CHAPTER 3

Homotopy

[Q]uæ quidem mentes si aliquam ex. gr. parabolæ proprietatem intime perspicerent, atque intuerentur, non illud quærerent, quod nostri Geometræ quærunt, ut parabolam rectificarent, sed, si ita loqui fas est, ut rectam parabolarent.

[Indeed, these minds, if they noticed & formed an extremely clear perception of some property of, say, the parabola, would not seek, as our geometricians do, to rectify the parabola ; they would endeavor, if one may use the words, to parabolify a straight line.*]*

— Ruđer Josip Bošković *Theoria philosophiae naturalis redacta ad unicam legem virium in natura existentium* (1763)
Enhlish translation (*A Theory of Natural Philosophy*) by J. M. Child (1921)

When you turn the corner
And you run into yourself
Then you know that you have turned
All the corners that are left.

— Langton Hughes, "Final Curve" (1951)

Motivation: River routing in VLSI, map simplification

3.1 Groundwork

Definitions

Recall that a ***path*** in a topological space X is a continuous function from the unit interval $[0,1]$ to X. A ***path homotopy*** in X between two paths π and π' is a continuous

path homotopic
alpha homotopic beta
concatenation of two paths
alpha dot beta
reversal of a path
bar pi
loop
basepoint

function $h\colon [0,1]\times[0,1]\to X$ such that $h(0,t)=\pi(t)$ and $h(1,t)=\pi'(t)$ for all t, and $h(s,0)=\pi(0)=\pi'(0)$ and $h(s,1)=\pi(1)=\pi'(1)$ for all $s\in[0,1]$. (We will omit the prefix 'path-' when it is clear from context.) For all $t\in[0,1]$, the function $s\mapsto h(s,t)$ is a path from $\pi(0)$ to $\pi(1)$.

Two paths π and π' are ***(path) homotopic*** in X if there is a path homotopy between them; we write $\boldsymbol{\pi\simeq_X\pi'}$ to denote that π and π' are homotopic in X. If the space X is clear from context, we simply write $\pi\simeq\pi'$. For example, any two paths in the plane with the same endpoints are homotopic in the plane; specifically, for any two paths $\pi,\pi'\colon[0,1]\to\mathbb{R}^2$, the linear interpolation map $h(s,t)=(1-t)\pi(s)+t\pi'(s)$ is a homotopy from π to π'.

It is often helpful to think of the second argument of a homotopy as "time", and the homotopy itself as a movie of one path continuously morphing into the other. On the other hand, it is *also* helpful to think of a homotopy as a map from a closed topological disk into X that satisfies certain boundary conditions. Proofs involving homotopy freely alternate between these two intuitions.

Important Properties

The following pair of lemmas imply that we can sensibly discuss path homotopy without worrying about any particular parametrization of the paths or any particular representation of the underlying space.

Lemma 3.1 (Invariance). *For any two paths $\pi,\pi'\colon[0,1]\to X$ and any continuous map $\Phi\colon X\to Y$, if $\pi\simeq_X\pi'$, then $\Phi\circ\pi\simeq_Y\Phi\circ\pi'$.*

Proof: For any homotopy $h\colon[0,1]^2\to X$ from π to π', the function $\Phi\circ h\colon[0,1]^2\to Y$ is a homotopy from $\Phi\circ\pi$ to $\Phi\circ\pi'$. □

Lemma 3.2 (Reparametrization). *For any two paths $\pi,\pi'\colon[0,1]\to X$ and any continuous map $\alpha\colon[0,1]\to[0,1]$ such that $\alpha(0)=0$ and $\alpha(1)=1$, we have $\pi\circ\alpha\simeq_X\pi$.*

Proof: The function $h(s,t)=t\pi(\alpha(t))+(1-t)\pi(t)$ is a homotopy from π to $\pi\circ\alpha$. □

Any two paths π and σ with $\pi(1)=\sigma(0)$ can be ***concatenated*** to form a longer path from $\pi(0)$ to $\pi(1)$; formally, we define the concatenation $\boldsymbol{\pi\cdot\sigma}$ as follows:

$$(\pi\cdot\sigma)(t):=\begin{cases}\pi(2t) & \text{if } t\le 1/2,\\ \sigma(2t-1) & \text{if } t\ge 1/2.\end{cases}$$

The ***reversal*** $\boldsymbol{\overline{\pi}}$ of a path π is the path $\overline{\pi}(t):=\pi(1-t)$. We immediately observe that $\overline{\pi\cdot\sigma}=\overline{\sigma}\cdot\overline{\pi}$.

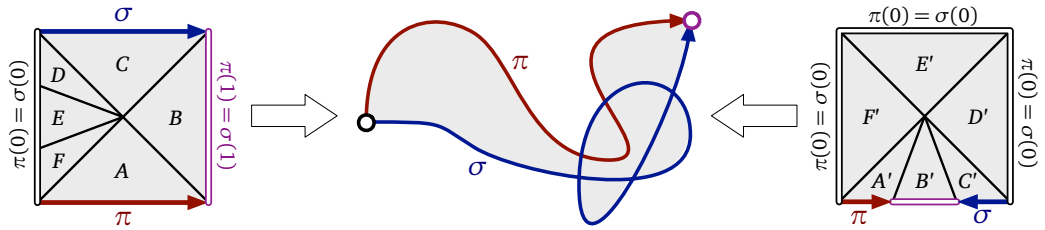
A ***loop*** is a path whose endpoints coincide; this common endpoint is called the loop's ***basepoint***. Two loops are homotopic in X if they are homotopic in X as paths; in

particular, any loop homotopy must fix the basepoint. A loop ℓ is ***contractible*** in X if it is homotopic in X to the constant loop $t \mapsto \ell(0)$. A homotopy in X from a contractible loop ℓ to the constant loop at its basepoint $\ell(0)$ is called a ***contraction*** of ℓ.

contractible
contraction
exercises for the reader
homotopy class
bracket alpha

Lemma 3.3. *Two paths π and σ with common endpoints are homotopic if and only if the loop $\pi \cdot \overline{\sigma}$ is contractible.*

Proof: Let T be the triangulation of the square $[0,1] \times [0,1]$ by a cycle of six triangles sharing a common vertex $(1/2, 1/2)$, with outer vertices $(0,0)$, $(0,1)$, $(1,1)$, $(1,0)$, $(2/3,0)$, $(1/3,0)$, as shown in the top left of the figure below. Let T' be a similar triangulation, but with outer vertices $(0,0)$, $(0,1/3)$, $(0,2/3)$, $(0,1)$, $(1,1)$, $(1,0)$, as shown in the bottom left of the figure below. Finally, let $\phi : [0,1]^2 \to [0,1]^2$ denote the piecewise-linear map that fixes $(0,0)$ and maps triangulation T to triangulation T', as indicated in the figure. The map ϕ is clearly a homeomorphism.



A path homotopy from π to σ is homotopic to a contraction of $\pi \cdot \overline{\sigma}$.

Now suppose $h\colon [0,1]^2 \to X$ is a path homotopy from π to σ. Straightforward definition-chasing implies that the map $h \circ \phi : [0,1]^2 \to X$ is a contraction of (a reparametrization of) the loop $\pi \cdot \overline{\sigma}$. Similarly, if $h' : [0,1] \times [0,1] \to X$ is a contraction of $\pi \cdot \overline{\sigma}$, then $h' \circ \phi^{-1}$ is a path homotopy from π to σ. □

The following lemmas are left as a series of straightforward but instructive exercises for the reader, similar to the proof of Lemma 3.3.

Lemma 3.4. *For any space X, path homotopy $\simeq_X$ is an equivalence relation. That is, for any paths $\pi, \pi', \pi'' : [0,1] \to X$ with the same endpoints, the following hold.*

(a) ***Reflexivity:*** *$\pi \simeq_X \pi$.*

(b) ***Symmetry:*** *If $\pi \simeq_X \pi'$, then $\pi' \simeq_X \pi$.*

(c) ***Transitivity:*** *If $\pi \simeq_X \pi'$ and $\pi' \simeq_X \pi''$, then $\pi' \simeq_X \pi$.*

We refer to the equivalence classes under path homotopy as ***homotopy classes***, and we write $[\pi]$ to denote the homotopy class of any path π.

Lemma 3.5. *For any paths $\pi, \pi', \sigma, \sigma' : [0,1] \to X$ with appropriately matching endpoints, the following hold.*

pi cdot sigma
pi bar
fundamental group
fundamental groupoid
retraction
id X
deformation retraction
deformation retract

(a) ***Reversal:*** *If* $\pi \simeq \pi'$*, then* $\overline{\pi} \simeq \overline{\pi'}$.

(b) ***Concatenation:*** *If* $\pi \simeq \pi'$ *and* $\sigma \simeq \sigma'$*, then* $\pi \cdot \sigma \simeq \pi' \cdot \sigma'$.

Lemma 3.5 implies that reversal and concatenation of homotopy classes are well-defined; we extend our notation by defining $[\pi] \cdot [\sigma] := [\pi \cdot \sigma]$ and $\overline{[\pi]} := [\overline{\pi}]$.

Lemma 3.6. *For any paths* $\pi, \sigma, \tau : [0,1] \to X$ *with appropriately matching endpoints, the following hold.*

(a) ***Identity:*** *If* σ *is a contractible loop, then* $\pi \cdot \sigma \simeq \pi$.

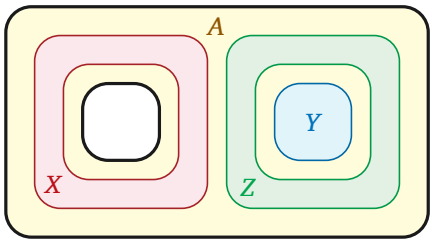
(b) ***Inverse:*** *The loops* $\pi \cdot \overline{\pi}$ *and* $\overline{\pi} \cdot \pi$ *are contractible.*

(c) ***Associativity:*** $(\pi \cdot \sigma) \cdot \tau \simeq \pi \cdot (\sigma \cdot \tau)$.

