_2 and b_3.



(b) Stretching $e = (a, b)$ to p. We add x temporarily and replace e by (a, x, p, x, b) and twang at x in both directions. The result is $(\mathrm{sp}(axp), \mathrm{sp}(pxb)) = (a, a_1, p, b_1, b_2, b)$.

Figure 2.2: The two atomic modifications: Twang and Stretch.

As long as there is at least one point in multiple contact we can apply a twang to the wrap. Furthermore, a polygonal wrap without any points in multiple contact is in fact a polygon. In other words, a twang helps us to restore a polygon from an arbitrary polygonal wrap and in some sense it removes self-touchings. Although a twang might produce more multiple contacts in the process, we will show that repeated twanging will in fact terminate and restore a polygon.

2.3 Stretch

Informally, a stretch is the operation of taking an edge e of the elastic band and attaching it to a point p. Similar to a twang, the band does not cross other points but instead wraps around them as it is shown in Figure 2.2b.

Definition 2.3. Given a polygonal wrap W. We say an edge e of W is *visible* to a point p if there is a point x in the interior of e such that the line segment px does not properly cross W. The point x is called the *spotted point*. The operation $\mathrm{St}(e, p)$ is defined for any edge $e = (a, b)$ of W and any vertex $p \in S$ if e is visible to p. To execute $\mathrm{St}(e, p)$, we replace (a, b) by $(\mathrm{sp}(axp), \mathrm{sp}(pxb))$, where x is the spotted point.

In other words, we first add the spotted point x as a pseudo-vertex to the polygonal wrap and replace (a, b) by (a, x, p, x, b). Afterwards, we twang at x twice such that x can be removed again.

A stretch is the main tool to modify a polygon and turn it into another polygon. But since it always creates at least one multiple contact, we need to apply a sequence of twangs in order to obtain a polygon.

2.4 Twang Cascade and Forward Move

Having defined the atomic operations, we can now define our transformations.

Definition 2.4. Given a polygon P with an edge e and a point p on an edge $vw \neq e$ such that p can see e, a *forward move* consists of a stretch $\mathrm{St}(e, p)$ with the additional condition that the spotted point x lies on the reflex side of the visit (v, p, w). We continue with a so-called *twang cascade*, which starts with the twang $\mathrm{Tw}(vpw)$ and repeatedly twangs as long as there are vertices in multiple contact. We write $\mathrm{ForwardM}(P, e, v)$ for the resulting polygon.

If we have more than one point to choose from, we always twang at the point with the smallest index in order to make a forward move completely determined by its initial stretch.

Lemma 2.5 ([DFOR10])**.** *A twang cascade always terminates.*

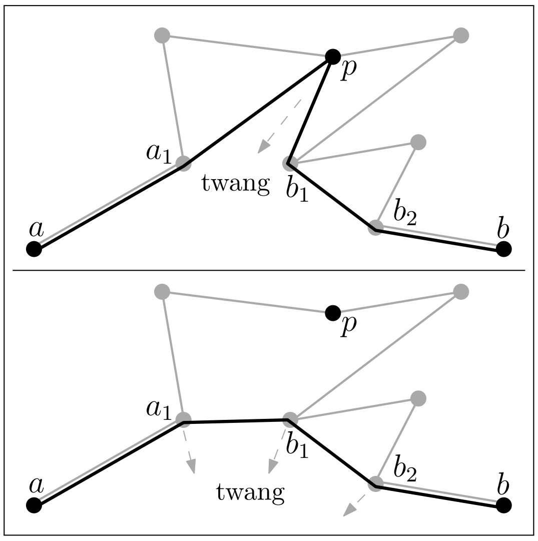
Damian et al. [DFOR10] have proven this lemma by showing that there is a fixed number $c > 0$ (depending only on S) such that the length of the polygonal wrap is decreasing by at least c with every twang. Since there is a lower bound on the length of a polygonal wrap, the twang cascade must terminate. The combinatorial upper bound on twang cascades that we give in Chapter 3 proves this lemma as well.

Since the twang cascade can only terminate if there are no points in multiple contact left, the resulting polygonal wrap must be a polygon. Hence, a forward move turns one polygon into another.

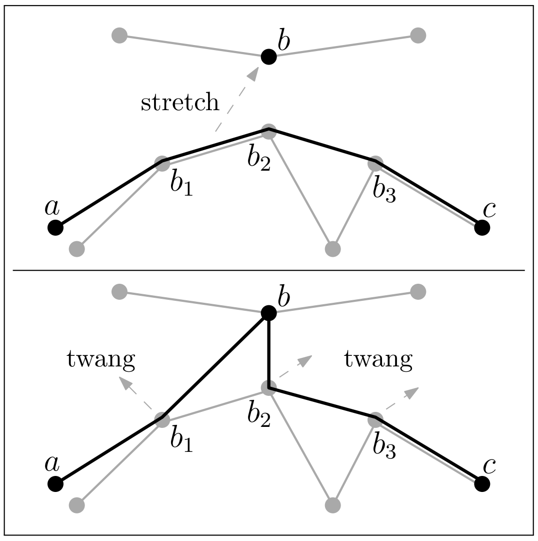
2.5 Reverse Move

In order to guarantee reversibility of the transformation, we add the time-reversal of a forward move to our transformation graph $\mathcal{G}_S$ and call it *reverse move*. Unfortunately, this is less natural and it is hard to verify how many reverse moves can be applied on a given polygon. It is not known whether or not a reverse move can be substituted by a sequence of forward moves. But at least we can decompose a reverse move into stretches and twangs as follows.

A *reverse stretch* can be executed by a sequence of one or more twangs which is illustrated in Figure 2.3a. A *reverse twang* consists of one stretch followed by a sequence of twangs as shown in Figure 2.3b, similar to a forward move. But the twang sequence might be empty or the initial stretch might not lie on the reflex side of the stretch vertex. Furthermore, the result of a reverse twang might not be a polygon. In all these cases the reversal would not be a forward move.



(a) A reverse stretch is a sequence of twangs: $\mathrm{Tw}(a_1pb_1)$, $\mathrm{Tw}(aa_1b_1)$, $\mathrm{Tw}(ab_1b_2)$ and $\mathrm{Tw}(ab_2b)$.



(b) A reverse twang initiated by the stretch $\mathrm{St}(b_1b_2, b)$ and followed by the twang sequence $\mathrm{Tw}(ab_1b)$, $\mathrm{Tw}(bb_2b_3)$, $\mathrm{Tw}(bb_3c)$.

Figure 2.3: A reverse move can be decomposed into stretches and twangs.

We can express a sequence of atomic operations of a reverse move by a regular expression: If s stands for a stretch and t for a twang, a forward move is expressed by st^+. A reverse stretch consists of one or more twangs: $s^{-1} = t^+$. A reverse twang can be achieved by using a stretch followed by a sequence of twangs: $t^{-1} = st^*$. Overall a reverse move matches the following regular expression:

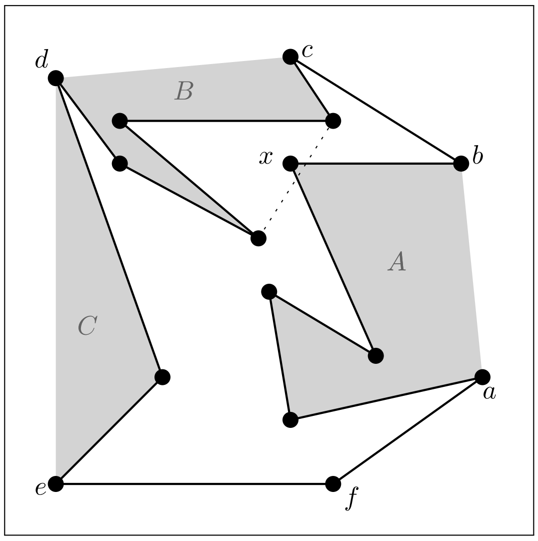
$$(st^+)^{-1} = (t^{-1})^+ s^{-1} = (st^*)^+ t^+$$

Note, however, that not every sequence matching this expression is a reverse move.

