Discrete Mathematics and Theoretical Computer Science DMTCS vol. **17:1**, 2015, 1–12

A randomized algorithm for finding a maximum clique in the visibility graph of a simple polygon

Sergio Cabello[1*] Maria Saumell[2†]

[1]*Department of Mathematics, IMFM and FMF, University of Ljubljana, Slovenia*
[2]*Department of Mathematics and European Centre of Excellence NTIS, University of West Bohemia, Czech Republic*

received 18th Oct. 2013, revised 28th Nov. 2014, accepted 15th Dec. 2014.

We present a randomized algorithm to compute a clique of maximum size in the visibility graph G of the vertices of a simple polygon P. The input of the problem consists of the visibility graph G, a Hamiltonian cycle describing the boundary of P, and a parameter $\delta \in (0,1)$ controlling the probability of error of the algorithm. The algorithm does not require the coordinates of the vertices of P. With probability at least $1-\delta$ the algorithm runs in $O\left(\frac{|E(G)|^2}{\omega(G)}\log(1/\delta)\right)$ time and returns a maximum clique, where $\omega(G)$ is the number of vertices in a maximum clique in G. A deterministic variant of the algorithm takes $O(|E(G)|^2)$ time and always outputs a maximum size clique. This compares well to the best previous algorithm by Ghosh *et al.* (2007) for the problem, which is deterministic and runs in $O(|V(G)|^2\,|E(G)|)$ time.

Keywords: visibility graph, maximum clique, simple polygon, randomized algorithm

1 Introduction

A (simple) ***polygon*** is the region of the plane bounded by a non-self-intersecting, closed, polygonal path. The polygonal path itself is part of the polygon; it is usually called its ***boundary***. A polygon P defines a (vertex) ***visibility graph*** $G = G(P)$ in a natural way. The vertices of G are the vertices of the polygon. There is an edge between two vertices v and v' of G whenever the edge segment connecting v and v' is contained in P. In particular, the edges of the polygon correspond to edges of the visibility graph G. In fact, the edges of the polygon define a Hamiltonian cycle in G.

Several optimization problems have been considered for geometrically constrained graphs. In this paper, we are interested in finding a maximum clique in the visibility graph of a polygon. A ***clique*** is a complete subgraph. Thus, a clique in a graph H contains a subset of the vertices of H with the property

*Email: `sergio.cabello@fmf.uni-lj.si`. Supported by the Slovenian Research Agency, program P1-0297, projects J1-4106 and L7-5459, and within the EUROCORES Programme EUROGIGA (project GReGAS) of the European Science Foundation.

†Email: `saumell@kma.zcu.cz`. Supported by projects NEXLIZ - CZ.1.07/2.3.00/30.0038, which is co-financed by the European Social Fund and the state budget of the Czech Republic, and ESF EuroGIGA project ComPoSe as F.R.S.-FNRS - EUROGIGA NR 13604.

that each two vertices are connected by an edge. The ***clique number*** of H, usually denoted by $\omega(H)$, is the number of vertices in a maximum clique.

A clique in the visibility graph of a polygon has a simple, nice geometric interpretation. A subset of the vertices of P forms a clique whenever they are pairwise visible. It is possible to see that the vertices of a clique of the visibility graph G are in convex position. In fact, the vertices $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ of P, enumerated as they appear along the boundary of P, form a clique if and only if the polygonal path Q defined by $\overline{v_{i_1}v_{i_2}}, \overline{v_{i_2}v_{i_3}}, \dots, \overline{v_{i_{k-1}}v_{i_k}}, \overline{v_{i_k}v_{i_1}}$ forms a convex polygon contained in P. In particular, the edges of Q have to be edges of G.

The visibility graph G of a polygon P can be computed efficiently in $O(|E(G)|)$ time using the algorithm of Hershberger [Her89] together with linear-time triangulation [Cha91]. See the algorithm by Ghosh and Mount [GM91] for polygons with holes. However, the inverse direction is unclear: given a graph G, it is not known how to decide whether this is the visibility graph of a polygon. In particular, we cannot reconstruct efficiently a polygon whose visibility graph is given. See the discussion by Ghosh and Goswami [GG13, Gho07, Gho97].

