ALGORITHMICALLY GENERATED PANTS DECOMPOSITIONS OF COMBINATORIAL SURFACES

NICHOLAS HAGEDORN

Abstract. We describe two algorithms that efficiently find a pants decomposition of a surface model given by taking a $2n$-sided regular polygon with unit length sides and gluing all the edges in pairs. The first algorithm closely follows Buser's proof that any surface S of genus $g \geq 2$ has a pants decomposition of length at most $C(g \operatorname{Area}(S))^{1/2}$ for some constant $C > 0$. The second algorithm finds a pants decomposition by estimating the size of the largest embedded ball at a randomly chosen point on the surface. We prove that the first algorithm always gives a pants decomposition of size at most $C'g$ for some constant $C' > 0$ in $O(ng + g^3)$ time. Empirically, we observe that the second algorithm outputs much shorter pants decomposition than the first.

1. Introduction

This paper aims to find an efficient algorithm to decompose a closed Riemannian surface into simpler pieces called pairs of pants. A pair of pants is any subset of the surface that is topologically a sphere punctured with three holes. If $\gamma_1, \gamma_2, \dots \gamma_{3g-3}$ are disjoint embedded loops on a surface S of genus g, so that each connected component of $S - \cup_{i=1}^{3g-3} \gamma_i$ is a pairs of pants, then we say that these loops form a *pants decomposition* of S. If the surface is endowed with a Riemannian metric, then we can compute the length of each loop. The *length* of a pants decomposition is the maximum length of a loop in the decomposition. An example of a pants decomposition for a genus two surface is shown in Figure 1.

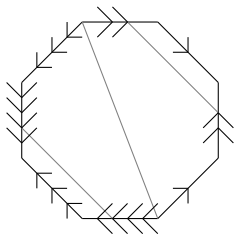


Figure 1. A pants decomposition of a genus two surface. Arrows on edges indicate the edge identifications. The three gray lines indicate curves that collectively decompose the surface into two pairs of pants.

Definition. The *Bers' constant* of a Riemannian surface S, denoted by $\mathfrak{B}_S$, is the smallest length of a pants decomposition of S.

Date: April 29, 2024.
2020 *Mathematics Subject Classification.* 14J29, 51F05.
Key words and phrases. Bers' constant, pants decomposition.

Since the Bers' constant $\mathfrak{B}_S$ measures how difficult it is to cut a surface S into simpler surfaces, it can indicate how complicated the geometry of S is. Understanding the worst case behavior of $\mathfrak{B}_S$ for surfaces with a fixed area and genus is a big open problem in the geometry of surfaces. We briefly give some background on what is known about $\mathfrak{B}_S$ before describing our results.

Each loop in the pants decomposition must be non-contractible on the surface. It is also interesting to show that any Riemannian surface S of non-zero genus has one non-contractible loop of length bounded by a function of its area. The smallest length of a non-contractible loop of S is called the systole of S and is denoted by $\operatorname{sys}(S)$. In 1949, Loewner proved that every torus T satisfies $\operatorname{sys}(T)^2 \leq \frac{2}{\sqrt{3}} \operatorname{Area}(T)$ [9] using the uniformization theorem. Later, Gromov proved much more general versions of Loewner's systolic inequality. See for instance [6] for more results about systoles. For example, Gromov proved the following theorem about systoles of surfaces.

Theorem 1.1. *[5] Any Riemannian surface S with genus $g \geq 1$ satisfies,*

$$\operatorname{sys}^2(S) \leq C \frac{\log(g)^2}{g} \operatorname{Area}(S)$$

for some constant $C > 0$ independent of S.

To find an upper bound on the systole requires finding one short loop on the surface, but to upper bound the Bers' constant requires finding several short loops, each of which are pairwise disjoint and non-homotopic. So the Bers' constant is much more difficult to estimate than the systole. Buser began proving results concerning the Bers' constant in the 80's. In 1981, he showed that a hyperbolic surface S with genus $g \geq 2$ satisfies $C^{-1} g^{1/2} \leq \mathfrak{B}_S \leq C g \log(g)$ for a constant $C > 0$ independent of S [1]. Later, Buser and Seppälä [3] proved that $\mathfrak{B}_S \leq Cg$, and Parlier [8] found a good bound on the constant, showing,

$$\mathfrak{B}_S \leq 4\pi(g-1) + 4\left(\cosh^{-1} \frac{1}{\sqrt{2}\sin(\frac{\pi}{12g-6})}\right).$$

The $\sqrt{g}$ lower bound on $\mathfrak{B}_S$ is sharp up to a constant for some hyperbolic surface S, and Buser [3] conjectured that this should also be the behavior of the upper bound on $\mathfrak{B}_S$ for any hyperbolic S. For a general Riemannian surface S of genus g, Buser also proved that $\mathfrak{B}_S \leq C(g \operatorname{Area}(A))^{1/2}$ [2]. This matches the known bound for a hyperbolic surface, since a hyperbolic surface of genus g has area $4\pi(g-1)$.

To prove an upper bound on $\mathfrak{B}_S$, Buser developed a procedure to successively find disjoint and non-homotopic loops on S. He then proved an upper bound on the length of these loops. See [2] for an exposition of this procedure. However, it is unknown how the loops in Buser's procedure actually behave; the procedure could potentially output much shorter loops than the upper bound that Buser proved. Furthermore, turning Buser's procedure into an algorithm that can be run on a computer poses some difficulties as it involves steps which are difficult to make algorithmic. For instance, the procedure involves shrinking a loop to the smallest possible length within its homotopy class.

In [4], Colin de Verdiere, Hubard, and de Mesmay discuss how to adapt Buser's procedure for a discrete surface model. Their discrete model is a triangulated surface, where loops are supported on the graph dual to the triangulation, and lengths of loops

are measured by the number of edges from the dual graph that they contain. They found that for a surface of genus g with n triangles, there is algorithm which finds a pants decomposition of length at most $C\sqrt{gn}$ in $O(gn)$ time for some constant $C > 0$. This matches Buser's upper bound estimate. Although more algorithmic in nature than Buser's procedure, their algorithm also contains some difficult steps to write into a computer algorithm, like doing surgery on curves. Our aim in this paper to present two simpler algorithms for finding pants decompositions inspired by Buser's procedure and to compare empirically the performance of the two algorithms.

We introduce a robust model of a genus g surface with area roughly g and constant diameter. The surfaces S in our model will be constructed by taking a regular polygon with $2n$ sides of unit length, and gluing together pairs of edges in an orientation preserving way. Generally, when we run our algorithm, we will pair the edges at random. In Theorem 6.1 of the appendix, we show that if we identify all the edges of a $2n$-gon in pairs uniformly at random, then the expected value of the genus of the resulting surface is at least $\frac{1}{6}n - \tilde{C}\ln n + \frac{1}{2}$ for constant $\tilde{C} > 0$, meaning the expected value of the genus is roughly linear with respect to n. Each loop in our pants decomposition model can be viewed as a collection of straight segments on this polygon, as in Figure 1. For some $k \leq 3g-3$, suppose we have already found the first $(k-1)$ loops $\gamma_1, \gamma_2 \ldots \gamma_{k-1}$ of our pants decomposition and would like to choose γ_k. The first $(k-1)$ loops cut the $2n$-sided polygon into several smaller polygons that we will call connected parts.

Now we describe the main idea in our first algorithm for finding the k^{th} loop γ_k for a pants decomposition of S, assuming $k > 1$. Let S' be a connected component of $S - \cup_{i=1}^{k-1}\gamma_i$ which is not a pair of pants. If S' contains multiple boundary components, we find a short curve $\alpha \subset S'$ between two loops γ_i and γ_j on the boundary of S'. Then the composition $\gamma = \gamma_i \circ \alpha \circ \gamma_j \circ \alpha^{-1}$ is another loop. We perturb this γ so that is is disjoint from γ_i and γ_j. Note that γ not contractible, since otherwise γ_i would be homotopic to γ_j. We choose γ_k to be this perturbation of γ. If S' rather has one connected boundary component, we create a surface $\tilde{S}$ by contracting said boundary component to a point. We find a curve γ on $\tilde{S}$ by the process used to find approximations of the systole of a surface. We choose γ_k to be the corresponding curve on S'.

Now we state the theorem about this algorithm.

Theorem 1.2. *Given a surface S from our model, the first algorithm outputs a pants decomposition of length at most Cg in $O(ng + g^3)$ time, where $C > 0$ is some constant.*

Now we describe the main idea of our second algorithm. If we have found $(k-1)$ loops so that each connected component of $S - \cup_{i=1}^{k-1}\gamma_i$ has genus zero, we find the rest of the loops of our pants decomposition using the first algorithm. Otherwise, we find the k^{th} loop in the following way. Let S' be a connected component in $S - \cup_{i=1}^{k-1}\gamma_i$ that has non-zero genus. Consider the graph of connect parts G described as follows: Each connected part in S' corresponds to a vertex of G, and each pair of connected parts that share a boundary in S' correspond to an edge in G. Notice that each simple loop in G corresponds to a loop in S', which consists of straight segments in some connected parts. Our goal is to find a short loop that be added to our pants decomposition by finding a short loop in G. To do this pick some starting vertex $v \in G$ and start growing a tree from v. Suppose after t steps, we have a sub-tree T_t of G. Choose one of the leaves of T_t closest to v, call it w. First suppose there is some edge e which contains w, so that $T_t \cup e$ contains a loop in G. If this loop corresponds to a loop in S' that can be

added to the pants decomposition, then let γ_k be this loop. Otherwise, let E_w be all the edges that contain w, but no other vertex of T_t and define $T_{t+1} = T_t \cup E_w$. Eventually, we will find a loop that can be added to our pants decomposition.

Theorem 1.3. *Given a surface S from our model, the second algorithm outputs a pants decomposition in $O(L(S)g^3)$ time, where $L(S)$ is the maximum number of identified edges that a curve in the resulting pants decomposition intersects.*

Empirically, for random surfaces in our model with genus $g \leq 120$, $L(S)$ seems to behave like $\tilde{C}g$, where $\tilde{C}$ is some constant smaller than 1. In fact, the second algorithm seems to almost always find a shorter pants decomposition than the first algorithm, though we are not able to prove this.

We now mention a few open questions.

(1) Can we prove that the length of the pants decomposition that the second algorithm outputs is at most Cg, for some small constant $C < 1$?
(2) Given our surface model, is there some algorithm that finds a pants decomposition of length at most Cg for some constant $C > 0$, in $\mathcal{O}(g^2)$ time?
(3) Given a pants decomposition of a surface we can create its pants decomposition graph as follows. Each vertex of a graph corresponds to a pair of pants on the surface. Two vertices are connected by an edge in the graph if the corresponding pair of pants intersect in some loop of the pants decomposition. The pants graph of the decomposition that first algorithm outputs will, roughly, look like a path. On the other hand, the pants graph of the decomposition that second algorithm outputs could be much more interesting. Does this pants graph often have a large spectral gap for surfaces obtained by randomly identifying pairs of edges in a $2n$-gon?