Lemma 3.6 implies that for any point $x \in X$, the set of homotopy classes of *loops with basepoint* x define a group under concatenation, with the class of contractible loops as the identity element. This group is called the *fundamental group* of X with basepoint x, and denoted $\pi_1(X, x)$. The set of homotopy classes of *all* paths in X do not define a group, because we cannot concatenate arbitrary paths, but rather a weaker algebraic structure called the *fundamental groupoid*.

Retractions

Finally, we define a useful class of maps from spaces to subspaces. A ***retraction*** from a space X to a subspace $Y \subseteq X$ is a continuous function $r : X \to Y$ whose restriction to Y is the identity map $\mathbf{id}_Y$ (defined as $\mathrm{id}_Y(y) = y$ for every point $y \in Y$). A retraction r is called a ***deformation retraction*** if there is a continuous function $R : [0,1] \times X \to X$ such that $R(0, \cdot)$ is the identity map on X and $R(1, \cdot) = r$. More succinctly, a deformation retraction is a retraction that is homotopic to the identity map. Whenever there is a (deformation) retraction from X to Y, we call Y a ***(deformation) retract*** of X.



X is a deformation retract of A; Y is a retract of A; and Z is neither.

For example, in the figure above, X, Y, Z are three subsets of an annulus A; specifically, X and Z are annuli and Y is a disk. It is not hard to prove that X and Y are retracts of A, and moreover, that X is a deformation retract. However, neither Y nor Z is a deformation retract of A, and Z is not even a retract of A; these facts can be proved by applying the following pair of lemmas.

cycle
circle
free homotopy
freely homotopic
free contraction
One can mechanically verify

Lemma 3.7. *For any retract Y of X, any loop ℓ in Y is contractible in Y if and only if ℓ is contractible in X.*

Proof: Fix a loop ℓ in Y. If h is a contraction of ℓ in Y, then trivially $i \circ h$ is a contraction of ℓ in X, where $i\colon Y \hookrightarrow X$ is the obvious inclusion map. On the other hand, if h is a contraction of ℓ in X, then for any retraction $r\colon X \to Y$, the map $r \circ h$ is a contraction of ℓ in Y. □

Lemma 3.8. *Let r be a deformation retraction from X to Y. Any loop ℓ in X is contractible in X if and only if the loop $r \circ \ell$ is contractible in Y.*

Proof: If $h\colon [0,1]^2 \to X$ is a contraction of ℓ, then $r \circ h\colon [0,1]^2 \to Y$ is a contraction of $r \circ \ell$. On the other hand, suppose $h'\colon [0,1]^2 \to Y$ is a contraction of $r \circ \ell$. Fix a map $R\colon [0,1] \times X \to X$ such that $R(0,\cdot) = \mathrm{id}_X$ and $R(1,\cdot) = r$. Then the map $h\colon [0,1]^2 \to X$ defined by setting $h(s,t) := R(s, h'(s,t))$ is a contraction of ℓ. □

Cycles and Free Homotopy

Recall that a ***cycle*** in a topological space X is a continuous function from the circle S^1 to X. For notational convenience, we measure angles on the circle S^1 in units of ***circles***, instead of radians or degrees; equivalently, we define the circle as the quotient space $[0,1]/(0 \sim 1)$ obtained by identifying the endpoints of the unit interval. Then any loop $\ell\colon [0,1] \to X$ can be transformed into an equivalent cycle $\ell^\circ\colon S^1 \to X$, or vice versa, by setting $\ell^\circ(t) = \ell(t)$.

A ***free homotopy*** between two cycles α and β is a continuous function $h\colon [0,1] \times S^1 \to X$ such that $h(0,\theta) = \alpha(\theta)$ and $h(1,\theta) = \beta(\theta)$ for all $\theta \in S^1$. Two cycles are ***(freely) homotopic*** if there is a free homotopy from one to the other. A cycle is (freely) contractible if it is homotopic to a constant cycle; a free homotopy between a cycle and a point is called a ***free contraction***.

Lemma 3.9. *A loop ℓ is path-contractible if and only if the corresponding cycle ℓ° is freely contractible.*

Proof: If $h\colon [0,1] \times [0,1] \to X$ is a path-contraction of ℓ, then the function $h^\circ\colon [0,1] \times S^1 \to X$ defined by $h^\circ(s,t) = h(s,t)$ is a free contraction of ℓ°.

On the other hand, let $h^\circ\colon [0,1] \times S^1 \to X$ be a free contraction of ℓ°. One can mechanically verify that the function $h\colon [0,1] \times [0,1] \to X$ defined by

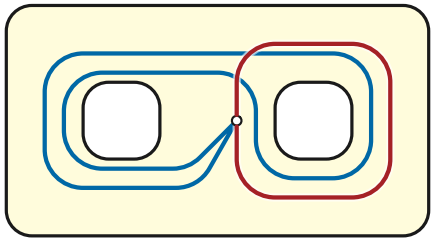
$$h(s,t) = \begin{cases} h\left(2s,\ \dfrac{1}{2} + \dfrac{2t-1}{2s-1}\right) & \text{if } s < t < 1-s, \\ h\left(\max\left\{2-2s, 2-2t, 2t\right\},\ 0\right) & \text{otherwise.} \end{cases}$$

is a path-contraction of ℓ. Intuitively, we modify the free contraction h° so that the basepoint $\ell(0)$ stays fixed and connected to the rest of the evolving cycle by a "tail"

One can verify mechanically
crossing sequence
reduced crossing sequence

consisting of a path and its reversal; after the cycle contracts to a point, we contract its "tail" back to the basepoint. □

Similarly, any path homotopy between two loops α and β can be interpreted as a free homotopy between the equivalent cycles α° and β°. However, two loops can be freely homotopic but not path-homotopic, even if they share a common basepoint.



Freely homotopic loops that are not path-homotopic

Arcs and Relative Homotopy

Arcs (paths between boundary points) and homotopy rel boundary???

3.2 Testing Homotopy

In this section, we describe a simple algorithm to determine whether a given polygonal loop in a polygon with holes is contractible, or equivalently, whether two given polygonal paths are homotopic. We need to consider polygons with holes, because homotopy in *simple* polygons is completely trivial.

Lemma 3.10. *Let P be an arbitrary simple polygon. Every loop in P is contractible.*

Proof: The Jordan-Schönflies theorem implies that P is homeomorphic to an arbitrary triangle, so by Lemma 3.1, we can assume without loss of generality that P *is* a triangle. Let $\ell : [0,1] \to P$ be an arbitrary loop in P. Fix a coordinate system with the origin inside P, and let $h : [0,1]^2 \to P$ be the map $h(s,t) := (1-s) \cdot \ell(t)$. One can verify mechanically that h is a contraction of ℓ. □

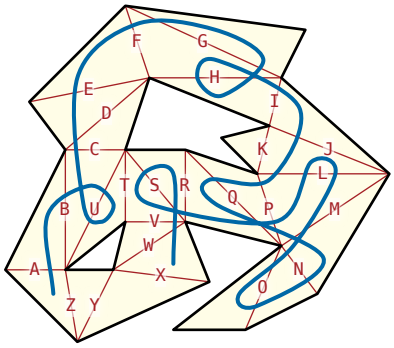
In a polygon with holes, however, two paths can have the same endpoints without being homotopic, intuitively by "winding around" the holes in different ways.

Our algorithm works in several phases. First, in a preprocessing phase, we compute a frugal triangulation of the polygon with holes, using (for example) the algorithm described in Chapter 1. Second, we compute the *crossing sequences* of the given polygonal paths, which record the sequence of diagonals that each path crosses. Third, we *reduce* the crossing sequence by repeatedly removing adjacent pairs of identical symbols. Finally, we report that the two paths are homotopic if and only if their reduced crossing sequences are identical.

cross
crossing sequence
X Delta pi
One can verify mechanically
canonical path

Crossing Sequences

Let Δ be a triangulation of a polygon P with holes, let π be a polygonal path in P. We say that π ***crosses*** an edge e of the triangulation if some subpath of π is contained in e and the edges of π preceding and following that subpath lie on opposite sides of the line through e. The ***crossing sequence*** $X_\Delta(\pi)$ of π is the sequence of edges of Δ that π crosses, in order along π.



A path with crossing sequence ABUUCDEFGGHHHIJLPQQPMNOONMLLPQRSSSVW

This definition is precise as long as π avoids the vertices of P, but if π passes through a vertex v, it may cross several edges incident to v simultaneously. Although it is possible (but rather messy) to resolve this ambiguity symbolically, it is more convenient to perturb π to a homotopic path π' that avoids the vertices, and then *define* the crossing sequence of π to be the crossing sequence of π'. To define π', we first ensure that vertices of P intersect only vertices of π, by subdividing edges of π that pass through concave vertices of P if necessary; we then move any vertex of π that lies on a vertex of P a small distance $\varepsilon > 0$ into the interior of P in an appropriate direction, for example along the ray that bisects the angle between the incident edges of P. One can verify mechanically that the resulting path π' is homotopic to π and lies in P if ε is sufficiently small, and that the resulting crossing sequence is independent of ε.

Lemma 3.11. *Any two paths with the same endpoints and the same crossing sequence are homotopic.*

Proof: To prove the lemma, given any path π, we construct a ***canonical path*** π' with the same endpoints and the same crossing sequence as π, and then argue that π and π' are homotopic.