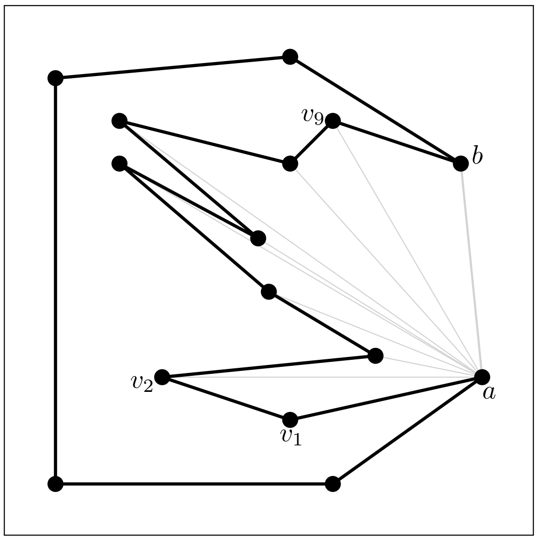
2.6 Pocket Reduction Algorithms

The main result of Damian et al. states that the transformation graph combining forward and reverse moves is connected. We present the key idea of their proof here. To this end, we need the following definitions:

Definition 2.6. Given a polygon P with vertex set S, let $c_1, \dots, c_k \in S$ be the points of $\partial\,\mathrm{CH}(S)$ in counter-clockwise order, where $\partial\,\mathrm{CH}(S)$ is the boundary of the convex hull of S. For $i = 1, \dots, k-1$, the sequence from c_i to c_{i+1} (and from c_k to c_1) is called *pocket* with *pocket lid* c_ic_{i+1}. The *size of a pocket* C is defined by $\mathrm{size}(C) := |S \cap \mathrm{CH}(C)|$, that is, the number of points of S contained in the convex hull of pocket C. A pocket of size two consists only of its lid and is called *reduced.* Fix a point $a \in \partial\,\mathrm{CH}(S)$, and let b be the counter-clockwise successor of a in $\partial\,\mathrm{CH}(S)$. The *canonical polygon* P_a is the



(a) A polygon with three non-reduced pockets: Pocket A with lid ab of size 6, pocket B with lid cd of size 7 (note that x is counted as well), and pocket C with lid ed of size 3. The three other pockets with lids bc, ef and fa are reduced.



(b) The canonical polygon P_a of point a: It has only one non-reduced pocket with lid ab and the vertices of the pocket occur clockwise around a.

Figure 2.4: Pockets and the canonical polygon.

Algorithm 2: Single Pocket Reduction

```
   Input: Polygon $P$ with a non-reduced pockets $B$.
   Output: Polygon $P'$ with $B$ reduced.
1  while size($B$) $> 2$ do
2      Choose edge-vertex pair $(e, x)$ such that:
           • $e$ is an edge of $P$ of a non-reduced pocket different from $B$.
           • $x \in B$ is a non-lid vertex on $\partial\,\mathrm{CH}(B)$ that sees $e$.
3      $P = \mathrm{ForwardM}(P, e, x)$
4  return $P$
```

polygon with only one non-reduced pocket A with lid ab, where the points of A occur in clockwise order around a.

See Figure 2.4 for some examples of pockets and a canonical polygon. There is exactly one canonical polygon for every point $a \in \partial\,\mathrm{CH}(S)$.

Next we will show that it is possible to turn any polygon P into the canonical polygon P_a, for a fixed a, by forward moves only. To this end, we reduce the pockets one by one using Algorithm 2 until there is only one pocket left. Note that, if B is the only non-reduced pocket of P, we use the edge ab, the only edge of pocket A, in line 2.

The following lemma helps us to show that Algorithm 2 always terminates and that it reduces pocket B in the end. This lemma is the reason why we defined the size of a pocket by the number of points inside its convex hull.

Lemma 2.7. *Let $W \to W'$ be a twang at a non-lid vertex. Let C be a pocket of W and C' the corresponding pocket in W', namely the pocket with the same lid as C. Then* $\operatorname{size}(C') \leq \operatorname{size}(C)$.

Proof. Let (xyz) be the sequence in W that is going to be twanged. It is replaced by $\operatorname{sp}(xyz)$ in W'. If neither xy nor yz is part of C then the twang does not affect C at all. If xy or yz is part of C then both are part of C since y is a non-lid vertex. But since $\operatorname{sp}(xyz) \subseteq \triangle xyz \subseteq \operatorname{CH}(C)$, the twang cannot increase the size of the pocket. $\square$

The initial stretch and the first twang of $\operatorname{ForwardM}(P, e, x)$ remove x from B and, since $x \in \partial \operatorname{CH}(B)$, they also remove x from $\operatorname{CH}(B)$ (x has to be a true corner). Hence the size of B is reduced by one. By Lemma 2.7 the following twang cascade does not increase the size of B. Therefore the algorithm reduces one pocket within n iterations, at most.

We repeat this single-pocket reduction algorithm until there is only one non-reduced pocket left, namely pocket A with lid ab. This can take up to $O(n^2)$ forward moves. This last pocket can then be brought into canonical form by forward moves, following Algorithm 3. After the i-th iteration, the points $v_0, \ldots, v_i$ are in canonical order and the forward move in line 6 only involves vertices of $\{v_i, v_{i+1}, \ldots, v_k\}$. Obviously this algorithm uses at most n forward moves.

Algorithm 3: Canonical Polygon

```
  Input: Polygon $P$ with only one non-reduced pocket $A$ with lid $ab$.
  Output: Canonical polygon $P_a$.
1 Let $a = v_0, v_1, \ldots, v_k, v_{k+1} = b$ be the canonical order of pocket $A$.
2 for $i = 1$ to $k$ do
3     Let $l_i$ be the line through $a$ and $v_i$.
4     Let $e_{i-1}$ be the pocket edge $v_{i-1}v_j$, with $j > i - 1$.
5     if $e_{i-1} \neq v_{i-1}v_i$ then
6         $P = \operatorname{ForwardM}(e_{i-1}, v_i)$
7 return $P$
```

For more details, especially why it is always possible to find an edge-vertex pair in line 2 of Algorithm 2, for the correctness of Algorithm 3, and for the exact complexity bounds, we refer to Damian et al. [DFOR10].

Now, to see that any two polygons P_1 and P_2 are connected in $\mathcal{G}_S$, we fix an $a \in \partial \operatorname{CH}(S)$ and turn P_1 into P_a with forward moves. Then we reverse the sequence of forward moves that would be needed to turn P_2 into P_a. Finally, we apply the resulting sequence of reverse moves to P_a in order to obtain P_2.

Overall we need at most $O(n^2)$ forward moves to turn P_1 into P_a and $O(n^2)$ reverse moves to turn P_a into P_2. Hence, the diameter of the transformation graph $\mathcal{G}_S$ is $O(n^2)$.

3 A Combinatorial Upper Bound

Damian et al. were not able to give a combinatorial upper bound on the length of a twang cascade and, therefore, no combinatorial bound on the running-time to execute a forward move was given. In their test, however, they show that, in general, these twang cascades are quite short. As a theoretical result they show, that there exist forward moves that have twang cascades of length $\Theta(n^2)$ as shown in Figure 3.1. In this section we will solve this open question and show that the length of a twang cascade is bounded by $n^3/2$ leaving a gap of factor $O(n)$.