The paper is organized as follows. We describe combinatorial surfaces and the paper's notation, in addition to proving a few useful results about these surfaces, in Section 2. We detail our first algorithm in Section 3 and our second in Section 4. We analyze both algorithms' performance in Section 5. Section 6 is an appendix, where we estimate the expected value of the genus for the surfaces in our model.

2. Combinatorial surfaces

In this section we describe our combinatorial surfaces in more detail and begin discussing some algorithms to compute their properties. The concept is motivated by the needed for an efficient way to discretely model a random surface and its pants decomposition. We begin with an orientable $2n$-gon for some integer $n \geq 4$ and pair all the edges of the polygon in some way. When all the paired edges are identified in an orientation preserving way, we obtain an orientable surface. Next, we distinguish a certain set of loops on this surface. The loops have to satisfy two conditions. First, these loops must form a subset of a pants decomposition. In particular, they are disjoint and non-homotopic. Second, each loop must be represented by a finite set of straight line segments that cross our $2n$-gon between two of its edges, but do not intersect the vertices of the $2n$-gon. The line segments cut up our $2n$-gon into smaller polygons, which we will refer to as connected parts. This leads to the following definitions.

Definition. A combinatorial surface S is given by a finite set of oriented polygons along with several edge pairings, which we get from the above procedure. When all the

paired edges are identified in an orientation preserving way, the result is a disjoint set of oriented surfaces with boundary. We say that S is a connected combinatorial surface if, after gluing the edges, we get only one connected component.

We refer to each polygon in S as a connected part, and let $|S|$ denote the number of connected parts in S. We write $c \in S$ to mean that S has a connected part c, and let $|c|$ denote the number of edges of c.

Definition. The distinguished edges of a combinatorial surface S are those which are not paired to any other edges. In particular they are subsets of the distinguished loops in the above procedure. When all the paired edges of the connected parts are identified, the distinguished edges lie on the boundaries of the resulting surfaces. We also refer to distinguished loops as cuts or boundary components of S.

Next, we describe the data structure that stores the information of our combinatorial surfaces and its distinguished loops. We assign an integer label to every edge of a connected part. Each non-negative integer label represents an edge pairing. That is, if two edges are both assigned the same non-negative integer, then the two edges are paired. Distinguished edges are labeled with negative labels, and all edges in a distinguished loop have the same label. So we can speak of a label of a distinguished loop. The data structure that holds this information is a two-dimensional array, where each array corresponds to a connected part and lists the integers corresponding to the edges, starting from an arbitrary edge and moving clockwise. We only care about the order of the edges of a connected part up to rotation, and we do not care about the permutation of the connected parts. Hence we can store these values in a two-dimensional array, arbitrary choosing the starting positions of edges in connected parts and permutation of the connected parts.

We represent one-dimensional arrays with vectors like $c = \begin{pmatrix} 0 & 1 \end{pmatrix}$ or with sets like $c = \{0, 1\}$. Similarly, a two-dimensional array is represented by a vector or set whose entries are vectors or sets. Note that the arrays in the entries do not need to have the same number of components. When storing information of a combinatorial surface or connected part, we consider such arrays as equivalent up to rotation. So, $\begin{pmatrix} \begin{pmatrix} 0 & 1 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 1 \end{pmatrix} \end{pmatrix} \sim \begin{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 1 \end{pmatrix} \end{pmatrix}$. We use $|c|$ to count the size of an array or set c. The notation S always refers to an orientable combinatorial surface. If c is an array, then $c(i)$ is the $(i+1)$th entry in c. If c does not have an $(i+1)$th entry, then $c(i)$ is the $(j+1)$th entry in c where j is the remainder after i is divided by $|c|$.

Here are a few examples of combinatorial surfaces given by two-dimensional arrays.

Example 2.1. $S = \begin{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix} \end{pmatrix}$ is a combinatorial surface with one connected part. When we identify the paired edges we get a torus.

Example 2.2. $S = \begin{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 & 2 & 3 & 2 & 3 \end{pmatrix} \end{pmatrix}$ is a combinatorial surface with one connected part. When we identify the paired edges we get a genus 2 surface. Figure 1 shows three loops on this surface. After we cut the surface along these loops we obtain a new combinatorial surface S' with four connected parts. This can be represented as,

$$S' = \begin{pmatrix} \begin{pmatrix} 0 & 1 & -1 & 1 & -2 \end{pmatrix} \\ \begin{pmatrix} 2 & 0 & 2 & -3 \end{pmatrix} \\ \begin{pmatrix} 3 & 4 & -4 & 4 & -5 \end{pmatrix} \\ \begin{pmatrix} 5 & 3 & 5 & -6 \end{pmatrix} \end{pmatrix}.$$

Next, we discuss some metric considerations for our combinatorial surfaces. The metric on a combinatorial surface S is given by letting the initial $2n$-gon be regular with unit side length. In particular, the area of such a surface is $n\frac{\sqrt{3}}{2}$ and its diameter is at most $\sqrt{3}$. Suppose γ is one of the distinguished loops on a combinatorial surface. Let $L_S(\gamma)$ be the number of distinguished edges in γ that do not belong to a connected part with three sides. We can also define L_S of a distinguished loop using the data structure perspective.

Definition. For a negative label b and connected part $c \in S$, let $I_c(b)$ count the number of times b appears in c. Because c can store the same element multiple times, $I_c(b)$ can be greater than one. Then define $L_S(b) = \sum_{c\in S, |c|\geq 4} I_c(b)$.

Our algorithms will find a pants decomposition of any combinatorial surface S with control on L_S of each distinguished loop. To relate this to the length of a pants decomposition under our metric on S, we make use of the following lemma.

Lemma 2.3. *If $\{\gamma_i\}_{i=1}^m$ is a set of distinguished loops on S from our model, then for any $\varepsilon > 0$ there is another set of loops $\{\gamma_i'\}_{i=1}^m$ on S so that,*

(1) γ_i' is homotopic to γ_i for each $1 \leq i \leq m$;
(2) $\{\gamma_i'\}_{i=1}^m$ are all disjoint;
(3) $\text{length}(\gamma_i') \leq 2L_S(\gamma_i) + \varepsilon$ for each $1 \leq i \leq m$.

Proof. For each $1 \leq i \leq m$, we create γ_i' as follows. The $\{\gamma_i\}_{i=1}^m$ cut S into a certain number of connected parts. First consider each three-sided connected part c where $\gamma_i \in c$. We can view c as a triangle in our $2n$-gon. Since the distinguished loops are disjoint, c contains contains some vertex v from the $2n$-gon. Let v_1, v_2 be the other two vertices of the connected part. Additionally, let l_i be the number of three-sided connected parts that γ_i passes through. Pick $\tilde{v}_1 \in vv_1$ and $\tilde{v}_2 \in vv_2$ such that they are each precisely $\varepsilon/2l_i$ away from v. Inside the $2n$-gon, add the segment $\tilde{v}_1\tilde{v}_2$ to γ_i'. We repeat this procedure for each such c.

Now consider each connected part c where $\gamma_i \in c$ and c doesn't have precisely three sides. Inside of c, consider each time γ_i passes through c, connecting two endpoints y_1 and y_2. If y_1 connects γ_i to a three-sided connected part, then let $\tilde{y}_1$ be the point along the same edge in c such that it connects to the segment of γ_i' inside the other connected part. If not, let $\tilde{y}_1 = y_1$. Do the same procedure to create $\tilde{y}_2$. Now add the straight line between $\tilde{y}_1$ and $\tilde{y}_2$ to γ_i'. Again repeat this procedure for each such c.

We now prove γ_i' is homotopic to γ_i. Consider a segment of γ_i' in a connected part c. If c was a triangle, then each of the endpoints of the segment have been shifted from the segment of γ_i in c. However, to each neighboring connected part c_0, if c_0 is not three-sided then the corresponding points of γ_i have also been shifted so that γ_i' is connected. If c_0 is three-sided, then note that the vertex v of c and c_0 is the same vertex. Then both points are precisely the same distance from that vertex along the same line segment, so they line up regardless. Now if c is not a triangle and neighbors a triangle, this case has already been covered. Lastly, if c is not a triangle and neighbors a connected part that also is not a triangle, the endpoints have never been shifted from γ_i and so are still connected. Thus γ_i' is still a loop. As each of the individual segments still connect to each other in the same order and each are homotopic to the corresponding segments in γ_i, γ_i' is homotopic to γ_i.

Now repeat all of the above for each $\gamma_i \in \{\gamma_i\}_{i=1}^m$. We already have the first claim. The second claim follows as well. Since no two segments start at the same edge in a connected part and the endpoints stay within the same edges, their order in each connected part stays the same. Now the segments are then only straightened. If two straight lines in a polygon cross, then there is no way for homotopic curves with the same endpoints to be disjoint. Since $\{\gamma_i\}_{i=1}^m$ are disjoint, $\{\gamma_i'\}_{i=1}^m$ are disjoint as well. Lastly, the third claim follows from the fact that a straight line across each connected part has length at most two, and the sum of the length of segments that cross three-sided connected parts adds up to at most ε. □

We now give an efficient way of finding the genus of a combinatorial surface S from its two-dimensional array.

Lemma 2.4. *Let S be a combinatorial surface. After we identify the paired edges, we obtain a surface whose genus g is given by,*

$$g = (2 - b + t - v + e - f)/2$$

where f is the number of connected parts, b is the number of boundary components, t is the total number of edges that lie on the boundary components, v is the number of vertices in S after identifying the paired edges, and e is the number of edges in S after identifying the paired edges that do not lie on a boundary component.

Proof. The formula for the Euler characteristic is $2 - 2g - b = v - E + f$ where E is the total number of edges. Now $E = e + t$ so $2 - 2g - b + t = v - e + f$ and $g = (2 - b + t - v + e - f)/2$. □

We now consider the computational complexity of computing the genus of a combinatorial surface.

Definition. For a combinatorial surface S we define the total number of edges to be,

$$n_S = \sum_{c \in S} |c|.$$

We also define b_S to be the number of boundary components of S. That is,

$$b_S = |\{x : x < 0 \text{ and there exists } c \in S \text{ such that } x \in c\}|.$$

Lemma 2.5. *For a combinatorial surface S, the value b_S can be computed in $\mathcal{O}(n_S)$ time.*

Proof. Create a set. Iterate through all $x \in c \in S$. In each iteration, add x to the set if $x < 0$. The length of the set is b_S. Each iteration takes constant time and there are n_S iterations, so the computation takes $\mathcal{O}(n_S)$ time. □

Proposition 2.6. *The genus of a combinatorial surface S can be computed in $\mathcal{O}(n_S)$ time.*

Proof. Lemma 2.4 means that we know the genus if we can compute the following values:

(1) The number of unique boundary components that border S.
(2) The number of total edges that border S that are part of boundary components.
(3) The number of unique vertices of S.

(4) The number of faces of S.
(5) The total number of edges that border S.

Values (2) and (5) can be computed by simply checking every value in S's information once. This can be done in one $\mathcal{O}(n_S)$-time pass through S. Value (1) can be found from Lemma 2.5, and (4) is the number of connected parts in S, which is $|S|$. We can find these two values in $\mathcal{O}(n_S)$ time as well. Hence it is enough to show we can find the number of unique vertices in $\mathcal{O}(n_S)$ time.