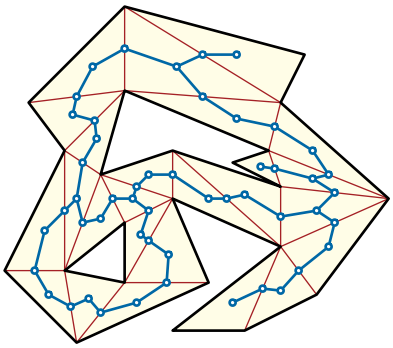
Let p and q be the endpoints of π, and let $e_1, e_2, \ldots, e_x$ be its crossing sequence. Let $\triangle_0$ denote the triangle containing p, and for each index $i > 0$, let $\triangle_i$ denote the triangle that shares edge e_i with $\triangle_{i-1}$. By construction, $q \in \triangle_x$. For each index i, let c_i denote the centroid of $\triangle_i$, and let m_i denote the midpoint of e_i. The canonical path π' is the polygonal path with vertices $p, c_0, m_1, c_1, m_2, \ldots, c_{x-1}, m_x, c_x, q$.

planar straight-line graph
skeleton
exercise of the reader
reduced crossing sequence
bigon
elementary reduction
elementary expansion

To complete the proof, we describe a homotopy from π to π'. Choose a sequence of points $p = p_0, p_1, \ldots, p_x = q$ in order along π, where each p_i lies in $\triangle_i$. (If π passes through a vertex of P, then some consecutive points $p_i, p_{i+1}, \ldots, p_j$ may all coincide with that vertex.) Next, for each i in order from 1 to x, replace the subpath of π from p_{i-1} to p_i with the polygonal chain p_{i-1}, m_i, p_i. Because the old and new subpaths both lie in the quadrilateral $\triangle_{i-1} \cup \triangle_i$, they are homotopic. Finally, for each i in order from 1 to $x-1$, replace the subchain m_i, p_i, m_{i+1} with m_i, c_i, m_{i+1}; these two subchains are homotopic in $\triangle_i$. □

Except for its first and last segments, every canonical path lies in a planar straight-line graph we call the ***skeleton*** of P, obtained by replacing each triangle $\triangle$ in the triangulation with one, two, or three line segments, each joining the centroid of $\triangle$ with the midpoint of an edge of $\triangle$ that is a diagonal of P. We leave the following useful lemma as a straightforward exercise of the reader.

Lemma 3.12. *The skeleton of any polygon P with holes is a deformation retract of P.*

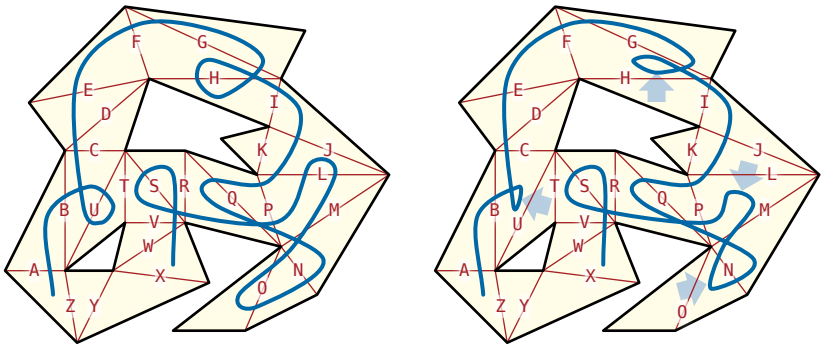


The skeleton of a polygon with holes.

Reduction

It is not hard to see that the converse of Lemma 3.11 is false; even in a triangulated polygon *without* holes, contractible loops can have nontrivial crossing sequences. However, if we ***reduce*** crossing sequences by repeatedly removing adjacent pairs of identical symbols, the resulting reduced crossing sequence of any path precisely encodes its homotopy class. Intuitively, each such reduction corresponds to continuously removing a ***bigon*** formed by the path and one of the edges of the triangulation.

Let us define reduction more carefully. We regard the crossing sequence as a finite string of abstract symbols. Let $x \cdot y$ denote the concatenation of any two strings x and y, and let ε denote the empty string. An ***elementary reduction*** is a transformation of the form $x \cdot aa \cdot y \mapsto x \cdot y$, where x and y are strings and a is a single symbol; an ***elementary expansion*** is the reverse transformation $x \cdot y \mapsto x \cdot aa \cdot y$. An ***elementary***



Tightening a path reduces its crossing sequence.

elementary transformation
equivalent
One can verify mechanically
trivial string
reduced string

transformation is either an elementary reduction or an elementary expansion. We call two strings ***equivalent*** if one can be transformed into the other by a finite sequence of elementary transformations. One can verify mechanically that equivalence is in fact an equivalence relation. We call a string ***trivial*** if it is equivalent to the empty string. Finally, a string is ***reduced*** if no symbol occurs twice in a row, and thus no elementary reductions are possible.

For example, the string EACBAABCEDDEADBCCBAEDDEADE is trivial; two different sequences of elementary reductions are shown below. In the first sequence, each elementary reduction removes the *leftmost* matching pair; the second sequence is more haphazard. In fact, as the following lemma implies, *any* sequence of elementary reductions eventually reduces this string to nothing.

EACB/\/\BCEDDEADBCCBAEDDEADE	EACBAABCE//EADBCCBAEDDEADE
EAC//CEDDEADBCCBAEDDEADE	EACBAABCEEADBCCBAE//EADE
EA/\/\EDDEADBCCBAEDDEADE	EACBAABC/\/\ADBCCBAEEADE
EAE//EADBCCBAEDDEADE	EACBAABCADBCCBA/\/\ADE
EA/\/\ADBCCBAEDDEADE	EACB/\/\BCADBCCBAADE
E/\/\DBCCBAEDDEADE	EACBBCADBCCB/\/\DE
EDB/\/\BAEDDEADE	EACBBCADB/\/\BDE
ED//AEDDEADE	EAC//CADBBDE
EDAE//EADE	EACCAD//DE
EDA/\/\ADE	EA/\/\ADDE
ED/\/\DE	EAA//E
E//E	E/\/\E
/\/\	/\/\

Reducing a trivial string in two different ways.

Lemma 3.13. *Every string is equivalent to exactly one reduced string.*

Proof: Let w be a string, and let $w \mapsto x$ and $w \mapsto y$ be distinct elementary reductions, meaning two different pairs of symbols are deleted. We claim that either $x = y$ or there is another string z such that $x \mapsto z$ and $y \mapsto z$ are elementary reductions. There are two cases to consider.

compactness argument

- If the pairs of symbols deleted by the elementary reductions $w \mapsto x$ and $w \mapsto y$ are disjoint, then we can write $w = w_1 \cdot aa \cdot w_2 \cdot bb \cdot w_3$ for some (possibly empty) substrings w_1, w_2, w_3 and some (possibly equal) symbols a and b, and without loss of generality, we have $x = w_1 \cdot w_2 \cdot bb \cdot w_3$ and $y = w_1 \cdot aa \cdot w_2 \cdot w_3$. In this case, we can take $z = w_1 \cdot w_2 \cdot w_3$.
- On the other hand, if the elementary reductions $w \mapsto x$ and $w \mapsto y$ delete overlapping pairs of symbols, they must be pairs of the *same* symbol. In this case, we can write $w = w_1 \cdot aaa \cdot w_2$ for some (possibly empty) substrings w_1 and w_2 and some symbol a, and we have $x = y = w_1 \cdot a \cdot w_2$.

Now consider two equivalent strings $x \neq y$. Let $x = w_1 \leftrightarrow w_2 \leftrightarrow \cdots \leftrightarrow w_n = y$ be a sequence of strings such that each adjacent pair is connected by an elementary reduction $w_i \mapsto w_{i+1}$ or an elementary expansion $w_i \leftarrow w_{i+1}$. Suppose for some index i, we have $w_{i-1} \mapsto w_i \leftarrow w_{i+1}$. If $w_{i-1} = w_{i+1}$, then we can remove w_{i-1} and w_i to obtain a shorter transformation sequence. Otherwise, there is another string z_i such that $w_{i-1} \mapsto z_i \leftarrow w_{i+1}$. Thus, by induction, we can modify the transformation sequence so that every elementary reduction appears before every elementary expansion. Let z be the shortest string in the resulting sequence. Both x and y can be transformed into z by elementary reductions. Because $x \neq y$, either $x \neq z$ or $y \neq z$; we conclude that at most one of x and y is reduced. □

Back to Homotopy

Now we claim that two paths with the same endpoints are homotopic if and only if their reduced crossing sequences are equal. One direction is straightforward.

Lemma 3.14. *Any two paths with the same endpoints and the same reduced crossing sequence are homotopic.*

Proof: Let α and β be two paths in P with the same endpoints, whose crossing sequences differ by canceling exactly one pair $e_i e_i$. We claim that α and β are homotopic; the lemma follows from this claim by induction. By Lemmas 3.8 and 3.12, we can assume without loss of generality that α and β are *canonical* paths. But then π must have the polygonal chain $c_{i-1}, m_i, c_i, m_i, c_{i-1}$ as a subpath; moreover, α can be transformed into β by contracting that subpath to the point c_{i-1}. □

The converse of Lemma 3.14 requires a bit more effort to prove, because we must consider *arbitrary* homotopies. Even if we are only interested in transforming one polygonal path into another, the definition of homotopy allows intermediate paths to be arbitrarily complex or pathological. Most of the proof is a compactness argument, which implies that any two homotopic paths are connected by a "nice" homotopy.

Lemma 3.15. *Any two homotopic paths have the same reduced crossing sequence.*

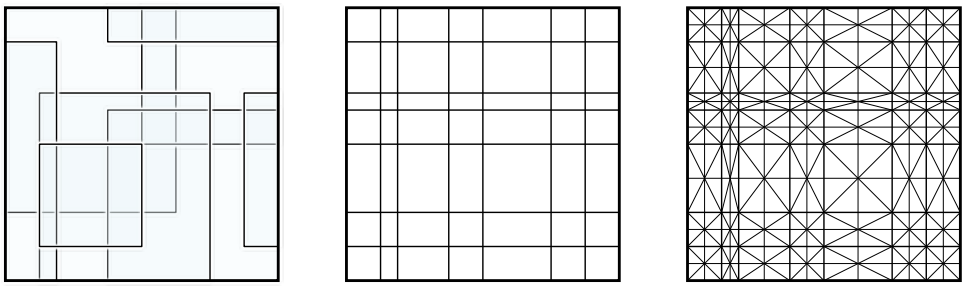
open neighborhood
open cover
compact
finite cover

Proof: Again, it suffices to prove that any contractible loop $\ell\colon [0,1] \to P$ has an empty reduced crossing sequence. In light of Lemma 3.11, we assume without loss of generality that ℓ is a *canonical* loop; moreover, we can assume that the basepoint of ℓ is the centroid of some triangle. Let $h\colon [0,1]^2 \to P$ be a contraction of ℓ. Lemmas 3.8 and 3.12 imply that without loss of generality, we can assume that the image of h lies entirely in the skeleton S of P.