We consider the following *restricted scenario*: the geometric coordinates of the polygon P are unknown. The information available as input is the visibility graph G and a Hamiltonian cycle C describing the boundary of P. As discussed before, with our current knowledge this information is strictly weaker than having the coordinates describing P. The objective is to find a maximum clique in G. The very same model and problem is considered by Ghosh, Shermer, Bhattacharya, and Goswami in [GSBG07] (see alternatively [Gho07, Section 6.7]), where an algorithm with time complexity $O(|V(G)|^2\,|E(G)|)$ is given. Using a modification of the ideas of Fisher [Fis97], Avis and Rappaport [AR85], or Bautista-Santiago *et al.* [BSDBL$^+$11], if the coordinates of the vertices are available we can obtain a better algorithm running in $O(|V(G)|\,|E(G)|)$ time.

For the aforementioned restricted scenario, we provide a deterministic algorithm finding a maximum clique in $O(|E(G)|^2)$ time, and a randomized algorithm finding a maximum clique in $O\left(\frac{|E(G)|^2}{\omega(G)}\log(1/\delta)\right)$ time with probability at least $1-\delta$. At a very high-level, our approach is an adaptation of the dynamic programming of Fisher [Fis97] and Bautista-Santiago *et al.* [BSDBL$^+$11] that infers enough information from the cycle C and the visibility graph G.

Our algorithms compare favorably with the result of Ghosh *et al.* [GSBG07]: The deterministic algorithm is faster than the algorithm in [GSBG07] when G is sparse, *i.e.* when $|E(G)| = o(|V(G)|^2)$; otherwise, the two algorithms have the same asymptotic running time. On the other hand, for constant values of δ the randomized algorithm is faster than the algorithm in [GSBG07] when $|E(G)| = o(|V(G)|^2)$, and when $|E(G)| = \Theta(|V(G)|^2)$ and $\omega(G)$ is super-constant. There exist situations where $|E(G)| = \Theta(|V(G)|^2)$ and the size of $\omega(G)$ is constant (see, for example, Figure 1), in which case the randomized algorithm does not give a speed-up.

We next introduce some notation. In Section 2 we provide geometric lemmas to extract certain geometric information from G and C. The description of the algorithm is in Section 3.

Notation. We assume that P has n vertices. Let $C = v_1v_2\dots v_nv_1$ be the Hamiltonian cycle describing the boundary of P. Throughout the paper, indices of vertices are treated modulo n, so that $v_{n+i} = v_i$. With a slight abuse of notation, we use v_i to refer to both the vertex of the polygon P and the corresponding vertex in the visibility graph G.

In our drawings we will assume that C gives the vertices in counterclockwise order; however, the algorithm is oblivious to the orientation.

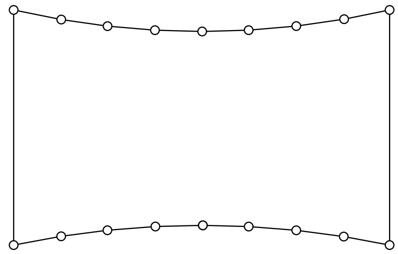

Figure 1: The visibility graph of this polygon has quadratic number of edges, while every maximum clique has only four vertices.

For any two vertices v_i and v_j of G, let $C(v_i, v_j)$ be the set of vertices in C as we walk from v_i to v_j along C, without v_i and v_j. For example, $C(v_i, v_{i+1}) = \emptyset$ and $C(v_i, v_{i+2}) = \{v_{i+1}\}$. Similarly as it is done for intervals, let $C(v_i, v_j] = C(v_i, v_j) \cup \{v_j\}$.

For each vertex v, $\deg(v)$ is the number of neighbours of v in G. It holds that $\sum_{v \in V(G)} \deg(v) = 2 \cdot |E(G)|$.

2 Geometric preliminaries

In this section we prove some geometric lemmas needed to argue the correctness of our algorithm. Some of these results are implicit in [GSBG07].