We can now construct a graph G whose vertices are the same as the vertices of S. Two vertices in G are connected by an undirected edge if an edge pairing in S implies that the two vertices are equivalent. Then two vertices are unique if there exists no path in G between them. The graph G has n_S vertices and n_S edges, so a breadth-first search algorithm can be run in $\mathcal{O}(n_S)$ time to find the number of unique vertices. Because there are $\mathcal{O}(n_S)$ edges, it takes $\mathcal{O}(n_S)$ time to construct the graph. Hence we can find the genus in the claimed time complexity. □

The results of this section can be summarized as follows: We introduce an algorithmic way to store combinatorial surfaces and show that this approach allows us to quickly find upper bounds on the lengths of pants decompositions and compute the genus of the surfaces.

3. The First Pants Decomposition Algorithm

In this section, we develop a relatively efficient algorithm for finding the pants decompositions of orientable combinatorial surfaces. The idea of the algorithm is based on Buser's theoretical algorithm [2]. However, the details are quite different given the discrete setting of the algorithm.

Lemma 3.1. *Let v be a curve on a combinatorial surface S that passes through connected parts of S a total of l times. Let S' be the result of cutting S along v. Then $n_{S'} = 4l + n_S$.*

Proof. Curve v can be divided into l segments, each of which connects two edges of a connected part. Say the ith such segment connects the x_ith and y_ith edge of connected part c_i. We can write,

$$c_i = \begin{pmatrix} e_{i,1} & e_{i,2} & \cdots & e_{i,x_i} & \cdots & e_{i,y_i} & \cdots & e_{i,|c_i|-1} & e_{i,|c_i|} \end{pmatrix}.$$

Cutting S along v is the same as cutting S along the l segments that make up v. If we cut c_i along the ith such segment, we obtain two new segments,

$$\begin{pmatrix} e_{i,1} & \cdots & e_{i,x_i} & b_1 & e_{i,y_i} & \cdots & e_{i,|c_i|} \end{pmatrix}$$
$$\begin{pmatrix} {e_{i,x_i}}^* & \cdots & {e_{i,y_i}}^* & b_2 \end{pmatrix}.$$

Note that the asterisks next to e_{i,x_i} and e_{i,y_i} in the second connected part indicate that, because we have split these edges in half, the edges with an asterisk have to gain a new value. Edges b_1 and b_2 are the sides of v, which also need new values. Regardless, the sum length of the two new connected parts is,

$$\big(x_i + |c_i| - y_i + 2\big) + \big(y_1 - x_i + 2\big) = |c_i| + 4.$$

Cutting S along the ith segment of v for all $1 \leq i \leq l$ obtains S', implying that:

$$n_{S'} = \sum_{c \in S'} |c| = \sum_{i=1}^{l} 4 + \sum_{c \in S} |c| = 4l + n_S.$$

□

Definition. For a combinatorial surface S, let G_S be the graph with $|S|$ vertices, each corresponding to a connected part of S, where vertices v_1, v_2 corresponding to connected parts c_1, c_2, respectively, share an edge if there exists $x \geq 0$ such that $x \in c_1$ and $x \in c_2$. That is, there is an edge between two vertices in G if an edge pairing in S joins their connected parts.

Lemma 3.2. *For a combinatorial surface S, G_S can be constructed in $\mathcal{O}(n_S)$ time.*

The following proof makes use of dictionaries. Recall that a *dictionary* or *associative array* is a function with a finite domain that maps *keys* to *values*. An unordered dictionary can be implemented using a hash table. Such an implementation means that it takes constant time to find a key's value or remove/insert a (key, value) pair. If D is a dictionary and i is a key of D, then $D(i)$ is i's value.

Proof. We first construct a dictionary D. The keys of D are the integers x such that there exists $c \in S$ such that $x \in c$. For a key x, D maps x to the following array of arrays: $\{(r, s) : S(r)(s) = x\}$. We can construct D by iterating through all n_S elements of S. For each element k at position $(r_k)(s_k)$ in S's two-dimensional array, we check if k is a key in D. If k is not a key in D, then we add it as a key in D that maps to the array $\{(r_k, s_k)\}$. If k is a key in D, then we append the array (r_k, s_k) to k's corresponding array. Each iteration takes constant time and there are n_S iterations, so the construction takes $\mathcal{O}(n_S)$ time.

We now construct G_S. For any connected part c with corresponding vertex v, we can find all of v's edges as follows: For all $x \geq 0$ such that $x \in c$, check the array that D maps x to. This array contains two arrays, (r_1, s_1) and (r_2, s_2). Let v_1 and v_2 be the vertices that correspond to connected parts r_1 and r_2, respectively. Add an edge in G_S between v and v_1 if $v \neq v_1$ and add an edge in G_S between v and v_2 if $v \neq v_2$. This constant time operation happens at most $|c|$ times. Finding all edges in the graph thus takes at most $\mathcal{O}(\sum_{c \in S} |c|) = \mathcal{O}(n_S)$ time. □

Lemma 3.3. *Let v be a curve on a combinatorial surface S. Then we can cut S along v into a set of connected combinatorial surfaces in $\mathcal{O}(n_S)$ time. Denote the result as* $\textsc{cut}(S, v)$.

Proof. The process detailed in Lemma 3.1 cuts S along v into S' in $\mathcal{O}(n_S)$ time. Our main task is to determine if v splits S into two connected surfaces. If so, we must split S' into these two distinct surfaces. To do this, construct G_S. Let c be one of the connected parts in S that has since been split into c_1 and c_2. We run a breadth-first search on G_S, starting at the vertex corresponding to c_1. If the search reaches c_2, then S is still connected and $\textsc{cut}(S, v) = \{S\}$. If not, we let S_1 be the surface with all the connected parts that the search reaches and let S_2 be the remaining connected parts. Then $\textsc{cut}(S, v) = \{S_1, S_2\}$. Constructing G_S and running the search algorithms takes $\mathcal{O}(n_S)$ time, so the claim holds. □

3.1. **The No Boundary Components Algorithm.** Let S be a combinatorial surface that has precisely one connected part c, has no boundary components, and can be decomposed into pairs of pants. This algorithm finds a non-contractible curve on S in $\mathcal{O}(n_S)$ time.

Let D be the dictionary whose keys $x \in c$ map x to $\begin{pmatrix} x_1 & x_2 \end{pmatrix}$ where x_1 and x_2 are the two integers $x_1 < x_2$ such that $c(x_1) = c(x_2) = x$. We construct D by iterating through all integers $0 \leq i < |c|$, adding i to $D(c(i))$. Let F be an empty set that will eventually store elements of c. We want to find an $x \in c$ such that the curve between $D(x)(0)$ and $D(x)(1)$ is non-contractible. The procedure NBC(c) in Algorithm 1 finds such an x. We now provide some commentary on this algorithm. Since the curve between $D(x)(0)$ and $D(x)(1)$ for some $x \in c$ passes through a connected part just once, if it is contractible, then it can be contracted by moving to the right of $D(x)(0)$ and the left of $D(x)(1)$, which we call contracting inwards, or to the right of $D(x)(1)$ and the left of $D(x)(0)$, which we call contracting outwards. We check for each of these possibilities in INWARDS and OUTWARDS in Algorithm 1. The set F contains $x \in c$ such that the curve between $D(x)(0)$ and $D(x)(1)$ is contractible. We use F to tabulate already-calculated results, which reduces the time complexity from $\mathcal{O}({n_S}^2)$ to $\mathcal{O}(n_S)$. We can use F in lines four and thirteen as, if a curve is homotopic to a curve which is already in F, then it is contractible as well, which reduces the necessary computation.

Proposition 3.4. *Let S be a surface that has precisely one connected part c, has no boundary components, and can be decomposed into multiple pairs of pants. Then the above* NO BOUNDARY COMPONENTS *algorithm finds a non-contractible curve on S in $\mathcal{O}(n_S)$ time.*

Proof. We need to show that some $x \in c$ satisfies INWARDS(x) and OUTWARDS(x), that the resulting chosen curve is non-contractible, and that the algorithm is linear with respect to n_S.

We first prove the second claim. For any $x \in c$, the straight curve between the two x edges, if contractible, can be either contracted by moving to the right of $D(x)(0)$ and the left of $D(x)(1)$ or to the right of $D(x)(1)$ and the left of $D(x)(0)$. Now INWARDS(x) and OUTWARDS(x) check for each of these, respectively. Since both return true, the curve is non-contractible.

Now suppose no such $x \in c$ satisfies both methods. Then, for each $x \in c$, the curve between the two x edges is contractible. Then every element of the fundamental group of S is the identity, the fundamental group is trivial, and S has genus $g = 0$. Yet S can be decomposed into multiple pairs of pants and has no boundary components, which means $g \geq 2$, which is the contradiction.

Lastly, we analyze the algorithm's time complexity. Constructing D takes $\mathcal{O}(|c|)$ time, and $|c| = n_S$. Then the only part of the algorithm that could take more than $\mathcal{O}(n_S)$ time is computing INWARDS(c, x) and OUTWARDS(c, x) for each $x \in c$. Now, for each $x \in c$, either INWARDS(c, x) and OUTWARDS(c, x) are both false or at least one is true. The former situation occurs only once, as the algorithm then terminates. Both INWARDS and OUTWARDS take at worst linear time with respect to n_S, so the former situation is $\mathcal{O}(n_S)$. In the latter case, every time the for loop in lines three or twelve is run, an additional element is added to F. Then the total cost of INWARDS(c, x) and OUTWARDS(c, x) over all $x \in c$ is bounded by the maximum size of F. But since $F \subseteq c$ and $|c| = n_S$, the total cost is $\mathcal{O}(n_S)$. Hence the algorithm takes $\mathcal{O}(n_S)$ time. □

Algorithm 1 Find a non-contractible curve on a surface S.

Require: $|S| = 1$

procedure INWARDS(c, x)
  $i, j \leftarrow D(x)(0),\ D(x)(1)$
  for $k \leftarrow 1$ to $\frac{j-i}{2}$ **do**
    if $c(i+k) \neq c(j-k)$ or $i+k = j-k$ or $c(i+k) \in F$ **then**
      return false, $\{c[l] \mid i \leq l < i+k\}$
    end if
  end for
  return true, $\{c[l] \mid i \leq l \leq j\}$
end procedure
procedure OUTWARDS(c, x)
  $i, j \leftarrow D(x)(0),\ D(x)(1)$
  for $k \leftarrow 1$ to $\frac{|c|+i-j}{2}$ **do**
    if $c(i-k) \neq c(j+k)$ or $i-k = j+k$ or $c(i-k) \in F$ **then**
      return false, $\{c[l] \mid j \leq l < j+k\}$
    end if
  end for
  return true, $\{c[l] \mid j \leq l \leq i+|c|\}$
end procedure
procedure NBC(c)
  for $x \in c$ **do**
    if $x \in F$ **then**
      continue
    end if
    if not INWARDS(c, x)(0) **and not** OUTWARDS(c, x)(0) **then**
      return x
    else
      $F \leftarrow F \cup$ INWARDS(c, x)(1) $\cup$ OUTWARDS(c, x)(1)
    end if
  end for
end procedure

3.2. **The One Boundary Component Algorithm.** Let S be a connected combinatorial surface that has precisely one distinguished loop, where each connected parted of S has precisely one boundary component edge, and that can be decomposed into multiple pairs of pants. This algorithm finds a non-contractible curve on S that is homotopically distinct and disjoint from the other distinguished loop on S in $\mathcal{O}(n_S)$ time.