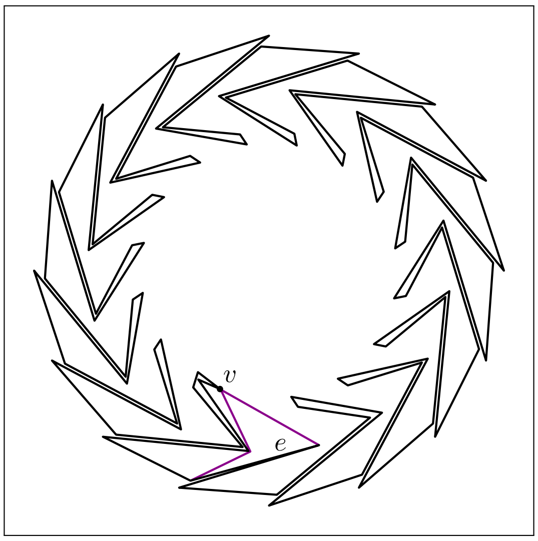
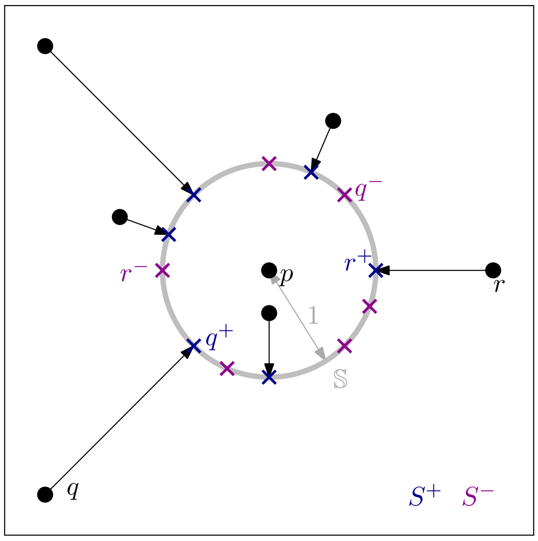


Figure 3.1: St(e, v) initiates a forward move with a twang cascade of quadratic length. The cascade twangs multiple times around the circle.

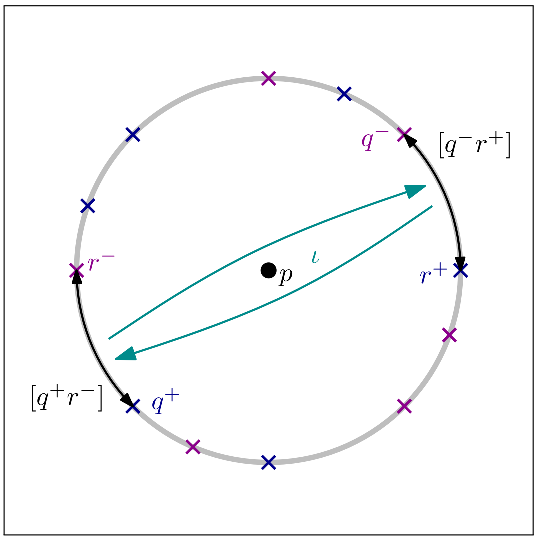
Given a set S of $n > 2$ points in the plane in general position and a polygonal wrap W that is created by a stretch on a polygon. We fix a point $p \in S$ and give an upper bound on the number of twangs that can occur at p in a twang cascade of W. We will show that this number is bounded by $n^2/2$.

3.1 Markers, Elementary Arcs and Radial Loops

Let $p \in S$ be fixed throughout this section and, for simplicity, consider it to be at the origin, that is, $p = (0, 0)$. First, we radially project every point in $S \backslash \{p\}$ to the unit circle in the way shown in Figure 3.2a.



(a) The radial projection and markers.



(b) Elementary arcs and the central inversion.

Figure 3.2: Basic definitions.

Definition 3.1. We call the following mapping *radial projection*:

$$\Pi\colon \mathbb{R}^2\backslash\{(0,0)\} \to \mathbb{S}, \qquad q \mapsto \frac{q}{\|q\|}$$

Here $\mathbb{S} := \{x \in \mathbb{R}^2\colon \|x\| = 1\}$ is the unit circle. Let $S^+ := \Pi(S\backslash\{p\})$ be the result of projecting $S\backslash\{p\}$. We define

$$S^- := -S^+ \qquad \text{and} \qquad S^\pm := S^+ \cup S^-$$

We call the elements of $S^\pm$ *markers*. For each point $q \in S\backslash\{p\}$, we write q^+ for $\Pi(q)$ and q^- for $-\Pi(q)$. The bijective map $\iota : \mathbb{S} \to \mathbb{S}$ with $x \mapsto -x$ is called *central inversion*.

The arcs between two neighbouring markers, visualized in Figure 3.2b, play an important role throughout this section.

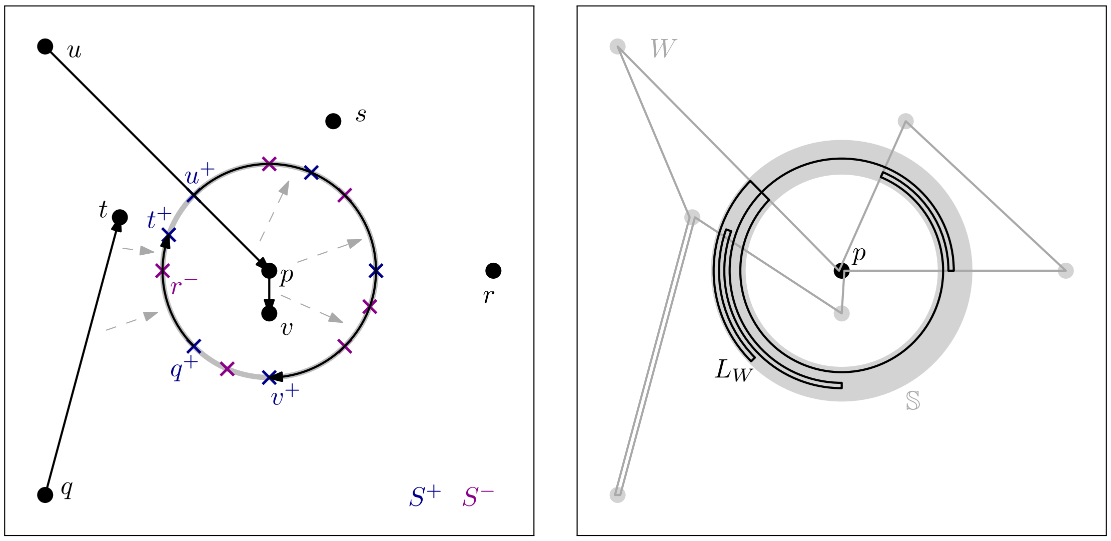
Definition 3.2. Let $x, y \in S^\pm$ be two markers. If the minor arc between x and y does not contain any other marker of $S^\pm$, we call it an *elementary arc* and write $[xy]$. Let Λ be the set of all elementary arcs.

Since we assume that S is in general position, S^+ and S^- are disjoint, and therefore $|S^\pm| = 2n-2$. This is also the number of elementary arcs. The central inversion induces a bijection on the markers $q^+ \mapsto q^-$ and $q^- \mapsto q^+$ for all $q \in S\backslash\{p\}$ and, therefore, a bijection on the set of elementary arcs Λ as well:

$$[xy] \qquad \mapsto \qquad [\iota(x)\iota(y)]$$

Next, we want to count how often the polygonal wrap goes around p in specific directions. This is well defined except for the spots where the wrap goes through p. For our purpose it is important to handle these cases as if the wrap went around the major arc side of p. We radially project the polygonal wrap to the unit circle in the following way:

Definition 3.3 (Radial loop)**.** The *radial projection* of an edge (q, r) of W with $q \neq p$ and $r \neq p$ is the sequence of all elementary arcs between q^+ and r^+ on the minor arc. For a subsequence (q, p, r) in W, the *radial projection* is the sequence of the elementary arcs between q^+ and r^+ on the major arc. If we incrementally replace every edge of W by its radial projection, we get the *radial loop* L_W. This is visualized in Figure 3.3.



(a) (q, t) is projected to $([q^+r^-], [r^-t^+])$. (u, p, v) is projected to the major arc between u^+ and v^+.

(b) The radial loop L_W is created by projecting the polygonal wrap W edge by edge to the unit circle.