For each triangle $\triangle_k$ in the triangulation of P, let U_k be the intersection of an open neighborhood of $\triangle_k$ with S. These open neighborhoods cover S. Moreover, because S is bounded away from the vertices of P, we can choose these neighborhoods so that any pairwise intersection $U_j \cap U_k$ is non-empty if and only if the corresponding triangles $\triangle_j$ and $\triangle_k$ share a common edge. In particular, every triple intersection $U_i \cap U_j \cap U_k$ is empty. We can also ensure that centroid of any triangle $\triangle_k$ lies only in the corresponding neighborhood U_k.

Any point in the parameter square $[0,1]^2$ lies in an open rectangle $\square = (a,b)\times(c,d)$ such that $h(\square \cap [0,1]^2)$ lies entirely inside some triangle neighborhood U_k. The set of all such rectangles defines an open cover of $[0,1]^2$. The compactness of $[0,1]^2$ implies (by definition!) that some *finite* subset of these open rectangles covers $[0,1]^2$. By shrinking each rectangle slightly, we obtain a finite cover of $[0,1]^2$ by *closed* rectangles. Finally, by extending each side of each of these closed rectangles across the entire parameter square, we partition the square into a *grid*, such that each closed grid cell maps to some triangle neighborhood. More concretely, there are two finite sequences $0 = s_0 < s_1 < \cdots < s_\sigma = 1$ and $0 = t_0 < t_1 < \cdots < t_\tau = 1$ of real numbers, such that for any indices i and j, the homotopy h maps the entire closed rectangle $\square_{i,j} = [s_{i-1}, s_i] \times [t_{j-1}, t_j]$ into some neighborhood U_k. For each i and j, choose one such neighborhood and call it $U_{i,j}$, and let $\triangle_{i,j}$ denote the corresponding triangle. Without loss of generality, every vertex of ℓ is equal to $\ell(t_j)$ for some index j.

For any two grid cells $\square_{i,j}$ and $\square_{i',j'}$ with non-empty intersection, the corresponding neighborhoods $U_{i,j}$ and $U_{i',j'}$ must also have non-empty intersection. In other words, neighboring cells in the grid map either to the neighborhoods of adjacent triangles or to the same neighborhood.



Constructing a grid triangulation of $[0,1]^2$ from an open rectangular cover.

Now we construct a piecewise-linear function $h'\colon [0,1]^2 \to S$ as follows. First

barycentric subdivision
One can verify mechanically
monotone

we construct the barycentric subdivision of the grid, by splitting each cell into eight triangles; the vertices of each triangle are the centroid $c_{i,j}$ of a grid cell $\square_{i,j}$, the midpoint of one of the edges of $\square_{i,j}$, and a vertex of that edge. For any indices i and j, we define $h'(c_{i,j})$ to be the centroid of the corresponding triangle $\triangle_{i,j}$. For any other vertex x of the grid triangulation, there are two possibilities. If all grid cells incident to x map to the same triangle neighborhood, we define $h'(x)$ to be the centroid of that triangle. Otherwise, the grid cells incident to x map into exactly two triangle neighborhoods; in this case, we define $h'(x)$ to be the midpoint of the edge between those two triangles. Finally, h' is affine on each triangle in the subdivision. One can verify mechanically the following claims:

- h' maps each triangle in the subdivision onto either a vertex or an edge of the skeleton S. (Thus, the image of h' lies in S, as claimed.)
- h' maps any path from the left edge of $[0,1]^2$ to the right edge along edges of the subdivision to a *canonical* loop in P.
- $h'(0,\cdot)$ is a contraction of (a reparametrization of) ℓ.

Call a path in the subdivided grid *monotone* if it starts on the left edge, ends on the right edge, and never moves to the left. Now consider a sequence of monotone subdivision paths, starting with the bottom edge of $[0,1]^2$ and ending with the top edge, such that adjacent paths differ by only one vertex. The crossing sequences of these paths (or rather, their images under h') evolve by elementary transformations. Thus, we obtain a sequence of elementary transformations that reduces the crossing sequence of ℓ to the empty string. □

The correspondence between homotopy and reduced crossing sequences has an important corollary that (as often happens in topology) is obvious in hindsight but surprisingly difficult to prove.[1]

Corollary 3.16. *Every polygon with holes contains a simple non-contractible loop.*

Proof: Let P be a polygon with holes; let Δ be an arbitrary triangulation of P; let P_i be one of the holes of P; and let ℓ denote a simple loop that traverses ∂P_i exactly once.

The triangulation Δ must contain an edge with exactly one endpoint in ∂P_i. This edge appears exactly once in the crossing sequence of ℓ (with respect to Δ), and therefore in the *reduced* crossing sequence of ℓ. We conclude that the reduced crossing sequence of ℓ is non-empty; Lemma 3.15 implies immediately that ℓ is non-contractible in P. □

Algorithm

Finally, we have all the ingredients of an algorithm to determine whether two polygonal paths in a polygon with holes are homotopic.

freely homotopic
.

Lemma 3.17. *Any string of length x can be reduced in $O(x)$ time.*

Proof: Lemma 3.13 implies that repeatedly deleting the leftmost matching pair correctly reduces the sequence. Assuming the input sequence is stored in a standard array, a naïve implementation of left-greedy reduction would run in $O(x^2)$ time, but we can improve the running time to $O(x)$ by storing the portion of the crossing sequence that is already reduced in a stack. We add a special "fencepost" symbol $\bullet$ to the beginning of the output array $\bar{X}$ to avoid boundary cases; this symbol is different from every edge label. □

```
LeftGreedyReduce(X[1..x]):
  x̄ ← 0
  X̄[0] ← •                    ⟨⟨fencepost⟩⟩
  for i ← 1 to x
      if X[i] = X̄[x̄]
          x̄ ← x̄ − 1           ⟨⟨pop⟩⟩
      else
          x̄ ← x̄ + 1
          X̄[x̄] ← X[i]         ⟨⟨push⟩⟩
  return X̄[1..x̄]
```

The linear-time left-greedy reduction algorithm.

Theorem 3.18. *Let P be polygon with holes with n vertices, and let α and β be polygonal paths in P with k edges in total. We can determine whether α and β are homotopic in P in $O(n \log n + nk)$ time.*

Proof: In a preprocessing phase, we triangulate P in $O(n \log n)$ time using the sweepline algorithm described in Chapter 1. We compute the crossing sequences of the loop $\alpha \cdot \bar{\beta}$ in $O(k + x)$ time, where x is the length of the crossing sequence, by simply walking along the loop, spending constant time at each vertex and at each crossing. Lemma 3.17 implies that we can reduce this crossing sequence in $O(x)$ time. Finally, we report that α and β are homotopic if and only if the reduced crossing sequence is empty. Each edge of α and β crosses every edge of the triangulation of P at most once, so $x = O(nk)$. □

3.3 Variations

Our homotopy-testing algorithm can be generalized and improved in several ways.

Cycles

With only minor modifications, the same algorithm can be used to determine whether two polygonal *cycles* are freely homotopic. The biggest change is that the crossing sequence of a cycle is not a string, but a ***cyclic string*** Formally, a cyclic string is an equivalence class of strings, where for any strings w and x, the concatenated

cyclically reduced
fence
signed crossing sequence
elementary reduction

strings $w \cdot x$ and $x \cdot w$ are cyclically equivalent; for example, `WE_ARE_NOWHERE_` and `HERE_WE_ARE_NOW` represent the same cyclic string. A cyclic string is ***cyclically reduced*** if no character occurs twice in a row; in particular, the "first" and "last" characters must be different.

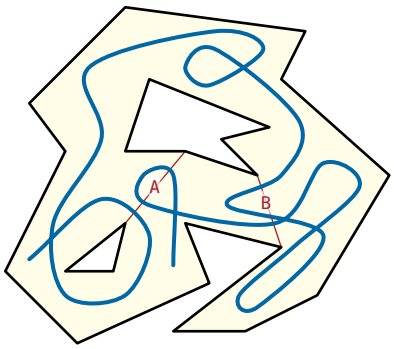
Easy modifications of our earlier arguments imply that every cyclic string of length x has a unique cyclic reduction, which can be computed in $O(x)$ time, and that two cycles are freely homotopic if and only if their cyclically reduced crossing sequences are identical. Two strings x and y are equivalent if and only if they have the same length and x is a substring of $y \cdot y$. The last condition can be tested in linear time using a string-matching algorithm by either Knuth *et al.* [19] or Boyer and Moore [5, 17].

Theorem 3.19. *Let P be polygon with holes with n vertices, and let α and β be polygonal cycles in P with k edges in total. We can determine whether α and β are freely homotopic in P in $O(n \log n + nk)$ time.*

Fences

We do not actually need to shatter the polygon with holes P into triangles; it is enough to find a collection of interior-disjoint segments, which we call ***fences***, that cut the interior of P into a single open disk. Intuitively, the fences define a tree whose vertices are the boundary components of P; thus, if P has h holes, we need exactly h fences. It is easy to extract h suitable fences from a triangulation or trapezoidal decomposition of P in $O(n)$ time.

Since a non-contractible loop can intersect the same fence twice in a row, we must compute and reduce ***signed*** crossing sequences, which record not only which fences a path crosses, but also in which direction. An elementary reduction of a signed crossing sequence removes an adjacent pair of identical symbols *with opposite signs*. Lemmas 3.13 and 3.17 generalize to signed crossing sequences with only minor modifications.