As every connected part of S has precisely one boundary component edge and $b_S = 1$, then every $c \in S$ has precisely one boundary component edge, all of which have the

same value $b < 0$. Then,

$$S = \Bigg\{ \underbrace{\begin{pmatrix} x_{|S|} & b & x_1 & x_{1,1} & x_{1,2} & x_{1,3} & \dots \end{pmatrix}}_{c_1},$$
$$\underbrace{\begin{pmatrix} x_1 & b & x_2 & x_{2,1} & x_{2,2} & x_{2,3} & \dots \end{pmatrix}}_{c_2}, \dots$$
$$\underbrace{\begin{pmatrix} x_{|S|-1} & b & x_{|S|} & x_{|S|,1} & x_{|S|,2} & x_{|S|,3} & \dots \end{pmatrix}}_{c_{|S|}} \Bigg\}$$

where $x_i, x_{i,j} \geq 0$ for each such i.

Also, let $S^* = ((x \in c : x \geq 0) : c \in S)$. That is, S^* is a copy of S where all of the b boundary component edges have been removed from the connected parts. The advantage of creating S^* is that all curves that were homotopic to the b boundary component on S are now contractible in S^*, as we contracted the b boundary component to a point. Simultaneously, any valid curves we find on S^* are also valid curves on S. This means that we no longer need to ensure the new curve is homotopically-distinct or disjoint from the previous curve; we only need to find a non-contractible curve on S^*. So, our process is as follows: merge the connected parts of S^* into one connected part c^* and create a graph G that keeps track of how the connected parts are glued together, apply NO BOUNDARY COMPONENTS to find a non-contractible curve on c^*, and lastly use G to find the corresponding curve on S.

From the definition of S^*,

$$S^* = \Bigg\{ \begin{pmatrix} x_{|S|} & x_1 & x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,|c_1|-2} \end{pmatrix},$$
$$\begin{pmatrix} x_1 & x_2 & x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,|c_2|-2} \end{pmatrix}, \dots$$
$$\begin{pmatrix} x_{|S|-1} & x_{|S|} & x_{|S|,1} & x_{|S|,2} & x_{|S|,3} & \dots & x_{|S|,|c_{|S|}|-2} \end{pmatrix} \Bigg\}.$$

We next create G and c^*. Let G be an undirected graph with $|S|$ vertices $v_1, \dots, v_{|S|}$ corresponding to the connected parts $c_1, \dots, c_{|S|}$, respectively, such that v_i and v_{i+1} are connected by an edge with label x_i for each $1 \leq i < |S|$. Similarly, $v_{|S|}$ and v_1 are connected by an edge with label $x_{|S|}$. Additionally let,

$$c^* = \begin{pmatrix} x_{|S|,1} & x_{|S|,2} & \dots & x_{|S|,|c_{|S|}|-2} & \dots & x_{2,1} & x_{2,2} & \dots & x_{2,|c_2|-2} & \dots & x_{1,1} & x_{1,2} & \dots & x_{1,|c_1|-2} \end{pmatrix}.$$

Note that c^* is the connected part of S^* that is the result of merging all connected parts of S^* together as per the edges of G. Now apply NO BOUNDARY COMPONENTS to c^* to find a non-contractible curve. This curve connects two edges of c^*, which we call e_1 and e_2. We now run a breadth-first search on G to find the shortest path p between the connected part with e_1 and the connected part with e_2. Our curve is the path p, which both exists on S^* and on S.

Proposition 3.5. *Let S be a connected combinatorial surface that has precisely one boundary component, where each connected parted of S has precisely one boundary component edge, and that can be decomposed into multiple pairs of pants. Then the above* ONE BOUNDARY COMPONENT *algorithm finds a non-contractible curve on S that is homotopically-distinct and disjoint from the previous cut on S in $\mathcal{O}(n_S)$ time.*

Proof. We first show that the curve found by ONE BOUNDARY COMPONENT satisfies the desired properties. By Proposition 3.4, the curve found on c^* is non-contractible. Since c^* is topologically the same as S^*, the same applies to the curve on S^*. As S^* is obtained by contracting the boundary component b of S and the curve is non-contractible on S^*, the curve on S is homotopically distinct from b and non-contractible. Since the curve does not intersect vertex points of the connected parts, it does not intersect the point where b is contracted, meaning the curve is disjoint from b.

Now we show that the algorithm runs in $\mathcal{O}(n_S)$ time. Constructing S^* and G takes $\mathcal{O}(n_S)$ time, and c^* can be constructed from running breadth-first search on G, which takes $\mathcal{O}(n_S)$ time as well. The remaining steps consist of running NO BOUNDARY COMPONENTS and another breadth-first search on G, both of which also take $\mathcal{O}(n_S)$ time, so the total time complexity is $\mathcal{O}(n_S)$ as well. □

3.3. **The Shorten Curve Algorithm.** The goal of this algorithm is to modify previously made cuts on a surface S to make the cuts shorter. We think about this as, right after we make a cut as per a different algorithm, we apply SHORTEN CURVE to "change our minds" about the cut we made. Note that this algorithm does not change the geometry of the surface—it only changes where the curve lies on the surface.

First construct dictionary D as per Lemma 3.2. Make a queue q of all $c \in S$ such that $|c| = 2$ and $\min(c) < 0$. We then repeat the following process:

If q is empty, end the algorithm. If not, take some c from the start of the queue. Then $c = \begin{pmatrix} x & b \end{pmatrix}$ for some $x \geq 0, b < 0$. Remove c from S. (Note that because the order of S does not matter, we can swap two elements of S to remove c in constant time.) Now use D to find $\tilde{c} \in S$ such that $x \in \tilde{c}$. Then replace the sub-array $\begin{pmatrix} b & x & b \end{pmatrix}$ in $\tilde{c}$ with $\begin{pmatrix} b \end{pmatrix}$. Lastly, remove the x key from D to update D in constant time and, if $|\tilde{c}| = 2$ and $\min(\tilde{c}) < 0$, then add $\tilde{c}$ to q. We then repeat the above steps.

Remark. The SHORTEN CURVE algorithm does not always run in $\mathcal{O}(n_S)$ time. However, the algorithm does run in $\mathcal{O}(n_S)$ time when we apply it after the MULTIPLE BOUNDARY COMPONENTS algorithm, which is detailed below, assuming there existed no c in S such that $|c| = 2$ and $\min(c) < 0$ before the MULTIPLE COMPONENTS algorithm was run. This is all we need due to the ultimate construction of Algorithm 2, whereby we only apply SHORTEN CURVE after MULTIPLE BOUNDARY COMPONENTS and MULTIPLE BOUNDARY COMPONENTS is the only algorithm that can add such a c to S.

We now prove this runtime result. While we have not yet introduced MULTIPLE BOUNDARY COMPONENTS, the only fact we need is that MULTIPLE BOUNDARY COMPONENTS(S) has at most two connected parts c_1 and c_2 such that $|c_1|, |c_2| = 2$ and $\min(c_1), \min(c_2) < 0$. Every element from q represents a connected part that ultimately gets removed from S, meaning at most $|S|$ elements are ever added to q. For each element in q, the only action that takes more than constant time is replacing the sub-array $\begin{pmatrix} b & x & b \end{pmatrix}$ in $\tilde{c}$ with $\begin{pmatrix} b \end{pmatrix}$. Then all other actions take $\mathcal{O}(|S|) \leq \mathcal{O}(n_S)$ time. Next note that if $|\tilde{c}| > 4$, then the modification of $\tilde{c}$ does not add another element to q and the modification takes $\mathcal{O}(n_S)$ time. If instead $|\tilde{c}| \leq 4$, then the modification of $\tilde{c}$ adds at most one element to q and takes constant time. As q initially starts with the two elements c_1 and c_2, the cumulative time taken by each of the descendants of c_1 and c_2 is at most a constant times the number of descendants of the connected part,

which is at most $|S| \leq n_S$, plus the time of the modification of the final descendant of the connected part, which is $\mathcal{O}(n_S)$. Hence the total runtime is $\mathcal{O}(n_S)$.

3.4. **The Multiple Boundary Components Algorithm.** Let S be a connected combinatorial surface with at least two distinguished loops. This algorithm finds a non-contractible curve on S that is homotopically distinct and disjoint from all the other distinguished loops on S.

We first construct a graph G by adding b_S more vertices to G_S: For all $x < 0$ where there exists $c \in S$ such that $x \in c$, we create a vertex v_x that shares an edge with all other vertices whose connected part contains x in its edges. We now run a breadth-first search algorithm b_S different times. Specifically, for all $x < 0$ where there exists $c \in S$ such that $x \in c$, we start a search from v_x and end the algorithm once we arrive at a vertex v_y with $y < x$. We keep track of the shortest path found from the breadth-first search algorithms. Call this path p and let the vertices that p connects be v_x and v_y. If we refer to the connected parts that vertices in G correspond to, then p is also a path on S between a connected part c_1 that has boundary component x as an edge and a connected part c_2 that has boundary component y as an edge. We can now construct the new non-contractible curve on S.

Let this curve start in c_1, right next to the x edge. Then have the curve follow the x boundary component across S, staying right next to the boundary component, until it has traveled the entire path of the x boundary component. At this point, have the curve follow p until it reaches c_2. Then have it similarly follow the y boundary component across S until it returns to c_2. Lastly, have the curve follow p back to its starting position in c_1. This is the desired curve.

Proposition 3.6. *Let S be a connected combinatorial surface with at least two distinguished loops. Then the above* MULTIPLE BOUNDARY COMPONENTS *algorithm finds a non-contractible curve on S that is homotopically distinct and disjoint from all the distinguished loops on S in $\mathcal{O}(n_S b_S)$ time.*

Proof. We first show the obtained curve satisfies the desired properties. Its construction ensures it only passes through edges of connected parts that are identified with other edges, so the curve is disjoint from all other distinguished loops on S. Next, the surface enclosed by the curve has genus zero and three boundary components (boundary component x, boundary component y, and the curve), so it is a pair of pants. Thus the curve is non-contractible. This also implies the curve is homotopically-distinct from previous cuts on S as it changes the topology of the remaining part of S.