Lemma 1 *Let v_i be a vertex of G and let U_i be the set of vertices visible from v_i. The order of the vertices of U_i along C is the same as the (clockwise or counterclockwise) order of the edges $\{v_i u \mid u \in U_i\}$ around v_i.*

Proof: Assume that C gives the counterclockwise ordering along the boundary of P; the clockwise ordering is similar. Consider any two edges $v_i u$ and $v_i u'$ with $u, u' \in U_i$. Assume that $u' \in C(u, v_i)$ (see Figure 2). Then $v_i u'$ lies counterclockwise between $v_i u$ and $v_i v_{i-1}$. Indeed, the diagonal $v_i u$ cuts the polygon P into two polygons P' and P''. Let P' be the polygon defined by $C(u, v_i)$, v_i and u. The vertex u' belongs to P'. This means that $v_i u'$ is a diagonal of P' and it lies counterclockwise between $v_i u$ and $v_i v_{i-1}$. $\square$

Let K_t be a clique in G. A consequence of the previous lemma is that the order of the vertices of K_t along the convex hull of $V(K_t)$ is the same as the order along the boundary of P. Indeed, the circular ordering along the convex hull of $V(K_t)$ is the same as the circular ordering of the edges of $E(K_t)$ from any vertex of $V(K_t)$.

Let Q be a simple polygon, and let $u_1 u_2 \dots u_m u_1$ be the Hamiltonian cycle describing its boundary. We say that vertex u_i is ***convex*** if the interior angle of Q defined by edges $u_{i-1}u_i$ and $u_i u_{i+1}$ is convex. Notice that Q is a convex polygon if and only if all its vertices are convex.

The following lemma gives a tool to extend cliques. See Figure 3, left, for an illustration.

Lemma 2 *Let v_1, v_ℓ, v_i, v_n be distinct vertices of $V(G)$ with $v_\ell \in C(v_1, v_i)$. Let U be a subset of vertices from $C(v_1, v_\ell)$ such that $U \cup \{v_1, v_\ell, v_i, v_n\}$ is a clique in G. Possibly U is empty. Let $v_j \in C(v_i, v_n)$. If v_j sees v_1, v_ℓ, v_i, v_n, then $U \cup \{v_1, v_\ell, v_i, v_j, v_n\}$ is a clique in G.*

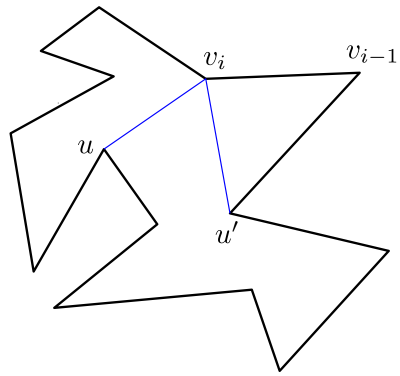

Figure 2: Since $u' \in C(u, v_i)$, $v_i u'$ lies counterclockwise between $v_i u$ and $v_i v_{i-1}$.

Proof: Consider the closed polygonal curve Q that follows the vertices of U, in the same order as they appear along $C(v_1, v_\ell)$, followed by $v_\ell, v_i, v_j, v_n, v_1$. Since all vertices of Q are visible from v_1 and have increasing indices, the path Q does not self-intersect. Thus Q is a polygon. Since Q is made of edges from $E(G)$, it is contained in P. All the vertices of Q, but those at v_i, v_j, v_n, are convex because $U \cup \{v_1, v_\ell, v_i, v_n\}$ is a clique. The vertices v_i, v_j, v_n of Q are convex because the edges $v_\ell v_j, v_i v_n, v_j v_1$ are in $E(G)$, respectively. $\square$