A path with signed crossing sequence $\mathtt{A}^-\mathtt{A}^+\mathtt{B}^+\mathtt{B}^-\mathtt{B}^+\mathtt{A}^-$

These signed crossing sequences have length at most $O(hk)$, which is significantly smaller than $O(nk)$ if the number of holes is small, but they are more difficult to compute

ray shooting
sentinel point
Dehn-Schönflies theorem

efficiently. Perhaps the simplest approach is to preprocess P for ***ray-shooting*** queries, which find the first segment (either edge or fence) hit by a ray shot from a given location in a given direction. After preprocessing P in $O(n \log n)$ time, any ray-shooting query can be answered in $O(\log n)$ time, using a data structure of Hershberger and Suri [18].[2] Computing a signed crossing sequence of length x requires exactly $k + x$ ray-shooting queries. Thus, the overall running time of this modified algorithm is $O(n \log n + hk \log n)$.

Theorem 3.20. *Let P be polygon with holes with n vertices, and let α and β be polygonal paths (or cycles) in P with k edges in total. We can determine whether α and β are (freely) homotopic in P in $O(n \log n + hk \log n)$ time.*

Sentinels

We can further simplify the algorithm by replacing each hole in P with a single ***sentinel point***. Finally, let $S = \{s_1, s_2, \ldots, s_h\}$ denote a set of h points, where each point s_i lies in the corresponding hole P_i. Because P is a subset of the space $\mathbb{R}^2 \setminus S$, any function from $[0,1]^2$ to P is also a function from $[0,1]^2$ to $\mathbb{R}^2 \setminus S$. It follows that any contractible loop in P is also contractible in $\mathbb{R}^2 \setminus S$. In fact, the converse is true as well.

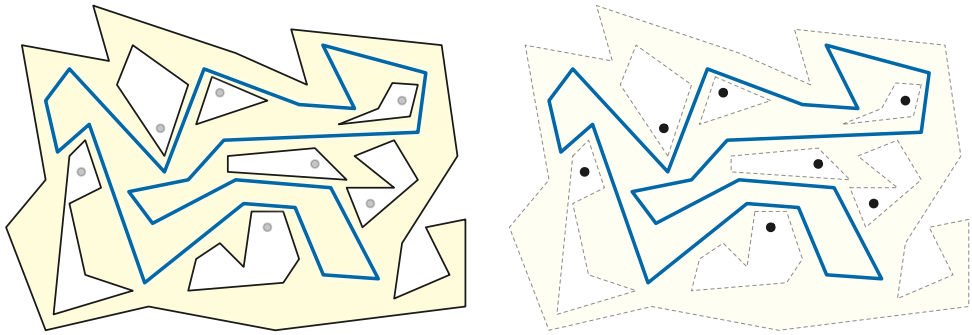
Lemma 3.21. *A loop in P is contractible in P if and only if it is contractible in $\mathbb{R}^2 \setminus S$.*

Proof: Let $P = P_0 \setminus (P_1 \cup \cdots P_h)$, where each P_i is a simple polygon, indexed so that each sentinel point s_i lies in the corresponding polygon P_i. The Dehn-Schönflies theorem implies that for each i, there is a homeomorphism $\phi_i : \mathbb{R}^2 \to \mathbb{R}^2$ whose restriction to the unit circle S^1 is the cycle ∂P_i. Without loss of generality, we can assume that $\phi_i(0) = s_i$, where 0 denotes the origin. Let $u : \mathbb{R}^2 \setminus 0 \to S^1$ be the function $u(x) = x/\|x\|$; this function is clearly continuous. Then the function $\Phi_i = \phi_i \circ u \circ \phi_i^{-1}$ maps $\mathbb{R}^2 \setminus s_i$ continuously onto P_i. Finally, let $r : \mathbb{R}^2 \setminus S \to P$ denote the function

$$r(x) = \begin{cases} x & \text{if } x \in P, \\ \Phi_0(x) & \text{if } x \text{ is outside } P_0, \\ \Phi_i(x) & \text{if } x \text{ is inside } P_i. \end{cases}$$

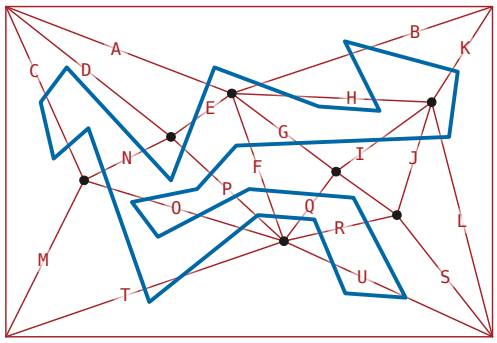
This function is continuous and obviously fixes P, so it is a retraction. (In fact, r is a *deformation* retraction.) The result now follows from Lemma 3.7. □

Thus, to test the contractibility of a loop in P, it suffices to work in the simpler space $\mathbb{R}^2 \setminus S$. We can construct a triangulation of $\mathbb{R}^2 \setminus S$—or more accurately, a triangulation of $R \setminus S$ for some sufficiently large rectangle R—in $O(h \log h)$ time. Any path or cycle composed of k segments crosses the edges of this triangulation at most $O(hk)$ times, so we can compute its reduced crossing sequence in $O(hk)$ time.

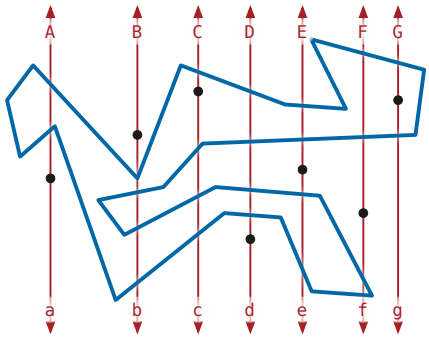


Replacing a polygon with holes with a set of sentinel points.

Theorem 3.22. *Let P be polygon with h holes and n vertices, and let α and β be polygonal paths (or cycles) in P with k edges in total. We can determine whether α and β are (freely) homotopic in P in $O(n + h \log h + hk)$ time.*



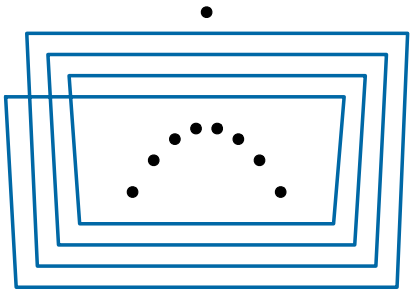
A triangulation of $R \setminus S$
and a loop with crossing sequence
CNPEABHHBBKJIGFPOOPFQRUURQFPOTTNC.



A partition of $\mathbb{R}^2 \setminus S$ into vertical slabs
and a loop with crossing sequence
AbcDeffeDcbbcDEFgGFEDCbA.

As a final practical simplification, we can subdivide the plane using vertical lines through the sentinel points instead of a triangulation. To avoid degenerate cases, we choose a coordinate system in which no two sentinel points have the same x-coordinate. After sorting the sentinels by their x-coordinates in $O(h \log h)$ time, we can easily compute the crossing sequence of any path with respect to the ray shooting up and down from the sentinel points in $O(k + x) = O(hk)$ time. Once again, Lemmas 3.14 and 3.15 generalize easily to this setting.

There are infinite families of paths and sentinel points that achieve the worst-case bound $x = \Theta(hk)$. For example, consider a path that spirals around a constant fraction of the sentinel points $\Theta(h)$ times, using a constant number of segments per turn. Unfortunately, it is not even possible to avoid this worst-case behavior by carefully choosing the triangulation or the vertical direction for slabs.[3]



A cycle with a long reduced crossing sequence, for *any* triangulation.

simple
rectification
in general position
trapezoidal decomposition
fence

3.4 Few Self-Intersections

The algorithm we just described can be improved even further for simple curves, or more generally, for curves that self-intersect only a small number of times. In this section we describe such an improvement, based on an algorithm of Cabello *et al.* [7], with some simplifications by Efrat *et al.* [14]. The key insight is that we can modify any cycle and any set of sentinel points, without changing the homotopy class of the cycle, but giving the cycle a much simpler combinatorial structure. Cabello *et al.* refer to this simplification process as ***rectification***.

Let $S = \{s_1, \ldots, s_h\}$ be a set of sentinel points in the plane, and let γ be a (not necessarily simple) polygonal cycle with k edges in $\mathbb{R}^2 \setminus S$. The algorithm proceeds in three stages:

1. Construct a *trapezoidal decomposition* of the cycle γ and points S.
2. Compute a *rectified* cycle $\overline{\gamma}$, composed of horizontal and vertical line segments, and a new set $\overline{S}$ of sentinels, which define the same crossing sequence as γ and S.
3. Simplify the rectified cycle as much as possible by sliding vertical edges.

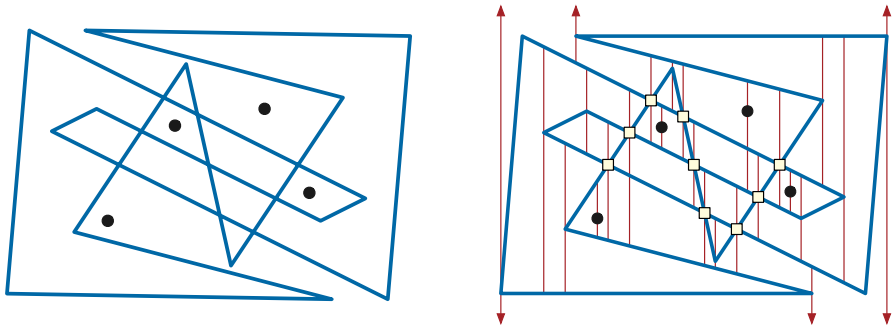
We describe and analyze each stage in more detail in the rest of this chapter. To simplify exposition, we assume that our input is in general position. Specifically, we assume that non-adjacent edges of γ intersect only in their interiors; at most two edges of γ intersect at any point; and all sentinel points, vertices, and self-intersection points have distinct x-coordinates. These genericity assumptions can be enforced using standard perturbation techniques [22]; alternatively, with some effort, the algorithm we describe can be generalized to deal with degenerate inputs directly (and more efficiently) [6].

