Algorithms on Polygonal Embeddings of Graphs*

Leizhen Cai

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
E-mail:lcai@cs.cuhk.edu.hk

April 26, 1996

Abstract

This paper is concerned with graphs embedded in the plane. We introduce the notion of polygonal embeddings and present an efficient algorithm for recognizing a polygonal embedding of a 2-connected graph. We also give an efficient algorithm for computing a shortest crossing-free path between any two vertices in a polygonal embedding.

Keywords: *embedded graph, polygonal embedding, cross-free path.*

1 Introduction

An *embedding* of a graph $G = (V, E)$ is a mapping of the vertices of G onto different points in the plane and the edges of G onto line segments that preserves the incidence relation between edges and vertices. Therefore an edge uv will be mapped onto the line segment linking the corresponding points of u and v. Furthermore, the point corresponding to a vertex v will not lie on any line segment that does not correspond to an edge incident with v in G. Figure 1 gives three different embeddings of a path with five vertices. We refer to an embedding of a graph as an *embedded graph*. In this paper, G will always denote an embedded graph. For simplicity, we also use vertex and edge, respectively, to refer to their corresponding point and line segment in the plane.

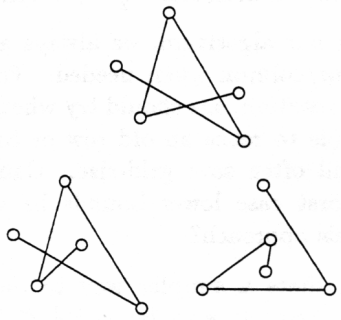

Figure 1: Three embeddings of a path with five vertices

Embedded graphs arise naturally when the geometric information of a graph is of concern. For example, consider the problem of triangulating a set S of points in the plane when the precise locations of points are unknown. We wish to find a triangulation T of S that is robust to inaccurate representation of the points in S in the sense that T remains to be a triangulation as far as each point in S is within distance ϵ from its actual location for some fixed number ϵ. This triangulation problem can be

*This work was partially supported by a Direct Grant for Research from the Chinese University of Hong Kong and an Earmarked Research Grant from the Research Grants Council of Hong Kong.

described as a problem on an embedded graph as follows: Construct an embedded graph G by taking each point in S as a vertex, and adding an edge between two points iff an 2ϵ-disc moving from one point to another does not collide with any other points in S. Then the triangulation problem is equivalent to the problem of finding a plane triangulation of G.

In this paper, we are interested in embeddings that have "nice" structures and efficient algorithms on such embeddings. For this purpose, we consider embeddings related to simple polygons and introduce the following notion of polygonal embeddings. An embedding G is *polygonal* if there is a polygon whose vertices are the vertices of the embedded graph such that no edge of G is in the exterior of the polygon; and such a polygon will be called the *host polygon* of the embedding. In Figure 1, the embedded graph on the top is polygonal but the two on the bottom are not polygonal. A polygonal embedding G also has a very close connection with the visibility graph G' of the host polygon P of G: it is an embedded subgraph of G' when we regard G' as an embedded graph.

The main results of this paper are an efficient algorithm for recognizing a polygonal embedding of a 2-connected graph, and an efficient algorithm for computing a shortest crossing-free path between any two vertices in a polygonal embedding.

2 Recognition of Polygonal Embeddings

In this section, we present an efficient algorithm for recognizing polygonal embeddings of 2-connected graphs. The complexity of the recognition problem for general graphs is unknown.

The main idea of the algorithm is to start with a lowest vertex (a vertex with the smallest y-coordinate) of G, and wrap around G counterclockwise. In the process, we wrap as close to G as possible. To guide the wrapping, we use the *vertex visibility graph* G' of G, which is defined to be the embedded graph on the vertices of G where two vertices u and v are joined by an edge e iff e does not intersect any other edges of G. The host polygon P is formed by growing a chain $p_1, p_2, \ldots$ of vertices. In the algorithm, p is the leading vertex in the chain and p' is the vertex next to p in the chain.

Algorithm *polygonal-embedding*

Input: An embedding G of a 2-connected graph.

Output: The host polygon $P = (p_1, \ldots, p_n)$ of G if G is polygonal.

Step 1. Construct the vertex visibility graph G' of G.

Step 2. Let p_1 be a lowest vertex in G, i.e., a vertex of G with the minimum y-coordinate.