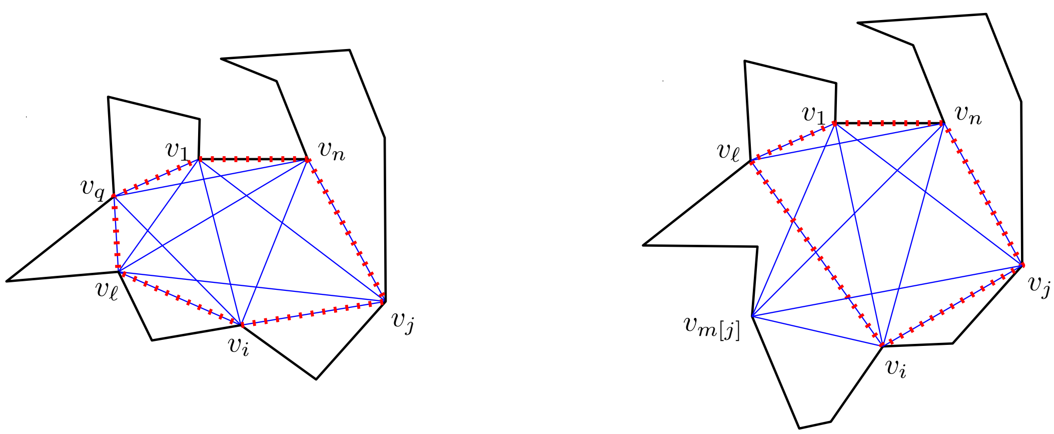

Figure 3: Illustrations for Lemmas 2 (left) and 3 (right). Solid edges belong to the visibility graph by hypothesis. Left: $U = \{v_q\}$ and the dotted polygonal curve shows Q. Right: The dotted polygonal curve shows Q_ℓ. Since $Q_{m[j]}$ is convex, so is Q_ℓ.

The following lemma shows that, in a certain configuration, the vertices visible from v_j have a very precise structure determined by a single index. See Figure 3, right, for an illustration.

Lemma 3 *Let $v_1v_iv_n$ be a triangle in G and let U be the set of vertices visible simultaneously from v_1, v_i and v_n. Let v_j be a vertex from $U \cap C(v_i, v_n)$. There exists some index $m[j] \in (1, i)$ such that $U \cap C(v_1, v_{m[j]}] = \{v_\ell \mid v_\ell \in U \cap C(v_1, v_i),\ v_\ell v_j \in E(G)\}$.*

Proof: Let $m[j] = \max\{\ell \in (1, i) \mid v_\ell \in U, v_\ell v_j \in E(G)\}$. Clearly,

$$U \cap C(v_1, v_{m[j]}] \ \supseteq \ \{v_\ell \mid v_\ell \in U \cap C(v_1, v_i),\ v_\ell v_j \in E(G)\}.$$

Consider any index $\ell \in (1, m[j]]$ such that $v_\ell \in U$. We want to argue that v_ℓ is visible from v_j. We argue this showing that the closed polygonal path $Q_\ell = v_1 v_\ell v_i v_j v_n v_1$ is a convex pentagon. Note that Q_ℓ cannot self-intersect: all vertices are visible from v_1 and thus radially sorted around v_1 because of Lemma 1. Thus Q_ℓ is a polygon. We have chosen $m[j]$ in such a way that $Q_{m[j]}$ is convex.

Since the edges v_1v_i, v_iv_n, v_jv_1, and v_nv_ℓ are in $E(G)$, Q_ℓ has a convex angle at all vertices, but possibly at v_i. Around v_i we have the following circular order of visible vertices because of Lemma 1: $v_j, v_n, v_1, v_\ell, v_{m[j]}$. Thus, the internal angle of Q_ℓ at v_i is smaller than the internal angle of $Q_{m[j]}$ at v_i, which was convex. It follows that all angles of Q_ℓ are convex and thus v_j sees v_ℓ. □

Lemma 4 *Consider the setting and notation in Lemma 3. Let v_j and v'_j be vertices from $U \cap C(v_i, v_n)$ with $j' > j$. Then $m[j'] \geq m[j]$.*

Proof: Let v_ℓ and $v_{\ell'}$ be vertices from $U \cap C(v_1, v_i)$ with $\ell' > \ell$. In the proof of Lemma 3 we have seen that, if v_j (or $v_{j'}$) sees $v_{\ell'}$, then it also sees v_ℓ (in that proof we had $\ell' = m[j]$, but the argument holds for any vertex in $U \cap C(v_1, v_i)$). By a symmetric argument, if v_ℓ (or $v_{\ell'}$) sees v_j, then it also sees $v_{j'}$. Therefore

$$\{v_\ell \in U \cap C(v_1, v_i) \mid v_\ell v_j \in E(G)\} \ \subseteq \ \{v_\ell \in U \cap C(v_1, v_i) \mid v_\ell v_{j'} \in E(G)\}$$

and the result follows from Lemma 3. □

3 The algorithm

Let K_t be a clique in G. An edge v_iv_j of K_t is ***lateral*** if all the vertices of $K_t - \{v_i, v_j\}$ are in $C(v_i, v_j)$ or in $C(v_j, v_i)$. Alternatively, v_iv_j is lateral for K_t if K_t is contained in any of the two polygons obtained by cutting P through the diagonal v_iv_j. In Section 3.1 we describe a subroutine LATERALMAXCLIQUE(G, C, e) to find a maximum clique that has e as lateral edge. This subroutine is deterministic.