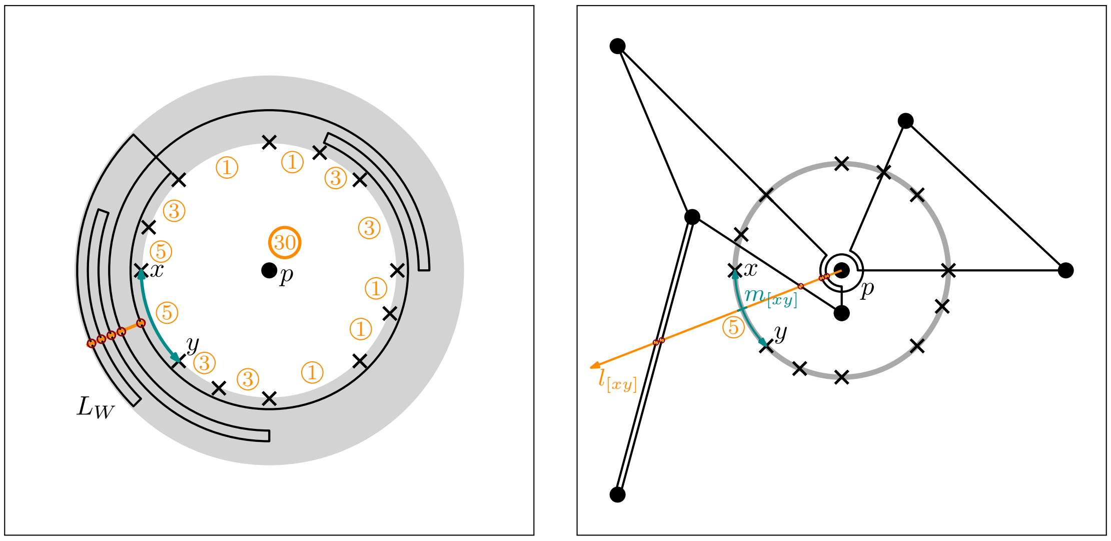
Figure 3.3: Projecting a polygonal wrap.

3.2 Coin System

Next, we introduce a coin system. We assign as many coins to an elementary arc as often as the radial loop winds around p in the direction of this elementary arc. Whenever we twang at p, we remove at least two coins. Since the current number of coins is always non-negative, the number of twangs at p is bounded by the initial number of coins divided by two.

Definition 3.4 (Coins)**.** For an elementary arc $[xy]$ let $c_W([xy])$ be the number of times $[xy]$ appears in the radial loop L_W. We assign $c_W([xy])$ many coins to $[xy]$. Furthermore, let $c_W(p)$ be the total number of coins on the unit circle centered at p, that is,

$$c_W(p) := \sum_{[xy] \in \Lambda} c_W([xy])$$



(a) $[xy]$ appears five times in L_W. Hence $c_W([xy]) = 5$. The total number of coins on p is 30.

(b) Intersections of $l_{[xy]}$ and the modified polygonal wrap.

Figure 3.4: The coin system.

We say that an edge e of W *contributes* to $[xy]$ if the radial projection of e contains $[xy]$. A subsequence $(e_0, \dots e_k)$ of W *contributes* to $[xy]$ as many coins as the number of times $[xy]$ appears in the radial projection of this subsequence.

If we modify the polygonal wrap W whenever it goes through p such that it goes around p on the major arc side, then we can also count the number of coins in a different way, as follows. For each elementary arc $[xy]$, let $m_{[xy]}$ be the centre point of $[xy]$ on the unit circle and let $l_{[xy]}$ be the half-line starting at p and going through $m_{[xy]}$. Let $c_W([xy])$ be the number of intersections of $l_{[xy]}$ and the modified polygonal wrap.

Note that $c_W([xy])$ is in both cases the number of times the modified polygonal wrap goes around p in the direction of $[xy]$. An example is shown in Figure 3.4.

3.3 The Combinatorial Upper Bound

In this section we want to prove the combinatorial upper bound on the length of twang cascades by bounding the number of twangs at each point p.

An important fact is the point symmetry of the elementary arcs. This gives us the following lemma which is used several times later on. The proof is given by a simple geometric fact which is visualized in Figure 3.5.

Lemma 3.5. *Given a fixed point $p \in S$ and two markers a, b on the unit circle centered at p. If $a \neq \iota(b)$ then the minor arc between a and b always contains at least two elementary arcs less than the major arc between a and b. The minor arc contains strictly*

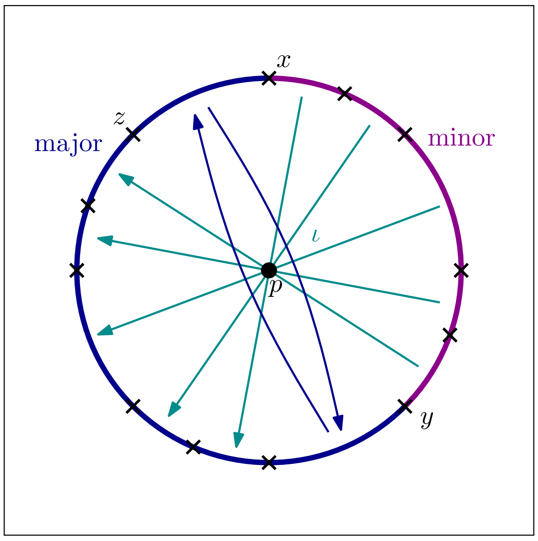


Figure 3.5: A minor arc has at least two elementary arcs less than the corresponding major arc.

less than $n-1$ arcs and the major arc strictly more than $n-1$. If $a = \iota(b)$, both arcs contain the same number of elementary arcs, namely $n-1$.

Proof. Let $a \neq \iota(b)$. Within this proof we say minor arc for the minor arc between a and b and major arc for the major arc between a and b. Every elementary arc $[xy]$ in the minor arc corresponds to exactly one elementary arc $[\iota(x)\iota(y)]$ on the direct opposite side. Therefore, $[\iota(x)\iota(y)]$ lies in the major arc, otherwise the minor arc would be longer than π, contradicting the definition of a minor arc.

The central inversion is a bijection on the elementary arcs. The major arc contains at least one elementary arc that is centrally inverted to an elementary arc lying on the major arc as well (for example, take $[ac]$ with c the marker next to a on the major arc). Hence, the minor arc contains at least two elementary arcs less than the major arc.

If $a = \iota(b)$, the central inversion gives us a bijection of the elementary arcs on both half-circles. □

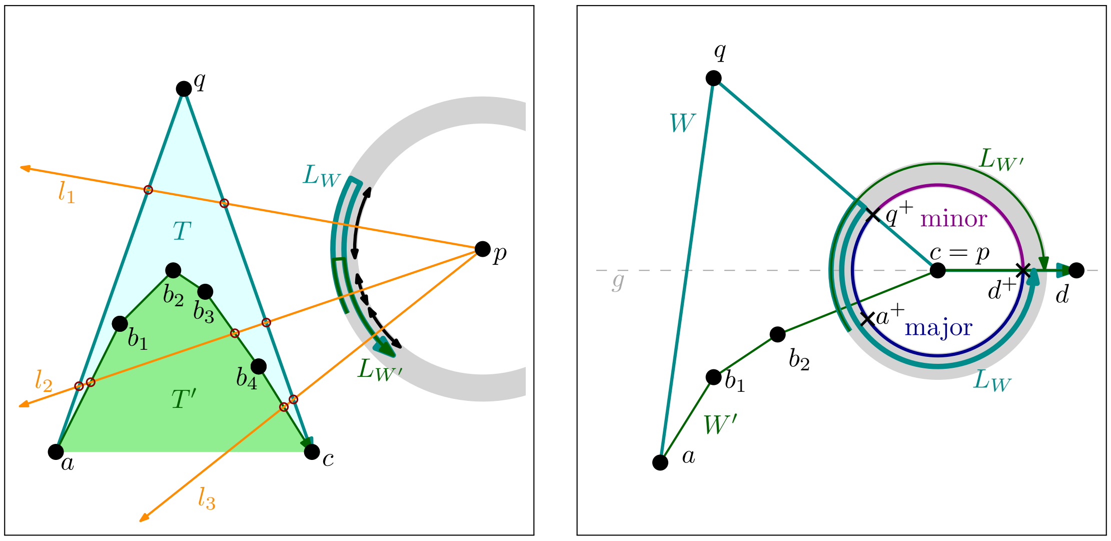
The following theorem states that the number of coins does not increase when we execute a twang. This is, of course, a very important property of the coin system; see also Figure 3.6.

