Topological Optimization Problems **Summer 2014**

Shortest path homotopic to a given path in polygons (June 10)

Speaker: Aruni Choudhary *Scribe: Cornelius Brand*

1 Problem statement

Let X be a topological space (here, think of X as $\mathbb{R}^2$ or some polygon endowed with the relative topology of $\mathbb{R}^2$ on the polygon). Then a continuous map $p: [0,1] \to X$ is called a *path*, $p(0)$ and $p(1)$ are the *endpoints* of p (specifically, x_0 is the *start point of* p). Given to paths p_1, p_2 with the same endpoints x_0 and x_1, a continuous mapping $H: [0,1] \times [0,1] \to X$ is called a *path-homotopy between* p_1 *and* p_2 if $H(t,0) = x_0$, $H(t,1) = x_1$ for all $t \in [0,1]$ and $H(0,\cdot) = p_1$ and $H(1,\cdot) = p_2$ Two paths p_1, p_2 are said to be *homotopic* if there is a path-homotopy between them, written $p_1 \sim p_2$. The relation $\sim$ forms an equivalence relation on the set of all paths in X.

A *polygon* is a closed subset of $\mathbb{R}^2$ that is bounded by a polygonal chain, i.e. a finite sequence of line segments forming a loop. A *polygon with holes* is a polygon P from which the union of the interiors of a finite number of polygons contained in P was removed. Every polygon is understood to be a polygon with holes.

Given some polygon of complexity n and a path π consisting of k line segments, the problem is now to find a shortest path that is homotopic to π.

2 The Algorithm

As stated, the input is a polygon P, say of complexity n, with holes and a path π, with starting point s and endpoint t. The described algorithm was given by Hershberger and Snoeyink [1].

2.1 High-level description

On a high level, the algorithm proceeds in five stages:

1. **Triangulate** P and obtain a triangulation Δ of P. While doing so, assign to each diagonal $\delta \in \Delta$ uniquely some label $\ell(\delta)$
2. **Record** the sequence S in which π passes through the diagonals of Δ
3. **Reduce** the crossing sequence S to the reduced sequence $\hat{S}$ as follows: While there is some consecutive occurrence of the same label, i.e., S has $\ell(\delta)\ell(\delta)$ as a substring for some $\delta \in \Delta$, remove this substring from S.
4. **Construct a sleeve** Σ from $\hat{S}$ as follows: Start with the triangle containing the start point of π and add it to Σ. Then, following the reduced sequence $\hat{S}$, glue copies of the triangles together at the diagonals corresponding to the diagonals in $\hat{S}$. It is important to use a *new copy* everytime one encounters a triangle, such that the resulting set is a topological disk (but not necessarily a polygon anymore)

5. **Compute a shortest path** in the sleeve between the endpoints of π, which is then the desired shortest path homotopic to π.

As for step 3, consider the string `AABBABABBACDDCA`. The corresponding reduced sequence is given through `ABA`. Write $u \sim_R v$ if two strings u and v have the same reduced sequence, and similarly, $p_1 \sim_R p_2$ if two paths have the same reduced sequence. One can show that $u \sim_R v$ forms an equivalence relation on the set of words.[1] Two paths in P are homotopic if and only if they have the same reduced sequence, that is, when Π denotes the set of paths in P, then $\Pi/\sim \; = \; \Pi/\sim_R$.

Steps 1 and 5 are non-trivial. Triangulating can be done with standard text book algorithms. We elaborate on step 5 in the following subsection.

2.2 Shortest paths in sleeves using funnels

In step 5 of the algorithm, we need to compute the shortest path between the endpoints of the sleeve. We do so using the notion of a *funnel*, as introduced by Lee and Preparata [2]. It is most convenient to define this structure by how it is computed.

