Summary of "Shortest Non-Crossing Walks in the Plane" (Jeff Erickson and Amir Nayyeri)

Somnath Meignana Murthy

17 January 2013

Abstract

Let G be an n-vertex plane graph with non-negative edge weights, and let k terminal pairs be specified on h face boundaries. We present an algorithm to find k non-crossing walks in G of minimum total length that connect all terminal pairs, if any such walks exist, in $2^{O(h^2)} n \log k$ time. The computed walks may overlap but may not cross each other or themselves. Our algorithm generalizes a result of Takahashi, Suzuki, and Nishizeki for the special case $h \leq 2$. We also describe an algorithm for the corresponding geometric problem, where the terminal points lie on the boundary of h polygonal obstacles of total complexity n, again in $2^{O(h^2)} n$ time, generalizing an algorithm of Papadopoulos for the special case $h \leq 2$. In both settings, shortest non-crossing walks can have complexity exponential in h. We also describe algorithms to determine in O(n) time whether the terminal pairs can be connected by any non-crossing walks.

Problem Formulation

There are two different variants of the shortest non-crossing walks problem: The goal is to compute a set of non-crossing ST -walks in G of minimum total length, or to report correctly that no such walks exist

The input consists of h disjoint simple polygons $P_1, P_2, \ldots, P_h$ in the plane, called obstacles, together with two disjoint sets $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_k\}$ of points on the boundaries of the obstacles, called terminals. A set of ST -walks is a set of walks $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ in G, where each walk ω_i connects s_i and t_i.

Geometric formulation

We consider the obstacles P_i to be open sets and without loss of generality we assume that each terminal is a vertex of some obstacle; let n denote the number of obstacle vertices.

Combinatorial formulation

The input consists of an n-vertex plane graph G = (V, E); a weight function w : E → R+; a subset $H = \{ f_1 , f_2 , \ldots , f_h \}$ of faces of G, called obstacles. Each terminal has degree 1 and each walk ω_i is forbidden to visit terminals s_j or t_j except at its endpoints. When h = 1, shortest non-crossing ST -walks are actually shortest paths joining corresponding terminals. For $h \geq 2$, there are inputs for which shortest non-crossing ST -walks must be non-simple

Lemma

Let $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ be vertices of degree 1 in a plane graph G, and let C_π the combinatorial embedding of their connection graph. G contains a set of non-crossing ST -walks if and only if C_π is a planar embedding

For each j, the crossing sequence X (σ_j , Ω) contains no non-empty even substring.

Any string of length at least 2^k with at most k distinct characters has a non- empty even substring

No bigon in H_{ij} is empty

The total degree of the bad vertices of C^*_{ij} is at most 4h − 4

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ be a minimum- length set of non-crossing walks in G, such that each walk ω_i connects terminals s_i and t_i. For all i and j, walk ω_j traverses loop l_i exactly 2^{j-i-1} times.

Shortest non-crossing ST -walks in an n-vertex planar graph with k terminal pairs and h obstacles can be computed in O(hn log k) time, if for every index i, terminals s_i and t_i lie on the same obstacle.

Shortest non-crossing ST -walks in the complement of h polygonal obstacles with total complexity n can be computed in $h^{O(h)}.n$ time, if for every index i, terminals s_i and t_i lie on the same obstacle.

Theorem

Let $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ be vertices of degree 1 in a plane graph G with n vertices. We can decide whether G contains a set of non-crossing ST -walks in O(n) time

Let $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ be distinct terminal points on the boundary of h disjoint closed polygonal obstacles $P_1, P_2, \ldots, P_h$ of total complexity n in the plane. We can decide whether there is a set of non-crossing ST-walks in $R^2 \setminus (P_1 \cup P_2 \cup \cdots \cup P_k)$ in O(n) time

Each walk ω_i crosses each shortest path σj at most 2^{2h-2} times.

Shortest non-crossing ST -walks in an n- vertex planar graph with k terminal pairs and h obstacles can be computed in $2O(h^2)n \log k$ time and 2O(h).n space.

Shortest non-crossing ST-walks in complement of h polygonal obstacles with total complexity n can be computed in $2^{O(h^2)}.n$ time & $2^{O(h)}.n$ space

Crossing Bounds

Any walk in a set of shortest ST -walks crosses a shortest path at most 2^k times.

Upper Bound

In the geometric setting, minimizing the length of the walks also minimizes the number of crossings between walks ω_i and shortest paths σ_j, but the combinatorial setting is more subtle. The goal is each walk ω_i crosses each shortest path σ_j at most $2^{O(h)}$ times. A substring is a contiguous sequence of symbols within a string. We call a substring of X (σ_j, Ω) even if any symbol appears an even number of times; for example, ELESSL is an even substring of the word SENSELESSLY

Lower Bound

We weight the edges between v and every other vertex and a loop edge l_i at each vertex si by setting $w(l_i) := 2^{in}$ for each i, and $w(uv) = w(vw) = \infty$, and setting $w(e) = 0$ for every other edge e. We define α_1 to be the empty walk, and for each $i \geq 2$, we define

$$\alpha_i := rev(\alpha_{i-1}) \ (v, s_{i-1}) \cdot l_{i-1} \cdot (s_{i-1}, v) \cdot \alpha_{i-1}$$

where . denotes concatenation operator. Finally, for each i, we define $\omega_i^* := (s_i, v) \cdot \alpha_i \cdot (v, t_i)$. Each walk ω^*_j traverses loop l_i exactly 2^{j-i-1} times if $i < j$, and does not traverse ω^*_i at all if $i \geq j$. Each walk ω^*_j crosses the shortest path σ from u to w exactly 2^{j-1} times, thus σ is crossed 2^n-1 times altogether. Ω^* is unique min-length set of non-crossing walks connecting terminals in G

Spanning Walks

Obstacles and terminal pairs naturally define a connection graph C whose nodes and arcs correspond to the obstacles f_i and terminal pairs (s_j, t_j). Any minimum- length set of non-crossing ST-walks, every walk is tight; or else, at least one walks shorter without introducing any crossings.

Tight Spanning Walks

We compute a shortest walk with a given crossing sequence X_i as follows:

First glue together x copies of G✂Σ along the copies of the shortest paths that ω crosses, to obtain a planar graph $\hat{G}$ of complexity $O(x_i n)$. Then compute shortest path $\hat{\omega}_i$ in $\hat{G}$ between S_i in initial copy of G✂Σ and t_i in final copy of G✂Σ, using linear-time shortest path algorithm. Finally, project the path $\hat{\omega}_i$ back into G to obtain the walk ω_i.