Theorem 3.6. *A twang $W \to W'$ at some point $q \in S \backslash \{p\}$ never increases the number of coins at p. In other words, $c_{W'}(p) \leq c_W(p)$.*

Proof. Let $s = (a, q, c)$ be the original subsequence in W, and let $s' = (a, b_0, \ldots, b_k, c)$ be the twanged subsequence of the twanged polygonal wrap W'.

Case 1: $a \neq p \neq c$ (see Figure 3.6a).

Consider the triangle $T = (a, q, c)$. The polygonal wrap W goes from a to q and



(a) Case 1:
l_1 crosses T but not S: two coins less.
l_2 crosses both twice: same number of coins.
l_3 crosses both once: same number of coins.

(b) Case 2: The major arc flips to the other side of p.

Figure 3.6: The two cases that can occur in Theorem 3.6.

then to c. The polygon $T' = (a, b_0, \ldots, b_k, c)$ is convex and lies inside T. Furthermore, T and T' share the edge (c, a), which is not part of the two subsequences s and s'.

Now, consider an elementary arc $[xy]$ with centre point $m_{[xy]}$. The number of coins $c_W([xy])$ is given by the number of crossings of W with the half-line $l_{[xy]}$. If $l_{[xy]}$ does not cross T, both edges (a, q) and (q, c) do not contribute to $[xy]$ and so no edge of s' contributes to $[xy]$ either since $T' \subseteq T$.

Suppose $p \notin T$**:** If $l_{[xy]}$ crosses T, it needs to intersect its boundary exactly twice since $l_{[xy]}$ does not go through any points of $S \backslash \{p\}$. But T' is convex and, therefore, $l_{[xy]}$ intersects the boundary of T' at most twice.

If $l_{[xy]}$ crosses (a, c) then it intersects the subsequence s exactly once. But then it also crosses the sequence s' exactly once. Hence the same number of coins, namely one, is contributed to $[xy]$.

If $l_{[xy]}$ crosses T but not (a, c) then it crosses the sequence $s = (a, q, c)$ twice so it contributes two coins. But since T' is convex, s' cannot contribute more than two coin. Hence s' assigns at most as many coins to $[xy]$ as s does.

Suppose $p \in T$**:** Then p needs to lie inside T' as well since the twang cannot go through p. But then $l_{[xy]}$ crosses the sequence s if and only if it crosses the sequence s'. This is due to the fact that the only case when $l_{[xy]}$ does not cross s occurs when $l_{[xy]}$ crosses (a, c). But then $l_{[xy]}$ does not cross s' either. The other direction follows analogously.

Hence both are crossed exactly once or not at all. In both cases the same number of coins is contributed to $[xy]$.

Overall the number of coins of $[xy]$ cannot be increased by twanging. Therefore, $c_{W'}(p) \leq c_W(p)$.

Case 2: $a = p$ or $c = p$ (see Figure 3.6b).
The points a and c cannot be both equal to p, because a hairpin is never twanged. Suppose that $c = p$. Let d be the successor of p, and let g be the line through p and d.

If the major arc of (q, p, d) contains the major arc of (b_k, p, d), that is, when the edge (p, q) lies on the same side as the edge (b_k, p), then $(a, b_0, \dots, b_k, p, d)$ in W' contributes at most the same number of coins to p as the sequence (a, q, p, d) in W. This follows by the same arguments as in case 1 (consider p to lie outside of T).

If the major arc of (q, p, d) does not contain the major arc of (b_k, p, d), that is, when the edge (q, p) lies on the other side of g than the edge (b_k, p). Then we have the following scenario:

- (a, q) contributes one coin to every elementary arc on the minor arc of a^+ and q^+.
- (q, p, d) contributes one coin to every elementary arc on the major arc of q^+ and d^+.
- $(a, b_0, \dots, b_k, p, d)$ contributes one coin to every elementary arc on the major arc of a^+ and d^+ which consists of the minor arc of a^+ and q^+ and the minor arc of q^+ and d^+

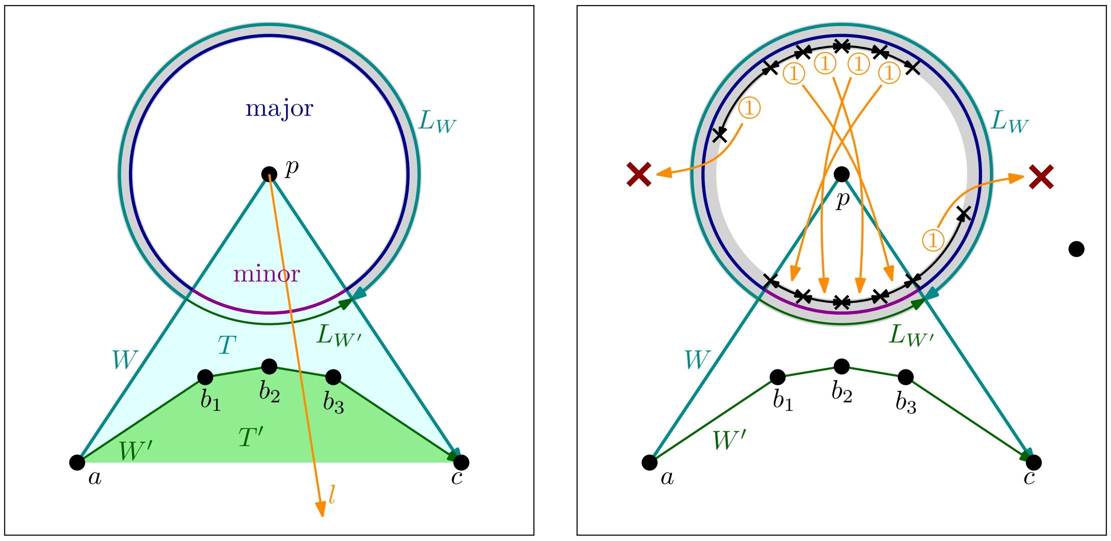
Hence, both sequences contribute to the minor arc of a^+ and q^+. Before the twang, W additionally contributes to the major arc of q^+ and d^+, but after the twang W' only contributes to the minor arc of q^+ and d^+.

By Lemma 3.5, the minor arc always contributes to fewer elementary arcs than the major arc. Therefore, the total number of contributions does not increase (and even decreases by at least two), which we wanted to show.

In both cases the number of coins at p does not increase. This completes the proof. $\square$

The next theorem considers the case of twanging at p. In order to bound the number of twangs by the number of coins, we need to make sure that the number of coins decreases each time we twang. An example is shown in Figure 3.7a.

Theorem 3.7. *Every twang $W \to W'$ at p decreases the number of coins at p by at least two. In other words, $c_{W'}(p) \leq c_W(p) - 2$.*



(a) The radial loop flips from the major arc to the minor arc.

(b) The coin movement if we twang at p.

Figure 3.7: A twang at p decreases the coins by at least two.

Proof. Let $s = (a, p, c)$ be the subsequence of W that is affected by the twang and let $s' = (a, b_0, \ldots, b_k, c, d)$ be the twanged subsequence of W'. The twang is exactly the replacement of s by s' in W.

Since S is in general position, a, p, c do not lie on one line. The sequence s induces a radial path from a^+ to c^+ on the major arc side and, therefore, contributes one coin to every elementary arc on the major arc.

The polygon $T' = (a, b_0, \ldots, b_k, c)$ lies inside the triangle $T = (a, p, c)$ and is convex. Since p is the a vertex of T, any half-line l starting at p and intersecting the interior of T must intersect (a, c). The same is true for the polygon T': Every line l intersecting T' must also intersect (a, c). Hence l crosses the sequence $(a, b_0, \ldots, b_k, c)$ at most once. This results in the fact that the twanged sequence $(a, b_0, \ldots, b_k, c)$ contributes exactly one coin to every elementary arc on the minor arc between a^+ and c^+.