The main idea of the algorithm for the general case is based on sampling several edges e of G, finding for each e a maximum clique in G that has e as lateral edge, and returning the largest of the cliques that are found. The precise details and analysis are given in Section 3.2.

Before we proceed, we give an algorithm to preprocess the representation of the graph. We assume that G is given with an arbitrary adjacency list representation. We want to have an adjacency list representation of G where, for each vertex v_i, the list ADJ$[i]$ has the (indices of the) neighbours of v_i in the same order as they appear along C, starting from v_{i+1} and finishing with v_{i-1}. As discussed in Lemma 1, the ordering along ADJ$[i]$ agrees with the clockwise or counterclockwise ordering of the edges emanating from v_i.

Lemma 5 *We can compute in $O\left(|E(G)|^2/\omega(G)\right)$ time all the lists ADJ$[i]$, $v_i \in V(G)$.*

Proof: For each vertex $v_i \in V(G)$, we collect the indices of the vertices adjacent to v_i, sort them in $O(\deg(v_i)\log(\deg(v_i))) = O(\deg(v_i)\log|V(G)|)$ time, and store them in $\text{ADJ}[i]$ sorted from $i+1$ to $i-1$. It remains to analyze the resulting time bound. Over all vertices, this takes time

$$O\left(\sum_i \deg(v_i)\log|V(G)|\right) \;=\; O(|E(G)|\log|V(G)|).$$

Given that we are aiming for an algorithm whose overall running time is roughly $O\left(|E(G)|^2/\omega(G)\right)$, it is enough for our purposes to show that $|E(G)|\log|V(G)| \le |E(G)|^2/\omega(G)$. Since $|E(G)| \ge \max\left\{|V(G)|, \omega^2(G)\right\}$, we have

$$\frac{|E(G)|}{\omega(G)} \;\ge\; \max\left\{\frac{|V(G)|}{\omega(G)}, \omega(G)\right\} \;\ge\; \sqrt{|V(G)|} \;\ge\; \log|V(G)|.$$

□

3.1 Maximum clique with a fixed lateral edge

Here we describe an algorithm $\text{LATERALMAXCLIQUE}(G, C, e)$ to find a maximum clique that has a given edge $e = v_iv_j$ as lateral edge. The running time of the algorithm is $O(|E(G)|)$.

If $e = v_iv_j$ does not belong to the cycle C, then we can decompose P along the diagonal e into two polygons P' and P'', search a largest clique containing e in each of the polygons P' and P'', and return the largest of both cliques. The visibility graphs of P' and P'' can be easily constructed in $O(|E(G)|)$: one is the subgraph of G induced by $C(v_i, v_j) \cup \{v_i, v_j\}$ and the other is induced by $C(v_j, v_i) \cup \{v_i, v_j\}$. Similarly, the cycle defining the boundary of both polygons P' and P'' is easily obtained from C. Therefore, it suffices to consider the case when e belongs to the cycle C.

Our algorithm goes along the lines of a dynamic-programming algorithm by Fischer [Fis97] to solve the following problem: Given a set of points in the plane labelled either "positive" or "negative" and a positive point p, find a convex polygon Q of maximum area such that: (i) p is the bottom vertex of Q; (ii) the vertices of Q are positive points; (iii) Q does not contain any negative point. The algorithm of Fisher can easily be adapted to optimize other criteria, instead of the area, such as the number of positive points in the boundary of Q. We recommend the presentation by Bautista-Santiago *et al.* [BSDBL+11], where the running time of Fischer is slightly improved.

We have to make two adaptations of the idea of Fisher to our setting. First, we do not have the coordinates of the vertices so we cannot deduce which vertices are above or below some given vertex. We work around this issue by using e as a reference: the vertices that are visible from both vertices of e lie in one halfplane defined by the line supporting e. Second, we adapt the dynamic programming to achieve a running time of $|E(G)|$. We next explain in detail the algorithm.

Without loss of generality we will assume henceforth that $e = v_1v_n$.