We next look at time complexity. Constructing G requires constructing G_S, which takes linear time in n_S by Lemma 3.2. It also requires adding the v_x vertices, which can be added in the same way as the other vertices in G_S: by using D, which also takes at most $\mathcal{O}(n_S)$ time. Now G is a graph with $|S| + b_S$ vertices and at most n_S edges. A breadth-first search runs in $\mathcal{O}(V + E)$ time on a graph with V vertices and E edges. Hence the b_S breadth-first searches run in

$$\mathcal{O}(b_S(|S| + b_S + n_S)) \leq \mathcal{O}(b_S(n_S + n_S + n_S)) = \mathcal{O}(n_S b_S)$$

time. □

3.5. **The Linear Decomposition Algorithm.** Putting all of the results in this section together, we have our final algorithm for finding pants decompositions. The LINEAR

Algorithm 2 The First Pants Decomposition Algorithm

procedure LINEAR DECOMPOSITION(S)
  if genus(S) $= 0$ and $b_S = 3$ **then** ▷ If S is a pair of pants.
    return 0
  else if $|S| = 1$ **then** ▷ If S has one connected part.
    $v \leftarrow$ NO BOUNDARY COMPONENTS(S)
    $S_{\text{new}} \leftarrow$ CUT(S, v)
  else if $b_S = 1$ **then** ▷ If S has one boundary component.
    $v \leftarrow$ ONE BOUNDARY COMPONENT(S)
    $S_{\text{new}} \leftarrow$ CUT(S, v)
  else ▷ If S has multiple boundary components.
    $v \leftarrow$ MULTIPLE BOUNDARY COMPONENTS(S)
    $S_{\text{new}} \leftarrow$ CUT(S, v)
    $S_{\text{new}} \leftarrow \{$SHORTEN CURVE($S'$) $\mid S' \in S_{\text{new}}\}$
  end if
  $l \leftarrow \sum_{S' \in S_{\text{new}}} L_{S'}(b)$ ▷ Here, b is the label of one side of v.
  for $S' \in S_{\text{new}}$ **do** ▷ S_{new} is a set of ≤ 2 surfaces achieved from the cut.
    $l \leftarrow \max(l,$ LINEAR DECOMPOSITION(S')) ▷ Decompose resulting surface(s).
  end for
  return l
end procedure

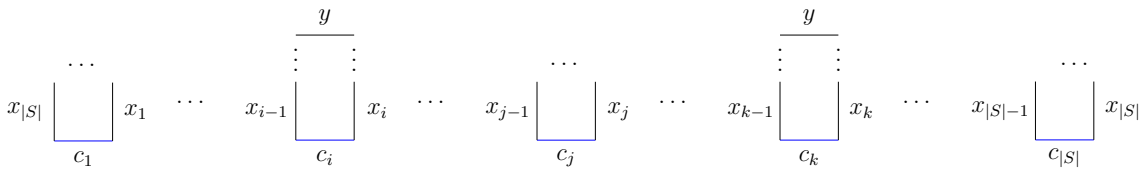
DECOMPOSITION algorithm is described in Algorithm 2. The algorithm recursively applies itself to the new surfaces created by cutting along these curves until the surface has been decomposed into pairs of pants. The fact that LINEAR DECOMPOSITION actually produces a pants decomposition is a consequence of the above propositions, which show that each step finds a new non-contractible, homotopically distinct, and disjoint curve on the surface. The one complication is that ONE BOUNDARY COMPOENNT requires each connected part to have precisely one boundary component edge. It turns out that this requirement is always met, and the result is a consequence of the proofs of Lemma 3.7 and Theorem 3.8.

Due to Lemma 2.3 and the fact that we use $L_S(b)$ in the algorithm, the pants decomposition that LINEAR DECOMPOSITION finds has a length of at most $2 \cdot$ LINEAR DECOMPOSITION $+ \varepsilon$, for any $\varepsilon > 0$. The same holds true for the algorithm we propose in Section 4. Note that the following analysis concerns LINEAR DECOMPOSITION. To bound the exact length of the decomposition, multiply all the bounds by two. We now analyze the time complexity and output of LINEAR DECOMPOSITION. It turns out that LINEAR DECOMPOSITION has a linear upper bound with respect to the genus of the surface, which is the reason the algorithm is named as such.

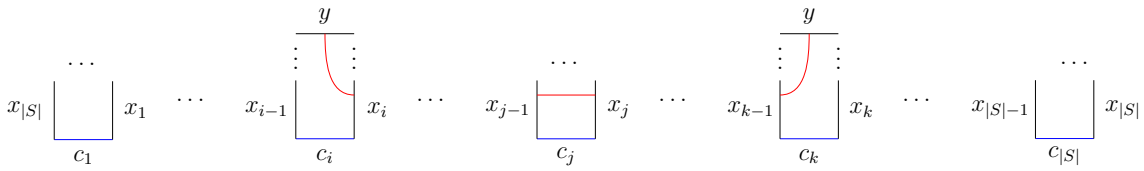
Lemma 3.7. *If S is a combinatorial surface with genus $g \geq 1$, $b_S = 1$, and each $c \in S$ has one boundary component edge, then,*

$$\text{LINEAR DECOMPOSITION}(S) \leq |S| + 4(g-1).$$

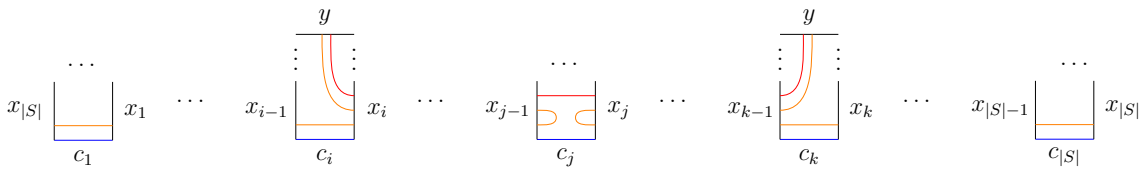
Proof. We induct on g. When $g = 1$, LINEAR DECOMPOSITION$(S) \leq \lceil \frac{|S|}{2} \rceil \leq |S| + 4(g-1)$ as per the ONE BOUNDARY COMPONENT algorithm. Now assume the result holds for such surfaces of genus g and consider some such surface S of genus $g+1$.



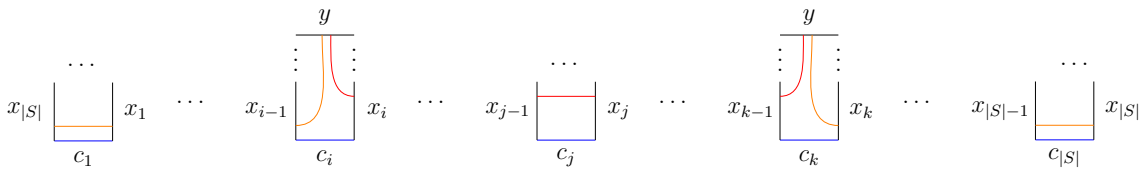
(A) The surface S with $b_S = 1$.



(B) S after one cut.



(C) S after two cuts but before applying SHORTEN CURVE.



(D) S after two cuts and applying SHORTEN CURVE.

FIGURE 2. A surface S with $b_S = 1$ after no cuts, one cut, two cuts before applying SHORTEN CURVE, and two cuts after applying SHORTEN CURVE.

As every connected part has precisely one boundary component edge and $b_S = 1$, then every $c \in S$ has precisely one boundary component edge, all of which have the same value $b < 0$. Then,

$$S = \left\{ c_1 = \begin{pmatrix} x_{|S|} & b & x_1 & \dots \end{pmatrix}, c_2 = \begin{pmatrix} x_1 & b & x_2 & \dots \end{pmatrix}, \dots, c_{|S|} = \begin{pmatrix} x_{|S|-1} & b & x_{|S|} & \dots \end{pmatrix} \right\}$$

where $x_i \geq 0$ for each such i. This view is shown in Figure 2a. We now analyze the first few cuts in LINEAR DECOMPOSITION(S). The algorithm first uses ONE BOUNDARY COMPONENT, which, since every connected part has exactly one b, finds a curve that travels alongside b. Specifically, the resulting curve starts from connected part c_i then travels along $x_i, x_{i+1}, \dots x_{j-1}$ to connected part c_j for some $i < j$, as shown in Figure 2b. The second cut then uses MULTIPLE BOUNDARY COMPONENTS followed by SHORTEN CURVE. The cut is shown both before and after using SHORTEN CURVE in Figure 2c and Figure 2d, respectively. This cut decomposes S into a pair of pants and a different surface. Note that this other surface might either be two unconnected surfaces S_1 and S_2 or be one connected surface.

Suppose the former case. Then $b_{S_1} = 1$ and $b_{S_2} = 1$, and the surfaces' genuses g_1 and g_2, respectively, are both less than or equal to g. Lastly, every connected part in each surface has precisely one boundary component edge. As such, we can apply the inductive hypothesis. Applying MULTIPLE BOUNDARY COMPONENTS increases the number of connected parts by two, so the total number of said parts is at most $|S| + 2$, meaning each of the first two curves has length at most $|S| + 2$. Lastly, $|S_1|, |S_2| \leq |S|$. So,

$$\begin{aligned}\text{LINEAR DECOMPOSITION}(S) &\leq \max\{|S_1| + 4(g_1 - 1), |S_2| + 4(g_2 - 1), |S| + 2\} \\ &\leq |S| + \max\{4(g-1), 4(g-1), 2\} \\ &\leq |S| + 4(g-1).\end{aligned}$$

Now consider the latter case, where the second cut decomposes S into a pair of pants and one connected surface with two boundary components. The LINEAR DECOMPOSITION algorithm now uses MULTIPLE BOUNDARY COMPONENTS again two decompose the connected surface into a pair of pants and a connected surface S' with $b_{S'} = 1$. Once again, $|S'| \leq |S| + 4$, S' has genus $g - 1$, and every connected part in the surface has precisely one boundary component edge. Moreover, then the maximum length of those three curves must be at most $|S| + 4$. By induction,

$$\begin{aligned}\text{LINEAR DECOMPOSITION}(S) &= \max\{\text{LINEAR DECOMPOSITION}(S'), |S| + 4\} \\ &\leq \max\{(|S| + 4) + 4(g-2), |S| + 4\} \\ &\leq |S| + 4\max\{g - 1, 1\} \\ &\leq |S| + 4(g-1).\end{aligned}$$

□

Theorem 3.8. *Suppose Algorithm 2 is run on a genus $g \geq 2$ connected combinatorial surface S with no boundary components and one connected part. Then,*

$$\text{LINEAR DECOMPOSITION}(S) \leq 4g - 4.$$

Proof. The first curve from LINEAR DECOMPOSITION(S) either decomposes S into two unconnected combinatorial surfaces S_1 and S_2 or into one combinatorial surface with two connected parts, each containing one boundary component. In the former situation, note $b_{S_1} = 1$ and $b_{S_2} = 1$. Moreover, $|S_1| = |S_2| = 1$, every connected part in each surface has precisely one boundary component edge, the surfaces' genuses g_1 and g_2, respectively, are both less than g. Applying Lemma 3.7 yields,

$$\begin{aligned}\text{LINEAR DECOMPOSITION}(S) &\leq \max\{|S_1| + 4(g_1 - 1), |S_2| + 4(g_2 - 1)\} \\ &\leq \max\{1 + 4(g-2), 1 + 4(g-2)\} \\ &\leq 1 + 4(g-2) \\ &\leq 4g - 4.\end{aligned}$$

Now consider the latter situation, where the first curve from LINEAR DECOMPOSITION(S) decomposes S into one combinatorial surface with two connected parts. Then the second curve applies the MULTIPLE BOUNDARY COMPONENTS algorithm, which cuts S into a pair of pants and a surface S' with four connected parts and genus $g - 1$. Moreover, the maximum length of these first two cuts is four and $b_{S'} = 1$. Lastly, every

connected part in the surface has precisely one boundary component edge. Then we can apply Lemma 3.7 to achieve,

$$\begin{aligned}\textsc{linear decomposition}(S) &\leq \max\{\textsc{linear decomposition}(S'), 4\}\\ &\leq \max\{4 + 4(g-2), 4\}.\end{aligned}$$