By Lemma 3.5 we know that there are two elementary arcs less on the minor arc than on the major arc, which completes the proof. □

One can imagine that all elementary arcs on the major side give one of their coins to the elementary arc on the opposite side if it lies on the minor arc side. But there are at least two elementary arcs whose opposite arcs are on the major arc side as well. Therefore, we remove one of their coins from the system. How the coins move is illustrated in Figure 3.7b.

In the next step we give an upper bound on the number of coins on each point p at the beginning of a twang cascade.

Lemma 3.8. *Let P be a polygon with vertex set S, and let p be a vertex of P. Then $c_P(p) \leq n^2 - n$.*

Proof. We will show that every edge of P can contribute to a maximum of $n-1$ elementary arcs.

First, consider an edge $e = (q, r)$ of P with $q \neq p$ and $r \neq p$. The radial projection of e is the sequence of elementary arcs in the minor arc between q^+ and r^+. By Lemma 3.5, this means that the projected sequence must contain less than $n-1$ elementary arcs and, therefore, contributes less than $n-1$ coins.

Now consider a sequence (q, p, r) of two edges through p in P. The radial projection is the major arc between q^+ and r^+ and, therefore, consists of less than $2n-2$ elementary arcs. We can say that each edge contributes to at most $n-1$ edges.

Concluding, P has n edges, each contributing to at most $n-1$ edges. Therefore, $c_P(p) \leq n(n-1)$. □

Lemma 3.9. *A stretch does not add more than* $3n-3$ *coins to* p.

Proof. In this proof we argue with twangs that will not occur in a twang cascade. To do this we ignore all other edges and do not care about potential crossings.

Let (q, r) be the edge that is stretched to v. Let $(q, a_1, \ldots, a_k, v, b_1, \ldots, b_l, r)$ be the stretched sequence. If we replace the edge (q, r) by the sequence (q, r, v, q, r), we replace one edge by four edges. In other words, we add the three edges $(r, v), (v, q), (q, r)$. By doing this we add at most $3n-3$ coins to p; at most $n-1$ for every edge, as shown in the proof of Lemma 3.8.

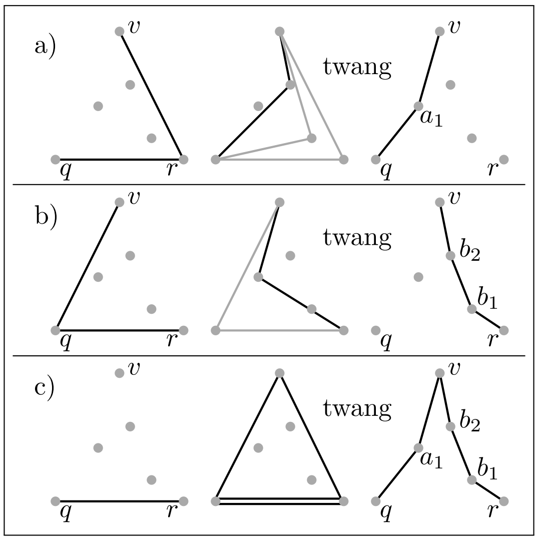


Figure 3.8: Applying a stretch from (q, r) to v: We bound the number of coins by considering the edge sequence (q, r, v, q, r) and then twanging (q, r, v) (a) and (v, q, r) (b) separately. The combination is shown in (c).

By twanging the sequence (q, r, v) multiple times, we can turn it into the sequence $(q, a_1, \ldots, a_k, v)$. This can be done by repeatedly twanging every point of the sequence between q and v that does not belong to $a_1, \ldots, a_k$, starting with v.

In the same manner we can turn the sequence (v, q, r) into $(v, b_1, \ldots, a_l, r)$.

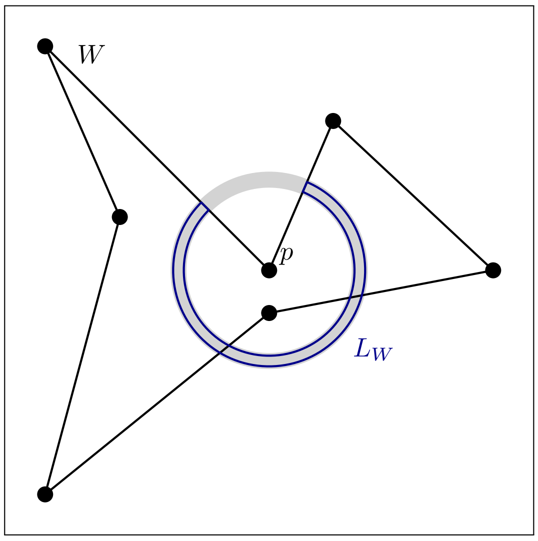
Twanging does not add any coins to the system, which is shown in Theorem 3.6. Therefore, a stretch can add at most $3n-3$ coins to p. □

In summary, there are no more than $(n^2-n)+(3n-3)=n^2+2n-3$ coins on p at the beginning of a twang cascade.

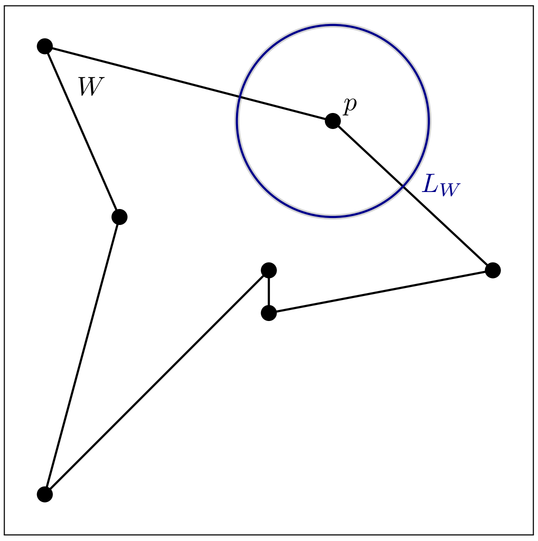
The next lemma provides a lower bound on the number of coins at each point. That the number of coins cannot be negative follows directly from the definition of the coins. This fact would be sufficient to prove an *asymptotic* upper bound of $O(n^3)$. The following lemma helps us to establish the more precise bound of $n^3/2$; see Figure 3.9.

Lemma 3.10. *Given a fixed point $p \in S$ and an arbitrary polygonal wrap W. The number of coins at p is at least $2n-2$.*

Proof. We show that, for any polygonal wrap W, the corresponding radial loop L_W goes around the whole unit circle of p, which consists of $2n-2$ elementary arcs, or that it goes over a half-circle at least twice, which contains at least $n-1$ elementary arcs by Lemma 3.5.



(a) The radial loop goes over a half-circle twice.



(b) The radial loop goes around the whole unit circle.

Figure 3.9: A lower bound on the number of coins is $2n-2$. One of the two cases above must occur.

Note that W visits every point of S at least once. So let q and r be the two neighbours of p in W. In the case that W visits p more than once, consider an arbitrary visit. Since S is in general position, q, p, r are not collinear and, therefore, L_W goes around the major arc between q^+ and r^+ for the subsequence (q, p, r).

If W closes its cycle from q to r via the major arc, this major arc is visited at least twice by L. Otherwise, if W closes its cycle from q to r via the minor arc, the whole circle is visited at least once by W. □

Summerizing, p has at most $n^2 + 2n - 3$ coins at the beginning of a twang cascade, and it cannot have less than $2n - 2$ coins in the end. Hence, at most $n^2 - 1$ coins can be removed during a twang cascade.

The following theorems summarize the results and state the upper bound on the number of twangs at each point.

Theorem 3.11. *In every forward move, the number of twangs in a twang cascade at each point p is bounded by $n^2/2$.*

Proof. Lemma 3.8 and Lemma 3.9 yield that, after a stretch, there are at most n^2+2n-3 coins on p. Theorem 3.6 gives us that the number of coins never increases. Finally, Theorem 3.7 says that the number of coins at p decreases by at least 2 for each twang at p.