2.2.1 Preprocessing

Given the sleeve Σ, we construct a funnel as follows: First, we remove all diagonals δ from Δ such that s and t lie on the same side of δ. Number the remaining diagonals $\delta_1, \ldots, \delta_k$ in the order they appear in the sleeve. Write d_i and d^i for start and end point of diagonal δ_i, respectively. A path from s to t will cross the diagonals in this order.

Also, by appropriately pruning the sleeve, we may assume that s and t are the end points of some diagonals.

2.2.2 The structure of the funnel

The funnel consists of three components: The point s is followed by a *path* (more precisely, the already constructed portion of the shortest path) of points, that at some point, called the *cusp*, splits up in two *funnel walls* (left and right, from the perspective of s), which are concave (from the perspective of Σ) polygonal chains, that is, they "turn left" and "turn right", respectively. The walls (which will be implemented using a double-ended queue) represent the shortest paths from the current cusp to the end of the respective funnel wall.

This relies on the following observation: For some diagonal δ_i, consider the shortest paths π_i and π^i from s to d_i and d^i in Σ, respectively. Then, π_i and π^i may have some segment, starting from s, in common, until at some point (i.e., the cusp), they split up and follow different, disjoint paths in Σ. These paths need to be concave, as otherwise, making them concave by dropping some vertices will only shorten the path, which is a contradiction to minimality.

2.2.3 Construction of the funnel

Initially, the path is empty, the cusp equals s and the left and right funnel walls contain the respective end points of δ_1. Now, the diagonals $\delta_1, \ldots, \delta_k$ are processed in this order. When some diagonal δ_j is added, one of d_j, d^j has already been added to one of the funnel walls, and the point p currently processed is

[1] Even more, if forms a *congruence relation*, i.e. it is compatible with concatenation

added to the respective opposite one. Now we want to retain the property that the funnel wall w (where w is left or right) containing p is the shortest path to p.

Observe how each point u in the funnel defines a wedge by extending the two edges incident with u in w in the direction of the funnel. Now, starting with the last point of w, we pop points from the funnel until the wedge defined by the point a to be popped next contains p . Then, p is pushed onto the funnel. If, during the process of popping, the cusp c is popped, the new cusp c is given by a (this means that the shortest path to p leads over the other funnel wall). After processing the last diagonal, the funnel will contain a shortest path to t, as desired.

2.3 Runtime analysis

Step 1 can be done in time $O(n \log n)$ (or even in randomized time $O(n \log^* n)$ using Seidel's algorithm). Step 2 is trivially done in time $O(x)$, where x is the length of the crossing sequence (which is a measure of the length of π). Step 3 can be done using a single stack: Process the symbols of S in turn. For each symbol s in S, if the top of the stack is distinct from s, push s onto the stack. If s equals the top of the stack, pop the top of the stack and throw away s. This gives the reduced sequence in time $O(x)$. Step 4 can also be implemented in time $O(x)$, for it takes time $O(1)$ for each of the x triangles to glue it to the already constructed part of the sleeve in an appropriate way. Computing the funnel in step 5 takes time $O(x)$, as each vertex in the sleeve is checked for updating the funnel only once, and there are x of these. This leads to time $O(x)$, because updating the funnel for each vertex takes amortized time $O(1)$. For, although in each step, there might be $O(x)$ vertices popped, in total, every vertex may be popped only once, giving $O(x)$ time for the funnel computation.

In total, this gives a running time of $O(n \log n + x)$.

3 Extensions

One might consider different measures for the path length. One such measure is the so-called *link-metric*, where the length of a path is not its euclidean length, but instead given by the number of line segments it contains. That is, the problem changes to computing a path homotopic to a given one that uses as few line segments as possible. One can show, however, that this is reducible to the case of euclidean shortest paths, by computing euclidean shortest paths and then, using a greedy method, minimizing the number of link segments in this path. This is elaborated in section 4 in the article by Hershberger and Snoeyink [1].

References

[1] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry*, 4(2):63 – 97, 1994.

[2] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.