Because S can be decomposed into pairs of pants, $g \geq 2$. Hence,

$$\textsc{linear decomposition}(S) \leq 4g - 4.$$

□

Lemma 3.9. *Each recursive call of* LINEAR DECOMPOSITION(S') *takes* $\mathcal{O}(n_{S'})$ *time.*

Proof. As shown by previous lemmas and propositions, every line in Algorithm 2 takes $\mathcal{O}(n_{S'})$ time aside from line ten. As per Proposition 3.6, line ten instead takes

$$\mathcal{O}\left(n_{S'} b_S\right) \text{ time.}$$

However, note that b_S starts at $b_S = 0$ for the first surface inputted into LINEAR DECOMPOSITION(S). Next, b_S only increases by at most two after each cut, since each boundary component has two unique sides. Lastly, the cut in line ten occurs whenever $b_S \geq 2$ and results in b_S decreasing by one. Then if LINEAR DECOMPOSITION(S') is called recursively, we know that $b_S \leq 3$. Hence,

$$\begin{aligned}\mathcal{O}\left(n_{S'} b_S\right) &\leq \mathcal{O}\left(3 n_{S'}\right)\\ &= \mathcal{O}\left(n_{S'}\right).\end{aligned}$$

□

Theorem 3.10. *Suppose Algorithm 2 is run on a genus $g \geq 2$ connected combinatorial surface S with no boundary components and one connected part. Then the algorithm takes $\mathcal{O}(n_S g + g^3)$ time.*

Proof. Though the algorithm is recursive, every call of the LINEAR DECOMPOSITION function either makes a new cut on the inputted surface or declares that the inputted surface is a pair of pants. As stated in Section 1, a pants decomposition consists of $3g - 3$ curves that decompose a surface into $2g - 2$ pairs of pants. Hence the LINEAR DECOMPOSITION function is called at most $5g - 5$ times.

Lemma 3.1 tells us that with each cut that passes through connected parts of S' a total of l times, $n_{S'}$ increases by at most $4l$. Notice that the proofs of Lemma 3.7 and Theorem 3.8 don't just bound LINEAR DECOMPOSITION(S), but also the maximum number of times a curve in the decomposition passes through connected parts of the surface. As such, Theorem 3.8 then gives that the largest possible value of $n_{S'}$ over all surfaces S' that we input into LINEAR DECOMPOSITION is $n_S + 4(4g-4)(3g-3)$. Lemma 3.9 implies the time complexity of each recursive call of LINEAR DECOMPOSITION(S') is $\mathcal{O}(n_{S'})$, meaning each recursive call takes at most $\mathcal{O}(n_S + 4(4g-4)(3g-3))$ time. With $5g - 5$ calls of LINEAR DECOMPOSITION(S'), the algorithm's worst-case total time complexity is,

$$\begin{aligned}\mathcal{O}\Big(\big(5g-5\big)\big(n_S + 4(4g-4)(3g-3)\big)\Big) &= \mathcal{O}\big(g(n_S + g^2)\big)\\ &= \mathcal{O}(n_S g + g^3).\end{aligned}$$

□

Corollary 3.11. *Let S be a random combinatorial surface created by identifying all the edges of a $2n$-gon in pairs uniformly at random. Suppose S has genus $g \geq 2$. Then the expected value of the time complexity of Algorithm 2 is $\mathcal{O}(g^3)$.*

Proof. Apply Theorem 6.1 to Theorem 3.10. □

4. The Second Pants Decomposition Algorithm

In this section, we present another algorithm that finds a pants decomposition for a genus $g \geq 2$ combinatorial surface. The goal of this second algorithm is to produce a pants decomposition with shorter average length than the decomposition produced by Algorithm 2. Though we do not achieve an upper bound on the length of the boundary components, the upper bound appears to be linear with respect to the genus and significantly smaller than the bound found in Section 3.

4.1. **The Genus Removal Algorithm.** Given a connected combinatorial surface S, this algorithms finds a non-contractible curve on S that is homotopically distinct and disjoint from all previous curves. This algorithm focuses on finding a curve that reduces the genus of the surface, as opposed to decreasing the number of boundary components.

Begin by constructing the dictionary D in the same manner as in the proof of Lemma 3.2. Let H be a graph with one vertex for each connected part of S and no initial edges. Pick a connected part $c_1 \in S$ with maximal $|\{x \geq 0 : x \in c_1\}|$ and conduct a breadth-first search algorithm of G_S that starts from c_1 and is modified for cycle detection. As we conduct the search, we connect vertices in H together as per the search so that H is an acyclic graph; we also label each edge with the shared edge that glues together the vertices' corresponding connected parts.

Specifically, when the vertex for some connected part $\tilde{c}$ is taken from the queue, we look at all $x \in \tilde{c}$ such that $x \geq 0$ and x is not the edge connecting $\tilde{c}$ to its parent connected part, information we get from H. We then use D to find the connected part c_x that each x joins to $\tilde{c}$. If c_x has not been visited, we add it to the queue and add an edge with label x in H between the vertices corresponding to $\tilde{c}$ and c_n. If not, we consider the unique curve between the two x edges in S whose path is given by the path in H between the two connected parts. If this curve cuts S into two disconnected combinatorial surfaces, of which one has genus zero and less than three boundary components, then we continue with the breadth-first search. Otherwise, we return said cut and end the algorithm.

Proposition 4.1. *Suppose S is a connected combinatorial surface with non-zero genus. The* genus removal *algorithm described above finds a non-contractible curve that is homotopically-distinct and disjoint from all previous cuts on S. In addition, it does so in $\mathcal{O}({n_S}^2)$ time.*

Proof. Note that the test at the end of the algorithm guarantees that, if a curve is found, then the curve is non-contractible and homotopically-distinct and disjoint from all previous cuts on S. We next show a suitable curve must be found. The glueing of two connected parts,

$$c_1 = \begin{pmatrix} e_1 & e_2 & \cdots & e_n & \cdots & e_{|c_1|-1} & e_{|c_1|} \end{pmatrix} \quad \text{and} \quad c_2 = \begin{pmatrix} f_1 & f_2 & \cdots & f_m & \cdots & f_{|c_2|-1} & f_{|c_2|} \end{pmatrix}$$

around some shared edge $e_n = f_m = x \geq 0$ is the new connected part,

$$\begin{pmatrix} e_1 & \cdots & e_{n-1} & f_{m+1} & \cdots & f_{|c_2|} & f_1 & \cdots & f_{m-1} & e_{n+1} & \cdots & e_{|c_1|} \end{pmatrix}.$$

Note that the glueing of c_1 and c_2 contains the same topological information as $\{c_1, c_2\}$. Now consider the connected part c formed by successively glueing pairs of connected parts in S around shared edges x if they are connected in H by an edge with label x. The result is a connected part that is homeomorphic to the entirety of S. Since c then has non-zero genus, its fundamental group is non-trivial. Then there exists some edge pairing, between edges labeled y, in c such that the direct path between the two edges forms a non-contractible curve that is homotopically distinct from all other curves in c. The breadth-first search of G_S conducted in the GENUS REMOVAL algorithm searches through all edges, so it will eventually look at each one of the two y edges. When it does, the algorithm considers the curve between the two y edges in S whose path is given by H. But this is the same as the curve between the y edges in c. Then the curve is non-contractible and homotopically distinct from all previous curves. Thus the algorithm finds a suitable curve.

We next look at the algorithm's time complexity. In Lemma 3.2, we showed that constructing D and G_S takes $\mathcal{O}(n_S)$ time. The breadth-first search consists of at most $\mathcal{O}(n_S)$ passes. At each edge in G_S, we either add an edge to H, which takes constant time, or cut along the potential curve and check the genus of the resulting surface(s), which takes $\mathcal{O}(n_S)$ time as per Lemma 3.3 and Lemma 2.4. Then the GENUS REMOVAL algorithm runs in $\mathcal{O}({n_S}^2)$ time. □

Definition. A length $n \geq 1$ chain is a set of connected parts $\{c_0, c_1, c_2, \ldots, c_n, c_{n+1}\}$ in S (where c_0 is not necessarily distinct from c_{n+1}) such that, for each $1 \leq i \leq n$, $|c_i| = 4$, c_i contains precisely two boundary component edges, and c_i neighbors both c_{i-1} and c_{i+1} in G_S. Moreover, $|c_0|, |c_{n+1}| > 4$.

Equivalently, a length $n \geq 1$ chain in a combinatorial surface S is a set of n squares, whose edges are identified to be all in one a row and who all have two boundary component edges. In addition, the connected parts at the end of the chain are both identified to connected parts that are not squares.

4.2. **The Chain Detection Algorithm.** Let S be a connected combinatorial surface that can be decomposed into pairs of pants. This algorithm aims to determine if S has at least one chain. For each $c \in S$, determine whether $|c| = 4$ and c contains precisely two boundary components. We return `true` if some $c \in S$ satisfies both these criteria and false otherwise.

Proposition 4.2. *Suppose S is a combinatorial surface that can be decomposed into pairs of pants. Then the* CHAIN DETECTION *algorithm described above correctly determines in $\mathcal{O}(n_S)$ time whether a combinatorial surface S has a chain.*

Proof. We need to prove S has a chain if and only if there exists $c \in S$ such that $|c| = 4$ and c contains precisely two boundary components.

To prove the forward direction, note that if S has a length $n \geq 1$ chain $\{c_0, c_1, c_2, \ldots, c_n, c_{n+1}\}$, then $|c_1| = 4$ and c_1 contains precisely two boundary components. This is the desired connected part in S.

To prove the converse, suppose there exists $c \in S$ such that $|c| = 4$ and c contains precisely two boundary components, which we call b_1 and b_2. Since b_1 and b_2 cannot be adjacent, we can write $c = \begin{pmatrix} x_1 & b_1 & x_0 & b_2 \end{pmatrix}$ for some $x_0, x_1 \geq 0$. Now consider the connected parts that connect to c through x_1 and x_0. If one of them has length four and precisely two boundary components, then iteratively consider the next connected part

that it neighbors. As $|S|$ is finite, if this process never ends, then the connected parts in question must form a loop. This means S has at most two boundary components and genus zero. But then S cannot be decomposed into pairs of pants. Then the process must terminate after examining $m \geq 3$ connected parts. Let c_0 and c_{m-1} be the two connected parts examined that do not satisfy the two criteria, and label each connected part in between the two from c_1 to c_{m-2}. Then $\{c_0, c_1, c_2, \ldots, c_{m-2}, c_{m-1}\}$ is a length $m-1$ chain in S. Hence S has a chain.

We lastly analyze the algorithm's time complexity. Examining the size of each $c \in S$ and the number of boundary components in each such c takes $\mathcal{O}(|c|)$ operations. So CHAIN DETECTION runs in $\mathcal{O}(\sum_{c \in S} |c|) = \mathcal{O}(n_S)$ time. □

4.3. **The Chain Removal Algorithm.** Suppose S is a connected combinatorial surface that can be decomposed into pairs of pants, and S contains a length n chain. This algorithm aims to cut S along one or two non-contractible curves that are homotopically-distinct and disjoint from all previous cuts on S and each other such that the resulting surface has at least n fewer connected parts than the original surface. The algorithm returns the resulting surface and the maximum length of the curves.