Since the minimum number of coins on each point is never less than $2n - 2$ by Lemma 3.10, the number of twangs at p is bounded by $(n^2 - 1)/2$. □

Now, if we sum over all n points, we get the final result.

Theorem 3.12. *In every forward move, the length of a twang cascade is bounded by $n^3/2$.*

Proof. This follows directly from Theorem 3.11. There are n points and at each point at most $n^2/2$ twangs can occur. Hence, a twang cascade can consist of at most $n^3/2$ twangs. □

3.4 Multiple Forward Moves

We can improve our bound on the length of twang cascades a bit if we consider m forward moves in a row, which is what a random generator would do.

At the beginning we can have at most $O(n^3)$ coins in the system. Executing m forward moves means a total of m stretches, which can add up to $O(mn)$ coins to the system. Therefore, all twangs together can at most spend $O(n^3 + mn)$ coins, in other words there can be at most $O(n^3 + mn)$ twangs overall. This leaves us with an average of $O(n^3/m + n)$ twangs per forward move. For $m \geq n$ this means that there are on average $O(n^2)$ twangs in one twang cascade and for $m \geq n^2$ there are on average only $O(n)$ twangs in one forward move.

Figuratively speaking, this shows that long twang cascades can only occur when the initial polygon is very twisted and has a lot of nested pockets and when the output polygon is the opposite: Only few nested pockets and much closer to a star-shaped polygon.

3.5 Complexity of a Forward Move

Finally, we analyse the time complexity of a forward move. For this we start by studying a single twang. A twang can be computed as in Algorithm 4.

```
Algorithm 4: Twang at $b$
  Input: Polygonal wrap $W$ and a sequence $abc$.
  Output: Twang sequence $\text{sp}(abc)$.
1 Compute $X = S \cap \triangle abc$.
2 Compute $s = \partial\,\text{CH}(X\backslash\{b\})$.
3 return $s\backslash\{ac\}$
```

The shortest path from a to c inside $\triangle abc$ not crossing W lies on the boundary of the convex hull of all points in $\triangle abc$ without b but including a and c. Since S is in general position we get $\text{sp}(abc)$ when we remove the line segment ac from the boundary of this convex hull. For the triangular range query in line 1 we can use a *Simplicial Partition Tree* as data structure. It needs $O(n \log n)$ preprocessing time (once at the very beginning) and has a query time of $O(\sqrt{n} \log^c n + h)$ [BCKO08]. With h we mean the size of the output, the number of points in the triangle, and c is a constant. To compute the convex hull in line 2, we can use *Chan's Exponential Search* [Cha96], which takes $O(h \log k)$ time, k being the number of points in $\partial\,\text{CH}(X\backslash\{b\})$. Hence, we can compute a twang in $O(\sqrt{n} \log^c n + h \log k)$ time. In the worst case, h and k lie in $\Theta(n)$, which leads to a worst case running time of $O(n \log n)$ for a twang.

A stretch essentially consists of two twangs. Therefore, we have the same asymptotic running time for it. Since the length of a twang cascade is bounded by $O(n^3)$, we have an upper bound on the running time of a forward move of $O(n^4 \log n)$.

For a sequence of m forward moves we can use the improvement of Section 3.4. For $m \geq n$, the running time of all forward moves together is bounded by $O(n^4 \log n + mn^2 \log n)$. Therefore, we have an average running time per forward move of $O(n^3 \log n)$.

4 Generating Random Polygons

In this chapter we present a probabilistic algorithm generating a random polygon on S at approximately uniform distribution. We use the concept of a *Metropolis-Hasting algorithm*, which is an example of a Markov chain Monte Carlo method [LPW09]. This means that the algorithm simulates a biased random walk on the transformation graph $\mathcal{G}_S$ of forward and reverse moves.

The transition probabilities depend on the degrees of both, the origin and destination polygons, in $\mathcal{G}_S$. Hence, we have to compute the complete neighbourhood of the latter in each iteration; see Algorithm 5.

Algorithm 5: Random Walk of Forward and Reverse Moves

```
   Input: Point set $S = \{s_1, \ldots, s_n\}$, mixing time $T$.
   Output: Random polygon $P$ with vertex set $S$ under approximately uniform
           distribution.
1  Generate a starting polygon $P_0$.
2  Generate a list $L_0$ of all polygons reachable by a single forward or reverse move
   from $P_0$.
3  $t = 0$
4  while $t < T$ do
5      Choose one polygon $P'$ uniformly at random from $L_t$.
6      Generate a list $L'$ of all polygons reachable by a single forward or reverse move
       from $P'$.
7      Flip a coin with probability $\min\left\{\frac{1}{2}, \frac{|L_t|}{2|L'|}\right\}$ of heads.
8      if heads then
9          $P_{t+1} = P'$
10         $L_{t+1} = L'$
11     else
12         $P_{t+1} = P_t$
13         $L_{t+1} = L_t$
14     $t = t + 1$
15 return $P_t$
```

We call the probability in line 7 *acceptance probability* because we only update our polygon if the coin comes up heads. In this case the iteration is called *accepted.* If the coin shows tails, we keep the same polygon for the next iteration similar to a simple lazy walk. We say the transition is *rejected.*

Since the length of L is equal to the degree of P, the coin flips in line 7 bias the walk to stay away from polygons of high degrees, which would otherwise be more likely than polygons of low degrees.

4.1 Correctness

The following theorem shows that the stationary distribution of the random walk, that is the distribution where a step does not change the probabilities, is indeed uniform.

The steps $(P_0, P_1, \dots)$ of the generated walk form a *Markov chain*, that is, the distribution of P_{t+1} only depends on the distribution of P_t and not on the steps before $P_0, \dots, P_{t-1}$.

Theorem 4.1. *The Markov chain $(P_0, P_1, \dots)$ generated in Algorithm 5 is irreducible and, therefore, has a unique stationary distribution π.*

Furthermore, this distribution is uniform, that is, $\pi(P) = \frac{1}{r}$ for all polygons P with vertex set S, where $r = |\mathcal{P}_S|$ is the number of all polygons on S.

Proof. Let T be the transition matrix of the Markov chain. This means that $T(P, P')$ is the probability to go from P to P' in one iteration. Note that it might be possible that there exists a forward or reverse move that does not change the polygon and, therefore, P' could be equal to P_t. This might happen when there is a loop in $\mathcal{G}_S$. Because of this we consider two cases.

1. First, let $T_{\text{acc}}(P, P')$ be the probability that the transition from P to P' is accepted. If P' and P are not adjacent, $T_{\text{acc}}(P, P')$ is zero. Otherwise, P' is chosen uniformly at random from all neighbours of P in line 5, that means with a probability of $\frac{1}{\deg(P)}$. Afterwards, P' is accepted with a probability of $\min\left\{\frac{1}{2}, \frac{|\deg(P)|}{2|\deg(P')|}\right\}$. Hence, we have

$$T_{\text{acc}}(P, P') = \frac{1}{\deg(P)} \cdot \min\left\{\frac{1}{2}, \frac{\deg(P)}{2\deg(P')}\right\} = \min\left\{\frac{1}{2\deg(P)}, \frac{1}{2\deg(P')}\right\}.$$

   Because of the symmetry on the right hand side we have $T_{\text{acc}}(P, P') = T_{\text{acc}}(P', P)$.

2. Second, let R_P be the probability that we reject the transition and stay at polygon P. We have

$$R_P = 1 - \sum_{P' \in \mathcal{G}_S} T_{\text{acc}}(P, P')$$

Both together give us the transition matrix T:

$$T(P, P') = \begin{cases} T_{\text{acc}}(P, P') & \text{if } P' \neq P \\ T_{\text{acc}}(P, P) + R_P & \text{if } P' = P \end{cases}$$

Damian et al. showed that the transformation graph is connected and since the transition of two adjacent polygons is always positive, we can reach every polygon P' from every other polygon P with positive probability in a finite number of steps. This means that the Markov chain is irreducible. Markov chain theory says that in this case there is a unique stationary distribution [LPW09]. This concludes the proof of the first part.