Step 3. Pick an arbitrary point p' that lies to the left of and is lower than p_1. Set p to be p_1 and set $i = 2$.

Step 4. While p is unmarked do the following: find the edge pq in G that has the maximum angle clockwise from pp' and then locate the edge pq' in G' that has the smallest angle clockwise from pq (in case that there are several edges collinear with pq' then pq' is the shortest edge among these edges); set $p_i = q'$, $p' = p$, $p = q'$, mark p, and increase i by one.

Step 5. If P is a simple polygon that includes all vertices of G, then G is polygonal else G is not polygonal.

See Figure 2 for an example of the execution of the algorithm.

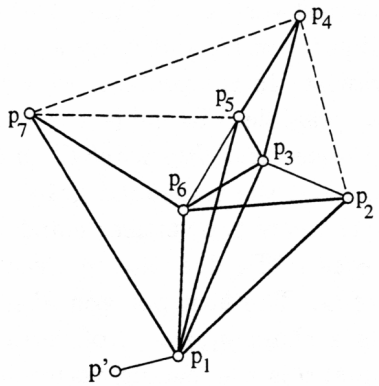

Figure 2: An execution of the algorithm *polygonal-embedding*. Dashed lines indicate edges in the vertex visibility graph G'.

We now sketch a proof for the correctness of the algorithm. It is clear that we only need to show that, whenever G is polygonal, P is the host polygon of G. For this purpose, we use induction to show that p_ip_{i+1} forms a side of the host polygon P of G. Consider p_ip_{i+1}. Suppose that it is not a side of P. Then there is a vertex r of G such that p_ir forms a side of P. Two cases arise depending on whether p_ir lies in between p_iq and p_ip_{i-1} (see Figure 3). In Case 1, there are two vertex-disjoint paths between r and p_{i-1} since G is 2-connected. This implies that there is a path Q between r and p_{i-1} that does not intersect p_ip_{i+1}. Then Q together with $p_{i-1}p_i$ and p_ir form a closed circle C whose interior is in the interior of P. However, vertex p_{i+1} is in the interior of C and thus in the interior of P, a contradiction. In Case 2, we can use a similar argument to deduce that r is in the interior of P, a contradiction. This establishes the correctness of the algorithm.

For the complexity of the algorithm, we note that the vertex visibility graph G' can be constructed in $O(n^2 \log n)$ time by fattening edges of G and applying a visibility graph algorithm of Ghosh and Mount [1]. The rest of the algorithm can be implemented in $O(n^2)$ time.

Theorem 2.1 *Given an embedding G of a 2-connected graph, it can be determined in time $O(n^2 \log n)$ whether G is a polygonal embedding.*

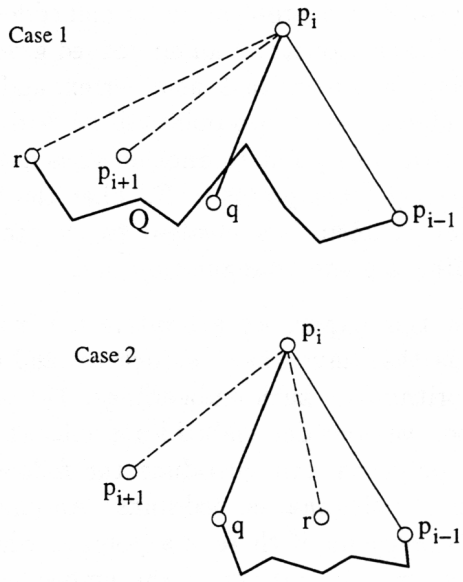

Figure 3: Two cases

We remark that the 2-connectivity of G is crucial for the algorithm to work. The algorithm may fail when G has cut vertices.

3 Shortest Crossing-Free Paths

In this section, each edge e of G is associated with a positive integer $w(e)$ as its weight. A path in G between two vertices is a *crossing-free path* if no edge in the path intersects any other edges and vertices in the path. A *shortest crossing-free path* between two vertices is a crossing-free path of the minimum weight. We consider the problem of finding a shortest crossing-free path in G between two given vertices.

In spite of extensive research on shortest path problems in graphs [5], to the best of my knowledge, the shortest crossing-free path problem has not been studied so far [2, 3, 4].