We first select the (indices of the) vertices of P that are visible from both v_1 and v_n. Let $J = \{i \in (1, n) \mid v_i \text{ visible from } v_1 \text{ and } v_n\}$. The set J can be constructed from $\text{ADJ}[1]$ and $\text{ADJ}[n]$ in time $O(|V(G)|) = O(|E(G)|)$.

For each $i \in J$ let us define

$$J_{<i} = \{j \in J \mid j < i,\, v_jv_i \in E(G)\} \quad\text{and}\quad J_{>i} = \{j \in J \mid j > i,\, v_iv_j \in E(G)\}.$$

See Figure 4, left. Note that, for example, $J_{<i}$ uses information about the visibility from v_i. Thus, $J_{<i} \cup J_{>i}$ is potentially smaller than J. For the algorithm we will need that $J_{<i}$ and $J_{>i}$ are represented as lists storing the elements sorted in increasing order. We can construct the lists $J_{<i}$, for all $i \in J$, in $O(|E(G)|)$ time, as follows. We first make a binary table $B[1..n]$ such that $B[j]$ is true if and only if $j \in J$. After this, we can construct $J_{<i}$ in time $O(\deg(v_i))$ by scanning $\textsc{Adj}[i]$: if $j \in \textsc{Adj}[i]$ satisfies $j < i$ and $B[j]$ is true, then we add j to $J_{<i}$. A similar approach works to construct in time $O(|E(G)|)$ the lists $J_{>i}$, for all $i \in J$. Since the elements of $\textsc{Adj}[i]$ are sorted, we also obtain the lists $J_{<i}$ and $J_{>i}$ sorted, as desired.

If J contains one single vertex or $J_{<i}$ is empty for all $i \in J$, then the largest clique containing e as lateral edge has size 3. We assume henceforth that J has more than an element and there are some edges $v_i v_j$ with $i, j \in J$.

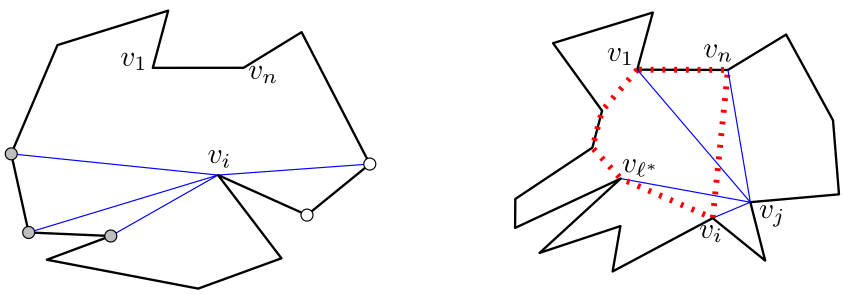

Figure 4: Left: Vertices filled in gray belong to $J_{<i}$, while vertices filled in white belong to $J_{>i}$. Right: Vertex v_j can be added to the clique associated to $\textsc{Opt}[\ell^*, i]$ (whose lateral edges are dotted) because it is visible from v_{ℓ^*}, v_i, v_1, and v_n.

For each $i \in J$ and $j \in J_{>i}$, we define $\textsc{Opt}[i, j]$ as the number of vertices in the maximum clique that has v_i, v_j, v_n and v_1 as consecutive vertices in the convex hull. Alternatively, $\textsc{Opt}[i, j]$ is the number of vertices in the maximum clique using vertices $\{v_i, v_j, v_n, v_1\}$ and a subset of $C(v_1, v_i)$. The algorithm finds the values $\textsc{Opt}[i, \cdot]$, $i \in J$, for increasing values of i, as follows. The following statement shows the recursive behavior of $\textsc{Opt}[\cdot, \cdot]$.

Lemma 6 *If $i \in J$ and $j \in J_{>i}$, then*

$$\textsc{Opt}[i, j] = \begin{cases} 4 & \text{if } \nexists \ell \in J_{<i},\ v_\ell v_j \in E(G), \\ 1 + \max\{\textsc{Opt}[\ell, i] \mid \ell \in J_{<i},\ v_\ell v_j \in E(G)\} & \text{otherwise.} \end{cases} \tag{1}$$

Proof: This is a standard proof in dynamic programming. We use induction on i. Consider $i \in J$ and $j \in J_{>i}$.