Use the CHAIN DETECTION algorithm to identify some $c = \begin{pmatrix} x_1 & b_1 & x_0 & b_2 \end{pmatrix} \in S$ with $x_0, x_1 \geq 0$ and $b_1, b_2 < 0$. Consider two situations, depending on whether $b_1 = b_2$.

If $b_1 \neq b_2$, we apply a modification of the MULTIPLE BOUNDARY COMPONENTS algorithm (instead of searching through G for p, we let p be the path on G from b_1 to c to b_2, which means the algorithm takes $\mathcal{O}(n_S)$ time), which adds a new boundary component $\tilde{b}$ to the surface. We lastly run the SHORTEN CURVE algorithm to obtain the final surface, which we return alongside $L_S(\tilde{b})$.

Now suppose $b_1 = b_2$ and let $b < 0$ be their common value. We remove c from S and add the two connected parts $\begin{pmatrix} x_1 & b \end{pmatrix}$ and $\begin{pmatrix} x_0 & b \end{pmatrix}$ to S. If the b boundary component is connected, then S must be non-orientable, a contradiction. So the b boundary component is two disjoint boundary components. Pick integers $b_3, b_4 < 0$ that are not previous boundary components on S. Running a breadth-first search along the path of b, we replace all b's with b_3 or b_4 such that b_3 and b_4 both form two distinct loops. We then run the SHORTEN CURVE algorithm to obtain the final curves. The end result is a surface with two additional cuts b_3 and b_4 and one fewer chain. We return the resulting surface and $\max\{L_S(b_3), L_S(b_4)\}$.

Proposition 4.3. *Suppose S is a connected combinatorial surface that can be decomposed into pairs of pants, and S contains a length n chain. Then the* CHAIN REMOVAL *algorithm described above cuts S along one or two non-contractible curves that are homotopically-distinct and disjoint from all previous cuts on S and each other such that the resulting surface has at least n fewer connected parts than the original surface. In addition, it does so in $\mathcal{O}(n_S)$ time.*

Proof. The CHAIN REMOVAL algorithm first uses CHAIN DETECTION to identify some connected part $c \in S$ that is part of some chain $\{c_0, c_1, c_2, \ldots, c_n, c_{n+1}\}$. Note $c = c_k$ for some $1 \leq k \leq n$. Since the two boundary components of c_1 cannot be adjacent, we can write $c_1 = (x_1, b_1, x_0, b_2)$, where $x_1, x_0 \geq 0$, $x_0 \in c_0$, $x_1 \in c_2$, and $b_1, b_2 < 0$. If $n \geq 2$, then $x_1 \in c_2 \implies c_2 = (x_2, b_1, x_1, b_2)$ for some $x_2 \geq 0$. If $n \geq 3$, then since x_1 connects c_2 to c_1 and c_2 connects to c_3, $x_2 \in c_3$ and so $c_3 = (x_3, b_1, x_2, b_2)$ for some $x_3 \geq 0$.

Continue the process inductively to achieve, for $1 \leq i \leq n$, $c_i = (x_i, b_1, x_{i-1}, b_2)$, where each $x_i \geq 0$ and $b_1, b_2 < 0$.

If $b_1 \neq b_2$, then CHAIN REMOVAL applies MULTIPLE BOUNDARY COMPONENTS, which we already proved in Proposition 3.6 cuts S along a non-contractible curve that is homotopically-distinct and disjoint from all previous cuts on S. In particular, it splits c_k into $\begin{pmatrix} x_k & b^* \end{pmatrix}$ and $\begin{pmatrix} x_{k-1} & b^* \end{pmatrix}$ for some new boundary component $b^* < 0$ and replaces all b_1, b_2 in S with b^*. Using our labeling of the connected parts $\{c_1, c_2, \ldots, c_n\}$, it is clear that the SHORTEN CURVE algorithm removes $c_1, c_2, \ldots, c_n$ from S and modifies c_0 and c_{n+1}. Then the resulting surface has at least n fewer connected parts than the original surface.

If $b_1 = b_2 = b$, then CHAIN REMOVAL splits c_k into $\begin{pmatrix} x_k & b \end{pmatrix}$ and $\begin{pmatrix} x_{k-1} & b \end{pmatrix}$. After each b is replaced with either b_3 or b_4 such that b_3 and b_4 form disjoint loops, the discarded region is a pair of pants as it has genus zero and its boundary components are b, b_3, and b_4. Then both b_3 and b_4 are non-contractible curves that are homotopically-distinct and disjoint from all previous cuts on S and each other—if they were not, then the cuts would result in a cylinder or circle. The connected parts in the chain out of one side of c_k are the same as in the $b_1 \neq b_2$ case, just with the b labels replaced with a new label. The same applies to the other side of the chain, again with a different label. Then the same reasoning as in the $b_1 \neq b_2$ case gives that the resulting surfaces has at least n fewer connected parts than the original surface.

We now analyze the algorithm's time complexity. We first call CHAIN DETECTION. By Proposition 4.2, this takes $\mathcal{O}(n_S)$ time. Changing c_k takes constant time, and SHORTEN CURVE takes $\mathcal{O}(n_S)$ time. If $b_1 = b_2$, running the breadth-first search along the path of b also takes $\mathcal{O}(n_S)$ time as every vertex and edge in the graph corresponds to an edge in a $c \in S$. Hence CHAIN REMOVAL takes $\mathcal{O}(n_S)$ time. □

4.4. **The Genus Decomposition Algorithm.** Algorithm 3, the GENUS DECOMPOSITION algorithm, is our second pants decomposition algorithm and the culmination of the algorithms presented in this section. Its key feature, and the reason for its name, is that it prioritizes cuts early on that decrease the genus of the surface. Just like the LINEAR DECOMPOSITION algorithm, the GENUS DECOMPOSITION algorithm uses recursion, with each recursive step cutting the surface along one or two curves before applying GENUS DECOMPOSITION to finish the decomposition on each of the resulting connected surfaces. The curves produced by GENUS DECOMPOSITION indeed result in a pants decomposition. This is due to the preceding propositions, which prove that the curves found in each step are non-contractible and homotopically-distinct and disjoint from each other. Lastly, just as for the LINEAR DECOMPOSITION algorithm, note that GENUS DECOMPOSITION finds a pants decomposition with a length of at most $2 \cdot \text{GENUS DECOMPOSITION} + \varepsilon$, for any $\varepsilon > 0$, due to Lemma 2.3 and the use of $L_S(b)$ in the algorithm. We now analyze the time complexity of GENUS DECOMPOSITION.

Theorem 4.4. *Suppose Algorithm 3 is run on a genus $g \geq 2$ connected combinatorial surface S, and let L be the maximum number of identified edges that a curve in the resulting pants decomposition intersects. Then the algorithm takes $\mathcal{O}(g{n_S}^2 + g^3L^2)$ time.*

Proof. Consider each recursive call GENUS DECOMPOSITION(S') within the overall algorithm. Since the total number of curves in the decomposition is $3g - 3$, $b_{S'} = \mathcal{O}(g)$.

Algorithm 3 The Second Pants Decomposition Algorithm

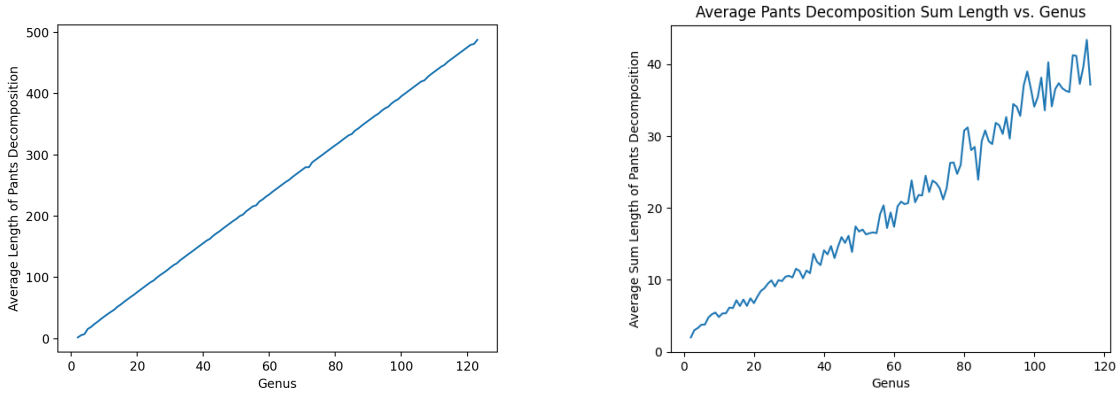
procedure GENUS DECOMPOSITION(S)
  if genus(S) $= 0$ and $b_S = 3$ **then** ▷ If S is a pair of pants.
    return 0
  else if CHAIN DETECTION(S) **then** ▷ If S has at least one chain.
    $S_{\text{new}}, l \leftarrow$ CHAIN REMOVAL(S)
  else if genus(S) $= 0$ **then** ▷ If S has genus 0 but is not a pair of pants.
    $v \leftarrow$ MULTIPLE BOUNDARY COMPONENTS(S)
    $S_{\text{new}} \leftarrow$ CUT(S, v)
    $S_{\text{new}} \leftarrow \{$SHORTEN CURVE($S'$) $\mid S' \in S_{\text{new}}\}$
    $l \leftarrow \sum_{S' \in S_{\text{new}}} L_{S'}(b)$ ▷ Here, b is the label of one side of v.
  else ▷ If S has non-zero genus and no chains.
    $v \leftarrow$ GENUS REMOVAL(S)
    $S_{\text{new}} \leftarrow$ CUT(S, v)
    $l \leftarrow \sum_{S' \in S_{\text{new}}} L_{S'}(b)$ ▷ Again, b is the label of one side of v.
  end if
  for $S' \in S_{\text{new}}$ **do** ▷ S_{new} is a set of ≤ 2 surfaces achieved from the cut.
    $l \leftarrow \max(l,$ GENUS DECOMPOSITION(S')) ▷ Decompose resulting surface(s).
  end for
  return l
end procedure

Now CHAIN DETECTION and CHAIN REMOVAL both take $\mathcal{O}(n_S)$ time, due to Proposition 4.2 and Proposition 4.3, respectively. On the other hand, MULTIPLE BOUNDARY COMPONENTS and GENUS REMOVAL take $\mathcal{O}({n_S}^2)$ time, as per Proposition 3.6 and Proposition 4.1, respectively. The remaining steps in the algorithm take $\mathcal{O}(n_S)$ time, due to additional prior results. Then each recursive call takes $\mathcal{O}({n_S}^2)$ time. Due to Lemma 3.1, $n_{S'}$ increases by at most $4L$ with each cut. With $\mathcal{O}(g)$ curves, we have $n_{S'} = \mathcal{O}(n_S + gL)$. There are $\mathcal{O}(g)$ recursive calls, so the entire algorithm takes $\mathcal{O}(g(n_S + gL)^2)$ time. Since either $n_S \geq gL$ or $n_S < gL$, Algorithm 3 has time complexity $\mathcal{O}(g{n_S}^2 + g^3L^2)$. □

5. Analysis

In this section, we analyze the length of the pants decompositions from Section 3's LINEAR DECOMPOSITION algorithm and Section 4's GENUS DECOMPOSITION algorithm. We create a random combinatorial surface of size $2n$ by randomly shuffling the array $\begin{pmatrix}1 & 1 & 2 & 2 & \cdots & n & n\end{pmatrix}$. Then, we run our two algorithms on these random combinatorial surfaces. Note that in all of the following figures, the y-axis is the output of the respective algorithm, which finds the maximum value of $L_S(b)$ over all boundary components b of S, even when the axis label uses the word length. To obtain the actual length of the curves of pants decompositions, multiply the y-axis by two. This is a consequence of Lemma 2.3.