Algorithms for shortest path problems fail to work for the shortest crossing-free path problem. In fact, the complexity of the shortest crossing-free path problem is unknown for general embedded graphs; even the complexity of finding a crossing-free path in G between two vertices is unknown. In this section, we present a polynomial-time algorithm for finding a shortest crossing-free path between two vertices in a polygonal embedding.

An important property of a polygonal embedding is that the host polygon of the embedding gives an ordering of the vertices along the boundary of the polygon. This ordering of vertices allows us to explore the embedded graph in an orderly fashion by using a dynamic programming approach.

Let v_1 and v_k be the two vertices between which we wish to find a shortest crossing-free path in G. For convenience, we may assume that the vertices of G are $v_1, \ldots, v_k, u_{n-k}, \ldots, u_1$ counterclockwise along the boundary of the host polygon P of G. Let $U_i = \{u_1, \ldots, u_i\}$ and $V_j = \{v_1, \ldots, v_j\}$. For convenience, we regard $U_k = \emptyset$ for any $k \notin \{1, \ldots, i\}$. Let $G_{i,j}$ denote the embedded subgraph of G that is induced by $U_i \cup V_j$. Then $G_{0,j}$ is the embedded subgraph of G induced by V_j and $G = G_{n-k,k}$.

In order to construct a shortest crossing-free path between v_1 and v_k, we first define two matrices $D_u = (d_u(i,j))_{1..n-k,1..k-1}$ and $D_v = (d_v(i,j))_{0..n-k,1..k}$ to compute the length of a shortest crossing-free path between v_1 and v_k. Let $d_u(i,j)$, $1 \le i \le n-k$ and $1 \le j < k$, be the length of a shortest crossing-free path between v_1 and u_i in $G_{i,j}$; and let $d_v(i,j)$, $0 \le i \le n-k$ and $1 \le j \le k$, be the length of a shortest crossing-free path between v_1 and v_j in $G_{i,j}$. Then $d_v(n-k,k)$ is the length of a shortest crossing-free path in G between v_1 and v_k.

We will compute the two matrices D_u and D_v by a dynamic programming algorithm. Suppose that Q is a shortest crossing-free path in $G_{i,j}$ between v_1 and u_i. Then, obviously, Q contains an edge $u_i x$ for some vertex $x \in U_{i-1} \cup V_j$. If $x = v_{j'}$ for some $1 \le j' \le j$ then the $[v_1, v_{j'}]$-section of Q is a shortest crossing-free path between v_1 and $v_{j'}$ in $G_{i-1,j'}$. Otherwise, $x = u_{i'}$ for some $1 \le i' < i$, and the $[v_1, u_{i'}]$-section of Q is a shortest crossing-free path between v_1 and $u_{i'}$ in $G_{i',j}$. Therefore, we have the following recurrence relation for $d_u(i,j)$, where $N_G(x)$ denotes the set of vertices adjacent to x in G:

For each $1 \le i \le n-k$ and $1 \le j < k$, $d_u(i,j) = \min\{w(u_i, v_{j'}) + d_v(i-1, j'), w(u_i, u_{i'}) + d_u(i', j) : v_{j'} \in N_G(u_i) \cap V_j$ and $u_{i'} \in N_G(u_i) \cap U_i\}$.

Similarly, we have the following recurrence relation for $d_v(i,j)$:

For each $0 \le i \le n-k$, $d_v(i,1) = 0$; and for each $0 \le i \le n-k$ and $2 \le j \le k$, $d_v(i,j) = \min\{w(v_j, u_{i'}) + d_u(i', j-1), w(v_j, v_{j'}) + d_v(i, j') : u_{i'} \in N_G(v_j) \cap U_i$ and $v_{j'} \in N_G(v_j) \cap V_j\}$.

It is important to note that in the above recurrence relations, we assume that $\min \emptyset = \infty$. Now we can use these two recurrence relations to compute the length of a shortest crossing-free path in G between v_1 and v_k. Figure 4 indicates the entries in D_u and D_v that might be used in computing $d_u(i,j)$ and $d_v(i,j)$. From the figure, we see that we can fill matrices D_u and D_v alternatively in a row-by-row manner:

```
d_v(0,1) := 0;
for j := 2 to k do compute d_v(0,j);
for i := 1 to n - k do
begin for j := 1 to k - 1 do
        begin
            compute d_u(i,j);
            compute d_v(i,j); { d_v(i,1) := 0 }
        end;
        compute d_v(i,k);
end;
```