Trapezoidal Decomposition

The first step in our algorithm is to construct a ***trapezoidal decomposition*** of the plane, defined by extending vertical segments called ***fences*** upward and downward from vertices of γ, self-intersection points of γ, and sentinel points in S, until they hit edges of γ. These fences and the edges of γ partition the plane into trapezoids, some of which are unbounded and others degenerate.

balanced binary search tree
priority queue

A classical algorithm of Bentley and Ottmann [2] constructs this decomposition in $O((k+s+h)\log k)$ time, where s is the number of self-intersection points. There are several faster algorithms, especially when the number of self-intersections is small [9, 11, 12, 15, 21], but the Bentley-Ottmann algorithm is much simpler to describe and implement, and using these faster alternatives would not significantly improve the overall running time of our contractibility algorithm. We give only a brief sketch of the Bentley-Ottmann algorithm here; more detailed descriptions can be found in most computational geometry textbooks.



A contractible cycle and four sentinel points, and their trapezoidal decomposition.
Squares mark self-intersection points.

Define trapezoidal decompositions and sketch Shamos-Hoey sweep-line algorithm (for polygons with holes / segments without intersections) in Chapter 1. Here we only need to discuss Bentley and Ottmann's changes to handle intersections.

Think of the k edges of γ and the h sentinel points as a set of $k+h$ line segments. The Bentley-Ottmann algorithm sweeps a vertical line L across the plane, maintaining the sequence of segments intersecting L, sorted by the y-coordinates of their intersection points with L. This intersection sequence changes whenever L encounters an endpoint of one of the segments or the intersection point of two segments; the algorithm maintains the sequence in a balanced binary search tree, so that segments can be inserted or deleted in $O(\log k)$ time. The positions where the intersection sequence changes are called *events*; the algorithm processes the events in order from left to right. The endpoints are all known in advance, so at any time it is easy to find the next vertex event. The next intersection event must be at the intersection point between two consecutive segments in the intersection sequence; these $O(k)$ consecutive pairs are maintained in a priority queue, so that the next intersection event can be found in $O(\log k)$ time. Finally, at each event, the algorithm inserts a fence through the appropriate point, whose top and bottom endpoints lie on the segments just above and below the event point. Altogether, handling each event requires a constant number of binary search tree operations, a constant number of priority queue operations, and a constant amount of other work; thus, each event can be processed in $O(\log k)$ time. There are $O(s+k+h)$ events, so

the total running time of the algorithm is $((s+k+h)\log k)$.

Once we have the trapezoidal decomposition, we insert a pair of coincident vertices into the cycle γ at each self-intersection point, increasing the number of edges from k to $k+2s$. This subdivision ensures that the edges of γ intersect *only* at vertices.[4]

vertical ranking
sigma above tau
sigma above2 tau
topological sort
horizontal ranking

Vertical and Horizontal Ranking

In the second stage of the algorithm, we replace γ with another cycle $\overline{\gamma}$ in the same homotopy class, composed entirely of horizontal and vertical line segments. At a high level, we replace each edge of γ with a horizontal line segment and each vertex of γ with a vertical line segment. The y-coordinates of the horizontal segments are determined by the following vertical ranking of the edges and sentinel points.

Let D be the set of interior-disjoint line segments consisting of the subdivided edges of γ and the sentinel points S (which we treat as zero-length segments). For any two segments σ and τ in D, write $\boldsymbol{\sigma \Uparrow \tau}$ if there is a vertical line segment with positive length whose top endpoint lies on σ and whose bottom endpoint lies on τ.

Lemma 3.23. *For any set D of interior-disjoint line segments, the relation $\Uparrow$ is acyclic.*

Proof: For the sake of deriving a contradiction, let $\sigma_1 \Uparrow \sigma_2 \Uparrow \cdots \Uparrow \sigma_r \Uparrow \sigma_1$ be a minimum-length cycle, where each σ_i is a segment in D. Obviously $r \geq 2$, because a segment cannot be just above itself. Similarly, $r \geq 3$, because $\sigma_1 \Uparrow \sigma_2$ and $\sigma_2 \Uparrow \sigma_1$ would imply that σ_1 and σ_2 have intersecting interiors.

Rotate the indices if necessary, so that the right endpoint of σ_1 has strictly smaller x-coordinate than the right endpoint of any other segment σ_i in the cycle. In particular, the right endpoints of σ_2 and σ_r are strictly to the right of σ_1. Thus, the right endpoint of σ_1 is both directly above some point of σ_2 and directly below some point of σ_r. It follows that $\sigma_r \Uparrow \sigma_2$, which contradicts the minimality of the cycle. □

Now write $\boldsymbol{\sigma \uparrow \tau}$ if some fence in the trapezoidal decomposition of D touches both σ and τ, and the fence point on σ is above the fence point on τ. The relation $\sigma \uparrow \tau$ trivially implies $\sigma \Uparrow \tau$; conversely, if $\sigma \Uparrow \tau$, we have $\sigma \uparrow \sigma_1 \uparrow \cdots \uparrow \sigma_r \uparrow \tau$ for some intermediate segments $\sigma_1, \ldots, \sigma_r$ in D. We can easily extract a directed acyclic graph representing the relation $\uparrow$ directly from the trapezoidal decomposition of D. To define the vertical ranking of D, we index the segments $D = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ using a standard topological sort, so that $\sigma_i \Uparrow \sigma_j$ implies $i < j$. Since we already have a trapezoidal decomposition of D, we can compute this vertical ranking in $O(k+s+h)$ time.

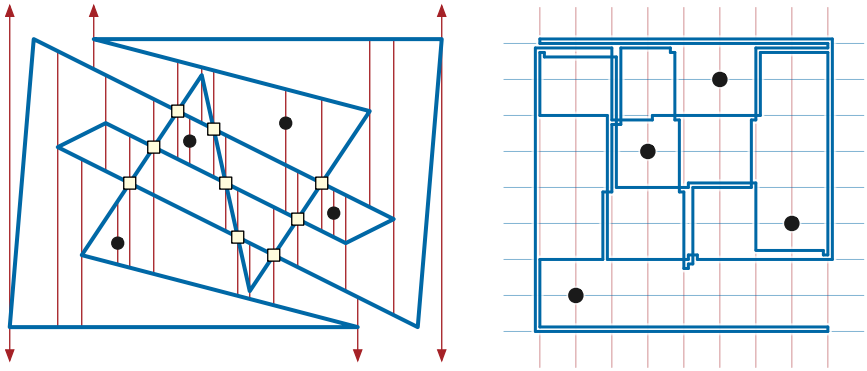
We also define a horizontal ranking of the vertices of D as the sorted order of their x-coordinates, which we computed when we constructed the trapezoidal decomposition.

rectification
reduction!of a rectified path
bracket in an orthogonal path
sliding brackets

In fact, only sentinel ranks matter. Within any horizontal slab between two sentinels, all horizontal segments can be squeezed onto a single horizontal line. Similarly, within any vertical slab between two sentinels, all vertical segments can be squeezed onto a single vertical line. Lots of zero-length edges, which can be removed. Lots of overlapping edges, which don't matter. Thus, in the rectified world, all cycle coordinates are odd integers between 1 and $2h+1$, and all sentinel coordinates are even integers between 2 and $2h$.

Rectification

Now we construct a ***rectified*** cycle $\overline{\gamma}$ and a new set $\overline{S}$ of sentinel points by replacing all x-coordinates with horizontal ranks and all y-coordinates with vertical ranks. Specifically, we replace each segment σ_i with a horizontal line segment $\overline{\sigma}_i$ whose y-coordinate is i and whose x-coordinates are the horizontal ranks of the endpoints of σ_i. For each pair of consecutive edges σ_i and σ_{i+1} of γ, we connect the corresponding endpoints of $\overline{\sigma}_i$ and $\overline{\sigma}_{i+1}$ with a vertical segment τ_i (dashed in the figures). Finally, for each sentinel point s_i, we define a new sentinel point $\overline{s}_i$ whose x-coordinate is the horizontal rank of s_i and whose y-coordinate is the vertical rank of s_i.

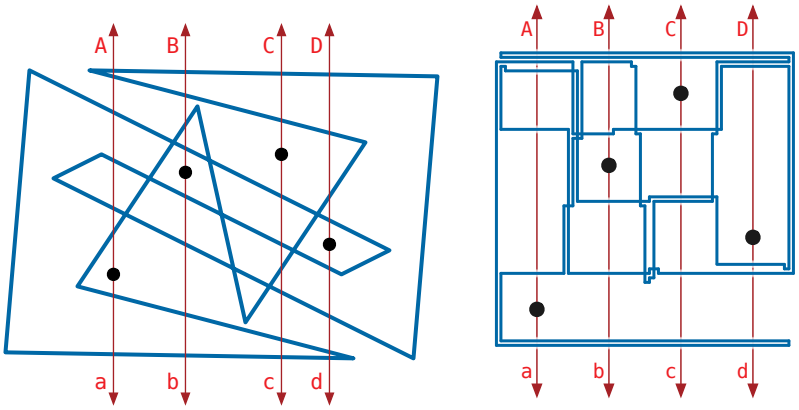


Rectified paths and points derived from a trapezoidal decomposition.
All vertices and edges actually lie on the grid.

If we define crossing sequences in terms of vertical rays shot up and down from each sentinel point, the cycles γ and $\overline{\gamma}$ have identical crossing sequences in their respective environments: $X_S(\gamma) = X_{\overline{S}}(\overline{\gamma})$. It follows that the original cycle γ is contractible in $\mathbb{R}^2 \setminus S$ if and only if the rectified cycle $\overline{\gamma}$ is contractible in $\mathbb{R}^2 \setminus \overline{S}$.