Figure 3a displays the average value of LINEAR DECOMPOSITION(S) for random combinatorial surfaces S given the genus of S. This figure uses 1,000 data points. The relationship appears linear, and it almost perfectly matches the upper bound of $4g - 4$ from Theorem 3.8, where g is the genus of S. So, LINEAR DECOMPOSITION rarely does



(A) LINEAR DECOMPOSITION. (B) GENUS DECOMPOSITION.

FIGURE 3. Average value of each algorithm over random combinatorial surfaces of a certain genus.

better than its upper bound. Since $\text{Area}(S) = \sqrt{3}g$, the data suggests that LINEAR DECOMPOSITION$(S) \sim C\sqrt{\text{Area}(S)g}$ for some constant $C > 0$. As Buser [2] proved that $\mathfrak{B}_S \leq C\sqrt{\text{Area}(S)g}$ for constant $C > 0$, it appears that the algorithm behaves very similarly to the upper bound.

The relationship between the genus of S and GENUS DECOMPOSITION(S) is less clear. In Figure 3b, we show the average value of GENUS DECOMPOSITION(S) for random combinatorial surfaces S given the genus of S. This figure uses 3,000 data points. Still, the figure suggests a strong linear relationship between the two variables.

Comparing the two algorithms in Figure 4 reveals that GENUS DECOMPOSITION produces significantly shorter pants decompositions than LINEAR DECOMPOSITION. In Figure 5 and Figure 6, we look at the lengths of the individual cuts in a decomposition. The linear aspects of LINEAR DECOMPOSITION are revealed in Figure 5a. Note that this pattern is what enables the proof of Theorem 3.8. This is in contrast with the more sporadic nature of the GENUS DECOMPOSITION algorithm.

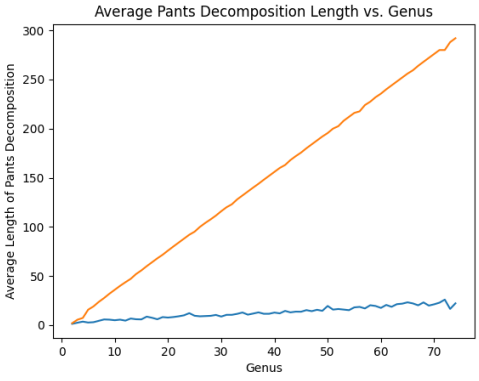
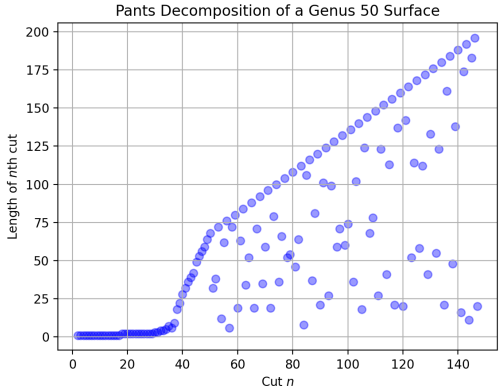
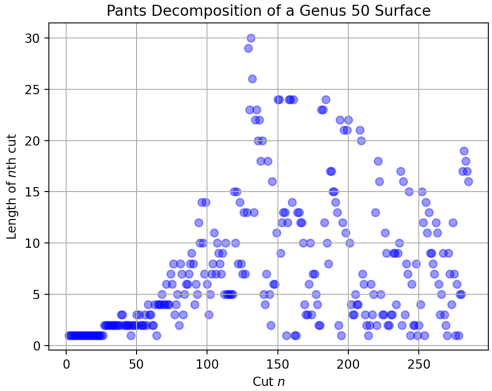


FIGURE 4. Average value of both LINEAR DECOMPOSITION(S), shown in orange, and GENUS DECOMPOSITION(S), shown in blue, for random combinatorial surfaces S given the genus of S.

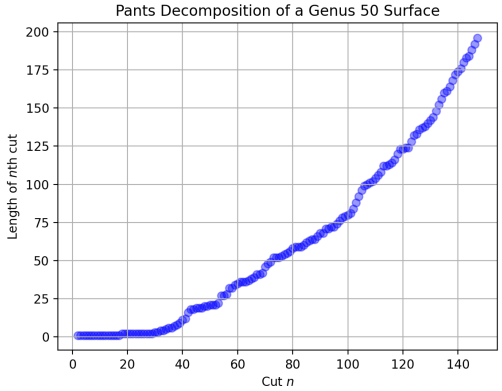


(A) LINEAR DECOMPOSITION.

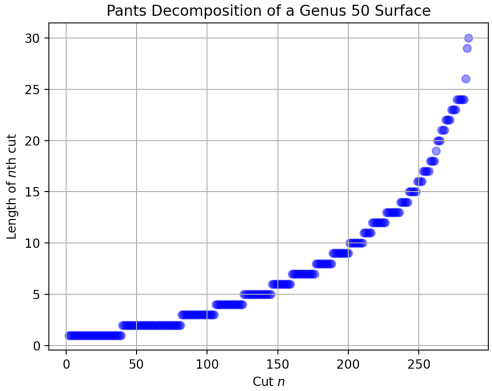


(B) GENUS DECOMPOSITION.

FIGURE 5. Lengths of the individual cuts in each algorithm, in the order the algorithm finds the cuts, for a fixed random combinatorial surface.



(A) LINEAR DECOMPOSITION.



(B) GENUS DECOMPOSITION.

FIGURE 6. Lengths of the individual cuts in each algorithm, sorted by the length of the cuts, for a fixed random combinatorial surface.

6. APPENDIX

In this appendix we prove that the surfaces in our model often have a large genus. We note that in [10] Brooks and Makover prove a tighter bound for the expected value of the genus for a random surface model that is similar to ours.

Theorem 6.1. *Suppose we identify all the edges of a $2n$-gon in pairs, uniformly at random. Then the expected value of the genus of the resulting surface is at least $\frac{1}{6}n - \tilde{C}\ln n + \frac{1}{2}$, for some constant $\tilde{C} > 0$.*

Proof. When we identify the edges of our $2n$-sided polygon, many of the vertices are identified. The identifications put a transitive relation on the vertices of the polygon. Let V be the number of equivalence classes of this relation. We can compute the Euler characteristic of the resulting surface to be $V - n + 1$, and so its genus is $\frac{n-V+1}{2}$. Our goal is to show that V is on average much smaller than n.

We identify the edges of our polygon step by step. On the k^{th} step, we randomly select two of the unidentified $2n-2(k-1)$ edges and identify them. We want to find an upper bound on the probability that the k^{th} identification identifies two vertices that are already identified, an event we call $R(k)$. Let e be the first edge we select on the k^{th} step and let e_1 and e_2 be the neighboring edges of e, where e intersects e_1 at vertex v_1 and e_2 at vertex v_2. Now for $R(k)$ to be true, the k^{th} identification must either identify a vertex to itself or to a different vertex that it is already identified to. The former case occurs when e is identified to a not-yet-identified neighbor, of which there are at most two, meaning the event occurs with probability at most $\frac{2}{2n-2(k-1)} = \frac{1}{n-k+1}$. The latter case occurs when either e_1 or e_2 is identified with another edge f, meaning a vertex w of f is already identified to v_1 or v_2, respectively. Then $R(k)$ occurs if e is identified with the other edge that shared vertex f. Since f could be the edge identified with either e_1 or e_2, there are once again at most two such possibilities, so the case occurs with probability at most $\frac{2}{2n-2(k-1)} = \frac{1}{n-k+1}$. As such,

$$\mathbb{P}[R(k)] \leq \frac{2}{n-k+1}.$$

Then the expected number of steps for which $R(k)$ occurs is at most,

$$\sum_{k=1}^{n} \mathbb{P}[R(k)] \leq \sum_{k=1}^{n} \frac{2}{n-k+1} = C \int_1^n \frac{1}{x} dx = C \ln n$$

for some constant $C > 0$. If a vertex does not lie on an edge for which $R(k)$ occurs, then it lies in an equivalence class with at least two other vertices. Every vertex lies on two edges, meaning at most $2C \ln n$ vertices lie on edges for which $R(k)$ occurs. Then the expected value of V is at most,

$$\frac{1}{3}(2n - 2C \ln n) + 2C \ln n \leq \frac{2}{3}n + \frac{4C}{3} \ln n.$$

Thus, for some constant $\tilde{C} > 0$, the expected value of the genus is at least,

$$\frac{1}{6}n - \tilde{C} \ln n + \frac{1}{2}.$$

□

Acknowledgements

I am extremely grateful to my PRIMES mentor, Elia Portnoy, for his excellent guidance throughout this project. He has been consistently kind and patient, and has been invaluable to my PRIMES experience. I am also very grateful to the MIT PRIMES-USA program for making this project possible.

References

[1] Peter Buser. Riemannshe flächen und längenspektrum vom trigonometrishen standpunkt. *Habilitation Thesis, University of Bonn*, 1981.

[2] Peter Buser. *Geometry and spectra of compact Riemann surfaces*. Birkhäuser Boston, 1992.

[3] Peter Buser and Mika Seppälä. Symmetric pants decompositions of Riemann surfaces. *Duke Mathematical Journal*, 67(1):39–55, 1992.

[4] Eric Colin de Verdiere, Alfredo Hubard, and Arnaud de Mesmay. Discrete systolic inequalities and decompositions of triangulated surface. *Discrete and Computational geometry*, 53, 2015.

[5] Misha Gromov. Systoles and intersystolic inequalities. *Institut des Hautes etudes Scientifiques*, 1992.
[6] Larry Guth. Metaphors in systolic geometry. *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010)*, I, 2010.
[7] Larry Guth, Hugo Parlier, and Robert Young. Pants decompositions of random surfaces. *Geometric and Functional Analysis*, 21, 2011.
[8] Hugo Parlier. A short note on short pants. *Canadian Mathematical Bulletin*, 57(4):870–876, 2014.
[9] Pao Ming Pu. Some inequalities in certain nonorientable riemannian manifolds. *Pacific J. Math*, 2:55–71, 1952.
[10] Eran Makover Robert Brooks. Random construction of riemann surfaces. *J. Differential Geom.*, 68:121–157, 2004.

Princeton High School, Princeton New Jersey, 08540
Email address: nicholas.d.hagedorn@gmail.com