Let $\pi \equiv \frac{1}{r}$ be the uniform distribution. In order to prove that π is stationary, we have to show that if there is a π-distributed random polygon and we apply one random transition, it still has distribution π. In other words $\pi \cdot T = \pi$.

With $T_{\mathrm{acc}}(P, P') = T_{\mathrm{acc}}(P', P)$ we get:

$$\begin{aligned}
(\pi \cdot T)(P) &= \pi(P) \cdot R_P + \sum_{P' \in \mathcal{P}_S} \pi(P') \cdot T_{\mathrm{acc}}(P', P) \\
&= \frac{1}{r}\left(1 - \sum_{P' \in \mathcal{G}_S} T_{\mathrm{acc}}(P, P')\right) + \sum_{P' \in \mathcal{P}_S} \frac{1}{r} \cdot T_{\mathrm{acc}}(P', P) \\
&= \frac{1}{r}\left(1 - \sum_{P' \in \mathcal{G}_S} T_{\mathrm{acc}}(P, P') - T_{\mathrm{acc}}(P', P)\right) \\
&= \frac{1}{r} \\
&= \pi(P)
\end{aligned}$$

Therefore, π is stationary, which concludes the proof. □

By the *Convergence Theorem of Markov Chains* [LPW09], every irreducible and aperiodic random walk converges to its stationary distribution. The irreducibility follows from the connectivity of $\mathcal{G}_T$ and the aperiodicity is given by the laziness of the random walk. This means that with a probability of at least $\frac{1}{2}$ the iteration is rejected and, therefore, the walk remains at the same polygon. The longer we run the algorithm the better the approximation will be.

4.2 Complexity of One Iteration

There are still some open questions concerning the running time of each iteration of this algorithm. Surely, line 6, where the neighbourhood of P' is generated, is the bottleneck of each iteration.

Certainly, the running-time to generate all neighbours reachable by a single forward move is bounded by $O(n^6 \log n)$, since there are at most n^2 vertex-edge-pairs for the initial stretch and we can simulate one forward move in $O(n^4 \log n)$. We do not consider our improvement from the end of Section 3.4. The argument does not work if we have reverse moves in our sequence as well since they can add a lot more coins to the system than forward moves do. The upper bound already suggests that this algorithm might not be practical for real-world application, but at least we could show that it runs in polynomial time.

Unfortunately, we had to add reverse moves to make sure that the transformation graph is connected. But we do not have an algorithm that efficiently computes all possible reverse moves from a given polygon P and it is possible that such an algorithm does not exist.

4.3 Mixing Time

The final point we want to discuss is the mixing time. We need to know how many iterations need to be executed in order to get close to the stationary distribution. Unfortunately, it is very difficult to show a specific upper bound on the mixing time,. Therefore, we only give a short introduction to this subject.

Definition 4.2. Let

$$d(t) := \max_{P \in \mathcal{P}_S} \|T^t(P, \cdot) - \pi\|$$

be the distance between the distribution of P_t and the stationary one in some reasonable distribution norm, for example in the *total variation norm*:

$$\|\mu - \nu\| := \frac{1}{2} \sum_{P \in \mathcal{P}_S} |\mu(P) - \nu(P)|$$

Then the mixing time is defined by

$$t_{\text{mix}}(\varepsilon) := \min\{t \colon d(t) \leq \varepsilon\}.$$

That means $t_{\text{mix}}(\varepsilon)$ is the smallest t such that the distribution after t iterations only differs by at most ε from the uniform distribution (in the total variation norm). Since $d(t)$ is a non-increasing function, we need to choose the input parameter T such that $T \geq t_{\text{mix}}(\varepsilon)$. It would be of high interest to understand whether $t_{\text{mix}}(\varepsilon)$ is polynomially bounded in n for a fixed ε.

There is a wide mathematical field trying to answer questions of this type; Randall [Ran06] gives a good overview. Some techniques use eigenvalues of the transition matrix and the diameter of the graph to bound the mixing time. But there is no complete theory to answer this question, and bounds on mixing times in other contexts are shown in very unique ways.

By looking at other examples like a random walk on an n-dimensional hypercube, which has a mixing time of $O(n \log(\frac{1}{\varepsilon}))$ [Ran06], we conjecture that our random walk is *rapidly mixing*, that means the mixing time is bounded polynomially in n and $1/\varepsilon$.

	n-dimensional hypercube	transformation graph $\mathcal{G}_S$
number of vertices	2^n	$O(4.6^n)$
average degree	n	$O(n^2)$
diameter	n	$O(n^2)$

Table 4.1: The average degree and the diameter of the n-dimensional hypercube and of our transformation graph differ only by a linear factor.

Table 4.1 shows the structural similarities of the n-dimensional hypercube and our transformation graph $\mathcal{G}_S$ which underlines our conjecture. But since $\mathcal{G}_S$ is only given implicitly we are not able to prove this.

5 Open Problems and Conclusion

As we have seen in the last chapter there are still several problems that need to be solved in order to obtain an efficient random generator for polygons.

1. Is there an efficient way to compute reverse moves?
2. Is the transformation graph $\mathcal{G}_S$ also connected by forward moves only?
3. Is there a combinatorial upper bound on the mixing time $t_{\text{mix}}(\varepsilon)$?
4. Is the random walk rapidly mixing? In other words, is the mixing time bounded polynomially in n and $1/\varepsilon$?

We assume the first question to be answered with no and the last two to be answered with yes. The second question is the most interesting one. If it had a positive answer, we would not need to compute any reverse moves in our algorithm. But in order to do so we either need to show that we can simulate a reverse move by a sequence of forward moves or we directly show the connectivity. Unfortunately, twang cascades are hard to predict and we barely know any of their properties, except that they do not increase the size or number of pockets. Nonetheless, we cannot search for isolated vertices in order to give a negative answer, as has been done for 2-flips [Hou96]. The problem is that we can apply a lot of forward moves to any polygon.

However, if we answered the second question with yes, then there would be a good chance for our Metropolis algorithm, using forward moves only, to run in polynomial time. In this case it might even be applicable in practice since Damian et al. [DFOR10] show in their tests that twang cascades are very short most of the time (even sub-linear). Our upper bound on the length of twang cascades of $O(n^2)$ and the resulting time complexity of a forward move of $O(n^5 \log n)$ will guarantee that there are no iterations with exponential running time.

Bibliography

[AH98] Thomas Auer and Martin Held. RPG – heuristics for the generation of random polygons. *Proc. 8th Canad. Conf. Comput. Geom (CCCG'98)*, pages 38–44, 1998.

[BCKO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd edition, 2008.

[Cha96] Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Computational Geometry*, 16:361–368, 1996.

[DFOR10] Mirela Damian, Robin Flatland, Joseph O'Rourke, and Suneeta Ramaswami. Connecting polygonizations via stretches and twangs. *Theory of Computing Systems*, 47(3):674–695, 2010.

[GNT00] Alfredo Garcia, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Computational Geometry*, 16(4):211–221, 2000.

[HHH02] Carmen Hernando, Michael E. Houle, and Ferran Hurtado. On local transformation of polygons with visibility properties. *Theoretical Computer Science*, 289(2):919–937, 2002.

[Hou96] Michael E. Houle. On local transformation of polygons. In *Proc. Computing: 2nd Australasian Theory Symp. (CATS'96)*, volume 18 of *Australian Comput. Sci. Commun.*, pages 64–71, 1996.

[LPW09] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times.* American Mathematical Society, 2009.

[Ran06] Dana Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science and Engineering*, 8(2):30–41, 2006.

[ZSSM96] Chong Zhu, Gopalakrishnan Sundaram, Jack Snoeyink, and Joseph S.B. Mitchell. Generating random polygons with given vertices. *Computational Geometry*, 6(5):277–290, 1996.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 1. Februar 2016

. . . .
Leon Sering