Sliding Brackets

Finally, we ***reduce*** the rectified path by sliding edges. We call an edge of an orthogonal polygon a ***bracket*** if both of its neighboring edges are on the same side of the line through the edge. To reduce the rectified cycle, we ***slide*** each bracket as far inward as



Both cycles have the (trivial) crossing sequence abcddcbaABcDDCBAABCDdcbAAbcdDcBA.

frozen bracket
spur
crossing sequence
elementary reduction
reduced crossing sequence

far as possible, shortening its adjacent edges. We stop sliding a bracket when it satisfies at least one of the following conditions:

- The bracket has distance 1 from a sentinel point inside the bracket; we call a bracket that satisfies this condition ***frozen***. Sliding a frozen bracket further would change the crossing sequence of the cycle, either by removing a *non-matching* pair of symbols, or by replacing one symbol with another.
- One of the edges adjacent to the bracket collapses to a point. This collapse either merges two parallel segments or collapses two parallel edges into a spur (which can be treated as a bracket with width zero). In either case, the number of edges decreases by at least 1.

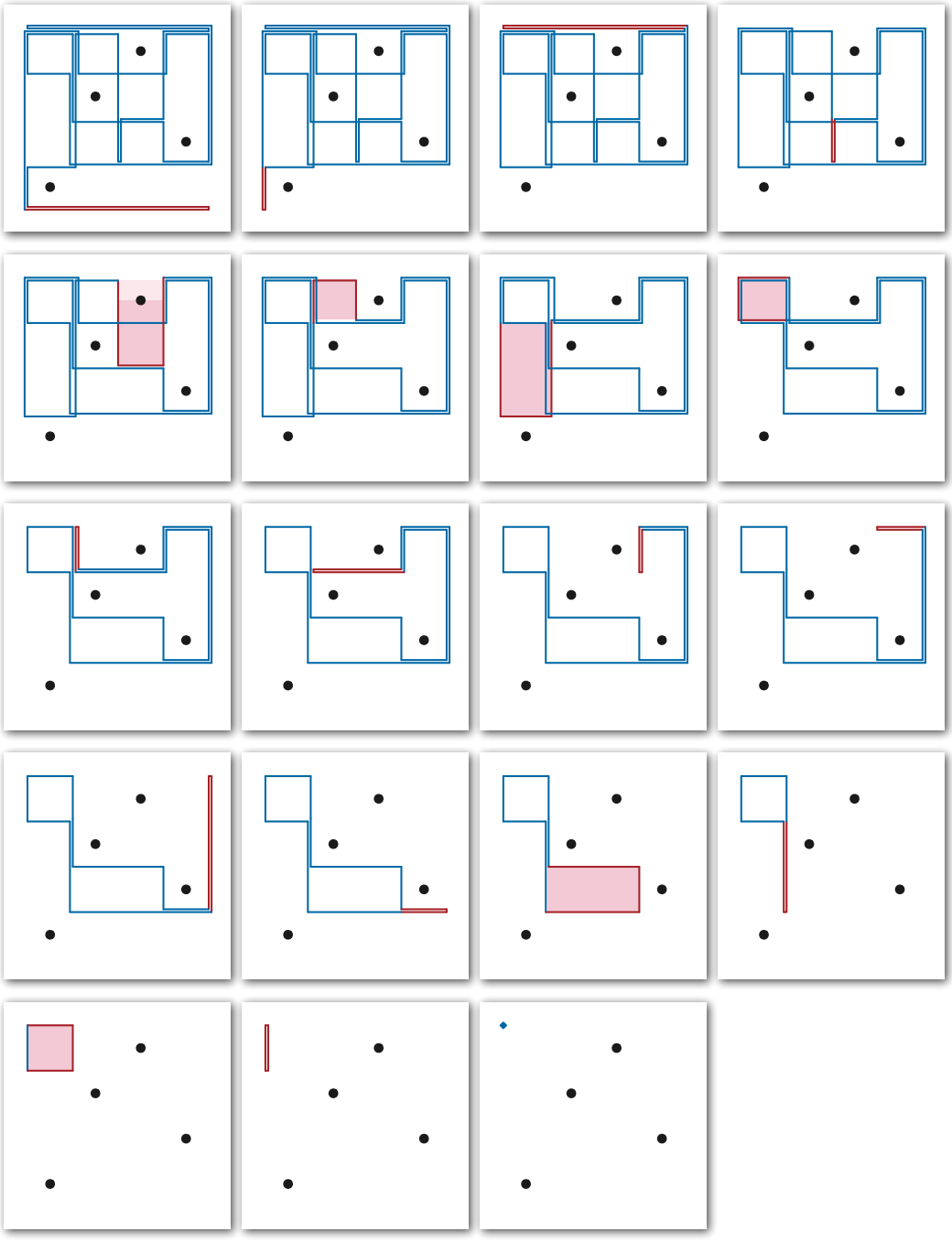
The algorithm ends when all remaining brackets are frozen. Since each bracket slide either freezes or removes a bracket, the algorithm ends after at most after $O(k+s)$ bracket slides. The figure on the next page shows a sequence of bracket slides contracting a rectified cycle.

The correctness of this algorithm rests on the following easy lemma.

Lemma 3.24. *A rectified cycle $\overline{\gamma}$ is contractible in $\mathbb{R}^2 \setminus \overline{S}$ if and only if it reduces to a constant cycle.*

Proof: Every bracket slide changes the crossing sequence of the rectified cycle by a (possibly empty) sequence of elementary reductions, and therefore preserves the homotopy class of the cycle. Moreover, if the current rectified cycle has a non-reduced crossing sequence, then it must also have at least one non-frozen bracket. Thus, when the algorithm ends, the crossing sequence of the current rectified cycle is reduced. We conclude that the crossing sequence of the reduced cycle $\overline{\gamma}'$ is equal to the reduced crossing sequence of the original rectified cycle $\overline{\gamma}$.

Suppose $\overline{\gamma}$ is contractible. Lemma 3.15 implies that $\overline{\gamma}'$ has an empty crossing sequence, and therefore lies between two vertical lines through sentinel points. It follows



A sequence of bracket slides contracting a rectified cycle.
All spurs are removed at the beginning; later spurs are removed as quickly as possible.
The fifth bracket slides is the only one that freezes a bracket.

layered range tree
active node

that $\overline{\gamma}'$ must lie entirely on the vertical line through the basepoint of $\overline{\gamma}$; otherwise, either the leftmost or rightmost vertical edge of γ would be an unfrozen bracket. But if the highest and lowest points of $\overline{\gamma}'$ were different, at least one of them would be the tip of a spike that would be removed by the reduction algorithm. We conclude that $\overline{\gamma}'$ is a constant cycle.

The other direction is straightforward; any sequence of bracket slides (and spur removals) *is* a homotopy. □

Layered Range Trees

To implement the bracket-slides efficiently, we preprocesses the rectified sentinel points $\overline{S}$ into a data structure that supports fast queries of the following form: Given a horizontal query segment σ, report the lowest sentinel point (if any) that lies directly above σ. Symmetric data structures can report the highest sentinel point below a horizontal segment, or the closest sentinel points to the left and right of a vertical segment.

Lemma 3.25. *Any set $\overline{S}$ of h points in the plane can be preprocessed in $O(h \log h)$ time into a data structure of size $O(h \log h)$, so that the lowest point above an arbitrary horizontal segment can be computed in $O(\log h)$ time.*

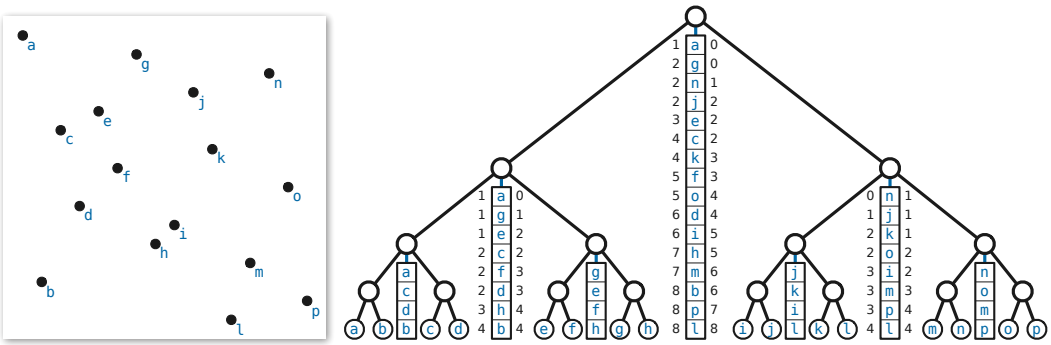
Proof: We use a data structure called a ***layered range tree***, which was proposed by Willard [24]. The layered range tree consists of a balanced binary search tree T over the x-coordinates of the sentinel points, with additional information stored at each node. To simplify queries, the x-coordinates are stored only at the leaves of T; the search keys for internal nodes are intermediate x-coordinates. For each node v of T, let l_v and r_v denote the smallest and largest x-coordinates in stored in the subtree rooted at v, and let $\overline{S}_v$ denote the subset of sentinel points with x-coordinates between l_v and r_v. Each node v in T stores the following information, in addition to the search key x_v.

- The x-coordinates l_v and r_v.
- The points $\overline{S}_v$, sorted by y-coordinate.
- For each point in $\overline{S}_v$, the number of points in $\overline{S}_{left(v)}$ with larger y-coordinate.
- For each point in $\overline{S}_v$, the number of points in $\overline{S}_{right(v)}$ with larger y-coordinate.

Given the h points $\overline{S}$, it is possible to construct a layered range tree for $\overline{S}$ in $O(h \log h)$ time; the total size of the data structure is $O(h \log h)$.