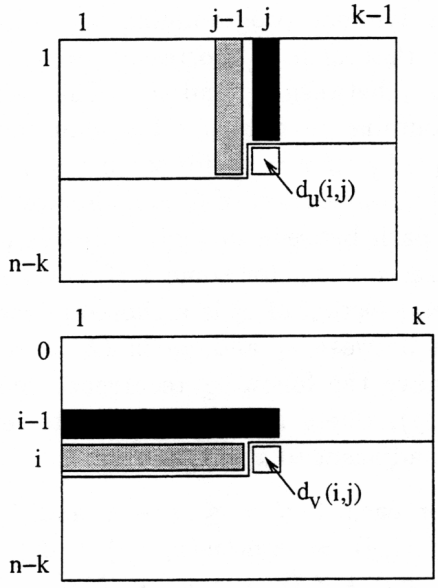

Figure 4: Dependence of entries in computing D_u and D_v. Black boxes indicate entries that might be used in computing $d_u(i,j)$, and shaded boxes indicate entries that might be used in computing $d_v(i,j)$.

To obtain a shortest crossing-free path in G between v_1 and v_k, we use $p(x)$ to record a vertex which is the current best candidate for the vertex that lies on a shortest crossing-free path from v_1 to x and immediately proceeds x. For each vertex u_i, whenever $d_u(i,j)$ is computed for some $1 \leq j < k$, $p(u_i)$ is updated to be a vertex in $U_{i-1} \cup V_j$ that produces the value $d_u(i,j)$. For each vertex v_j, $p(v_j)$ is updated in a similar fashion. Now a shortest crossing-free path in G from v_1 to v_k can be obtained by tracing back $p(v_k)$ recursively.

Figure 5 gives an example of the execution of the algorithm on a polygonal embedding.

It is clear that the algorithm runs in time $O(n^3)$, and we have the following theorem:

Theorem 3.1 *A shortest crossing-free path between any two vertices in a polygonal embedding of a weighted graph can be found in $O(n^3)$ time.*

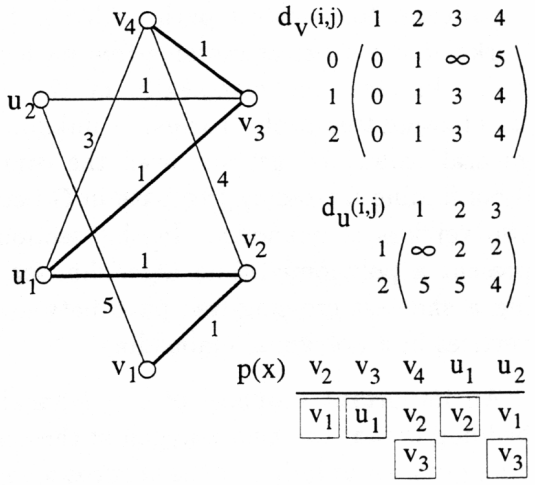

Figure 5: The execution of the shortest crossing-free path algorithm on a polygonal embedding. Thick edges indicate the resulting path. Vertices inside boxes are the final vertices stored in $p(x)$.

4 Concluding Remarks

In this paper, we have introduced the notion of polygonal embeddings of graphs, and presented an efficient algorithm for recognizing polygonal embeddings of 2-connected graphs. Furthermore, we have also given an efficient algorithm for finding a shortest crossing-free path between any two vertices in a polygonal embedding.

The subject of embedded graphs has not been well studied, and many interesting problems remain open. We list a few open problems pertaining to the material discussed in this paper.

1. What is the complexity of determining whether an embedding of a general graph is polygonal?

2. Define interesting and useful classes of embedded graphs.

3. What is the complexity of finding a

crossing-free path between two given vertices of a general embedded graph?

4. Design efficient algorithms for the single-source and the all-pair shortest crossing-free path problems on polygonal embeddings.

References

[1] S.K. Ghosh and D.M. Mount, "An Output-Sensitive Algorithm for Computing Visibility Graphs", *SIAM J. Comput.*, Vol. 20, No. 5, pp. 888-910, 1991.

[2] L. Guibas, private communication, 1995.

[3] H. Edelsbrunner, private communication, 1995.

[4] J. O'Rourke, private communication, 1995.

[5] R.E. Tarjan, "*Data Structures and Network Algorithms*", SIAM, 1983.