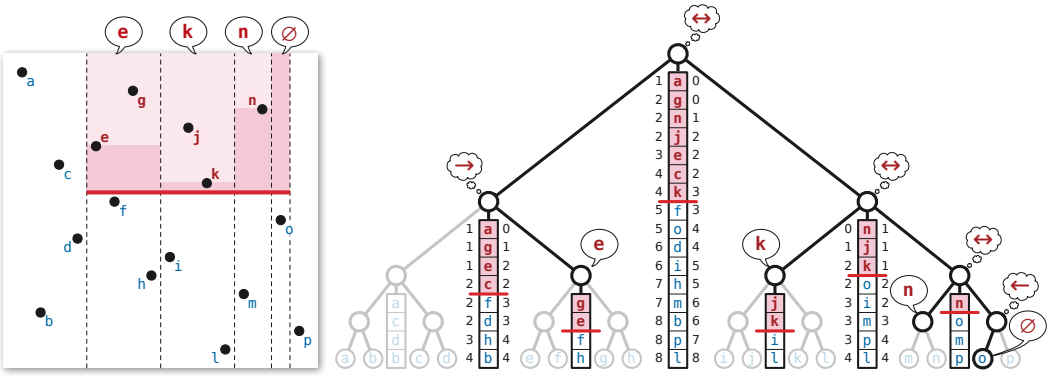
Now consider a query segment σ with endpoints (l, b) and (r, b) where $l < r$. We call a node v ***active*** for this segment if $l_v \le l$ and $r \le r_v$, but the parent of v is not active. There are at most two active nodes at each level of the tree, so there are $O(\log h)$ active nodes altogether, which we can easily find in $O(\log h)$ time by searching down from the root.

For any node v, let $lowest(v, b)$ denote the index of the lowest point in $\overline{S}_v$ that lies above the line $y = b$, or 0 if there is no such point. We can compute $lowest(root(T), b)$ in $O(\log h)$ time by binary search; for any other node v, we can compute $lowest(v, b)$



A layered range tree for 15 points (with some internal structure omitted).

in $O(1)$ time from $\mathit{lowest}(\mathit{parent}(v), b)$ and the left or right ranks stored at $\mathit{parent}(v)$. Thus, we can compute $\mathit{lowest}(v, b)$ for every active node v in $O(\log h)$ time. The answer to our query is the lowest of these $O(\log h)$ points. □



Answering a query in a layered range tree.

Final Analysis

We can easily choose a sentinel point inside each hole of P in $O(n)$ time; for example, we can use the lowest point on the boundary of each hole. Building a trapezoidal decomposition of the polygonal cycle γ and the sentinel points with the Bentley-Ottmann algorithm requires $O((h+k+s)\log k)$ time. Building the rectified cycle $\overline{\gamma}$ and new sentinel points $\overline{S}$ requires $O(h+k+s)$ time. Finally, reducing the rectified cycle requires $O(h \log h)$ time to build the layered range tree for $\overline{S}$, plus $O((k+s)\log h)$ time to perform the $O(k+s)$ bracket slides.

Theorem 3.26. *Let P be polygon with h holes with n vertices, and let γ be a polygonal cycle in P with k edges and s self-intersection points. We can determine whether γ is contractible in P in $O(n+(h+k+s)\log(h+k))$ time.*

This algorithm is faster than directly computing and reducing the crossing sequence (Theorem 3.22) when *either* the cycle has few self-intersections *or* the environment is more complex than the cycle. Ignoring logarithmic factors, the new running time is faster than $O(n + h \log h + hk)$ whenever $s = o(hk)$. Even in the worst case, when $s = \Theta(k^2)$, the running time of the new algorithm simplifies to $O(n + (h + k^2) \log(h + k))$, which is still faster than $O(n + h \log h + hk)$ when k is significantly smaller than h.

3.5 Exercises

1. Collected "exercises for the reader":
   a) Prove Lemma 3.4.
   b) Prove Lemma 3.5.
   c) Prove Lemma 3.6.
   d) Prove Lemma 3.12.
2.

crossing number

Notes

1. (page 12) Corollary 3.16 also follows easily from the observation that the circle S^1 is not simply connected. Despite its stunning obviousness, actually *proving* this fact from scratch requires essentially the same compactness argument that we used to prove Lemma 3.15.

2. (page 15)

Explain Hershberger-Suri ray-shooting data structure via geodesic triangulations?

3. (page 16) Cabello *et al.* [7] observed that we can further improve the worst-case length of crossing sequences using a suitable spanning tree of the sentinels, instead of vertical lines or triangulations. The crossing number of a spanning tree T is the maximum number of edges that can be crosses by a single line. Using a clear reweighting argument, Chazelle and Welzl [10, 23] proved that any set of h points in the plane has a spanning tree—in fact, a simple spanning *path*—whose crossing number is at most $O(\sqrt{h})$. (This bound is best possible; consider a $\sqrt{h} \times \sqrt{h}$ integer grid.) Moreover, algorithms of Edelsbrunner *et al.* [13], Agarwal and Sharir [1], and Matoušek [20] can be combined to compute such a spanning tree can be constructed in $O(h \log h)$ time. Adding a single infinite ray to this spanning tree gives us a set of h fences, which any path of k segments crosses at most $O(\sqrt{hk})$ times. Moreover, this crossing sequence can be computed in $O(\sqrt{hk} \log h)$ time using ray-shooting queries [18]. (In fact, Hershberger and Suri's ray-shooting data structure consists of a triangulation of the spanning tree with crossing number $O(\sqrt{h} \log h)$; a ray-shooting query is answered by walking through this triangulation in $O(1)$ time per triangle.)

Altogether, we obtain a homotopy-testing algorithm that runs in $O(n + h \log h + k\sqrt{h} \log h)$ time, improving Theorem 3.22 by a factor of $O(\sqrt{h}/\log h)$. However, this improved algorithm is unlikely to be efficient in practice unless h is extremely large; the algorithm for constructing the spanning tree is both complex and numerically fragile.

More advanced techniques Efrat *et al.* [14] and Bespamyatnikh [3, 4] can be used to test whether two paths are homotopic in $O(n + (h + h^{2/3}k^{2/3} + k)\,\text{polylog}(h + k))$ time. Except for the polylogarithmic factors, this running time is likely to be optimal [7, 16]. Again, these improvements are unlikely to be efficient in practice.

4. (page 19) The insightful reader may notice that if the input cycle γ is simple, we are essentially done at this point. Given the trapezoidal decomposition, it is straightforward to compute the subset of S that lies in the interior of γ in $O(h + k)$ additional time; γ is contractible if and only if this subset is empty.

I think the most annoying thing about secondary sources
is not that they contain errors or off-the-wall interpretations,
but that they NEVER include the specific information one is looking for.

— Craig Smoryński, *History of Mathematics: A Supplement* (2008)

Bibliography

[1] Pankaj K. Agarwal and Micha Sharir. Applications of a new space-partitioning technique. *Discrete Comput. Geom.* 9(1):11–38, 1993. (26)

[2] Jon Louis Bentley and Thomas A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.* C-28(9):643–647, 1979. (18)

[3] Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms* 49(2):284–303, 2003. (26)

[4] Sergei Bespamyatnikh. Encoding homotopy of paths in the plane. *Proc. LATIN 2004: Theoretical Infomatics*, 329–338, 2004. Lecture Notes Comput. Sci. 2976, Springer-Verlag. (26)

[5] Robert S. Boyer and J. Strother Moore. A fast string searhcing algorithm. *Commun. ACM* 20(10):762–772, 1977. (14)

[6] Christoph Burnikel, Kurt Mehlhorn, and Stefan Schirra. On degeneracy in geometric computations. *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, 16–23, 1994. (17)

[7] Sergio Cabello, Yuanxin Liu, Andrea Mantler, and Jack Snoeyink. Testing homotopy for paths in the plane. *Discrete Comput. Geom.* 31:61–81, 2004. (17, 26)

[8] Bernard Chazelle. An algorithm for segment-dragging and its implementation. *Algorithmica* 3:205–221, 1988. ()

[9] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.* 6(5):485–524, 1991. (18)

[10] Bernard Chazelle and Emo Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.* 4(1):467–489, 1989. (26)

[11] Kenneth L. Clarkson, Richard Cole, and Robert E. Tarjan. Erratum: Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.* 2(3):341–343, 1992. (18)

[12] Kenneth L. Clarkson, Richard Cole, and Robert E. Tarjan. Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.* 2(2):117–133, 1992. (18)

[13] Herbert Edelsbrunner, Leonidas Guibas, John Hershberger, Raimund Seidel, Micha Sharir, Jack Snoeyink, and Emo Welzl. Implicitly representing arrangements of lines or segments. *Discrete Comput. Geom.* 4(1):433–466, 1989. (26)

[14] Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Comput. Geom. Theory Appl.* 35(3):162–172, 2006. (17, 26)

[15] David Eppstein, Michael T. Goodrich, and Darren Strash. Linear-time algorithms for geometric graphs with sublinearly many edge crossings. *SIAM J. Comput.* 39(8):3814–3829, 2010. ArXiv:0812.0893. (18)

[16] Jeff Erickson. New lower bounds for Hopcroft's problem. *Discrete Comput. Geom.* 16:389–418, 1996. (26)

[17] Zvi Galil. On improving the worst case running time of the Boyer-Moore string matching algorithm. *Commun. ACM* 22(9):505–508, 1979. (14)

[18] John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms* 18:403–431, 1995. (15, 26)

[19] Donald Knuth, James H. Morris, Jr., and Vaughan Pratt. Fast pattern matching in strings. *SIAM J. Comput.* 6(2):323–350, 1977. (14)

[20] Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.* 10(1):157–182, 1993. (26)

[21] Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.* 1(1):51–64, 1991. (18)

[22] Raimund Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.* 19:1–17, 1998. (17)

[23] Emo Welzl. On spanning trees with low crossing numbers. *Data Structures and Efficient Algorithms: Final Report on the DFG Special Joint Initiative*, 233–249, 1992. Lecture Notes Comput. Sci. 594, Springer. (26)

[24] Dan E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.* 14(1):232–253, 1985. (23)