If $\{\ell \in J_{<i} \mid v_\ell v_j \in E(G)\}$ is empty, then no vertex in $C(v_1, v_i)$ is visible simultaneously from v_1, v_n, v_i, v_j, and thus $\textsc{Opt}[i, j] = 4$, as the lemma says. This covers all base cases.

Consider the case where $\{\ell \in J_{<i} \mid v_\ell v_j \in E(G)\}$ is nonempty. We have to show that $\textsc{Opt}[i, j] = 1 + \max\{\textsc{Opt}[\ell, i] \mid \ell \in J_{<i},\ v_\ell v_j \in E(G)\}$. We do this in two steps.

Let $U \subseteq C(v_1, v_i)$ be such that $U \cup \{v_i, v_j, v_n, v_1\}$ is a maximum clique defining $\textsc{Opt}[i,j]$. Let v_t be the vertex with highest index in U and set $U' = U \setminus \{v_t\}$. Clearly $t \in J_{<i}$ and $v_t v_j \in E(G)$. Since $U' \cup \{v_t, v_i, v_n, v_1\}$ is a clique with $U' \subset C(v_1, v_t)$, we have by induction that $\textsc{Opt}[t,i] \geq |U'| + 4 = |U| + 3$. Therefore

$$\textsc{Opt}[i,j] \;=\; |U| + 4 \;\leq\; 1 + \textsc{Opt}[t,i] \;\leq\; 1 + \max\{\textsc{Opt}[\ell, i] \mid \ell \in J_{<i},\, v_\ell v_j \in E(G)\}. \tag{2}$$

To show the other inequality, consider $\ell^* \in J_{<i}$ such that $v_{\ell^*} v_j \in E(G)$ and

$$\textsc{Opt}[\ell^*, i] \;=\; \max\{\textsc{Opt}[\ell, i] \mid \ell \in J_{<i},\, v_\ell v_j \in E(G)\}.$$

See Figure 4, right. By induction hypothesis, there exists $U \subseteq C(1, \ell^*)$ such that $U \cup \{v_{\ell^*}, v_i, v_n, v_1\}$ is a clique of size $\textsc{Opt}[\ell^*, i]$. Since $v_{\ell^*} v_j, v_i v_j, v_1 v_j, v_n v_j \in E(G)$, it follows that $U \cup \{v_{\ell^*}, v_i, v_j, v_n, v_1\}$ is a clique because of Lemma 2. Thus

$$\textsc{Opt}[i,j] \;\geq\; 1 + \textsc{Opt}[\ell^*, i] \;=\; 1 + \max\{\textsc{Opt}[\ell, i] \mid \ell \in J_{<i},\, v_\ell v_j \in E(G)\}. \tag{3}$$

Combining Equations (2) and (3), the remaining case is done. □

Using Equation (1), each value $\textsc{Opt}[i,j]$ can be computed in $O(\deg(v_i))$ time. This means that all $\textsc{Opt}[i,j]$, $i \in J$ and $j \in J_{>i}$ can be computed in $O((\deg(v_i))^2)$ time. However, this can be done faster by using a few additional observations, as we will show next.

Lemma 7 *Assume that the values $\textsc{Opt}[\ell, i]$ are already available for each $\ell \in J_{<i}$. Then we can compute the values $\textsc{Opt}[i,j]$ for all $j \in J_{>i}$ in $O(\deg(v_i))$ time.*

Proof: We obtain the values $\textsc{Opt}[i, \cdot]$ in increasing order of the second term. We use the following properties; see Lemmas 3 and 4:

- For each $j \in J_{>i}$, there is a value $m[j] \in J_{<i}$ such that

$$J_{<i} \cap (1, m[j]] \;=\; \{\ell \in J_{<i} \mid v_\ell v_j \in E(G)\}.$$

- If $j, j' \in J_{>i}$ and $j < j'$, then $m[j] \leq m[j']$. In other words, $m[j]$ is nondecreasing in $j \in J_{>i}$.

For each $\ell \in J_{<i}$, let

$$B[\ell] \;=\; \max\{\textsc{Opt}[\ell', i] \mid \ell' \in J_{<i}, \ell' \leq \ell\}.$$

We compute the values $B[\ell]$, $\ell \in J_{<i}$, in $O(\deg(v_i))$ by scanning $\textsc{Opt}[\cdot, i]$ and keeping the prefix maximum.

Next, we compute the index $m[j]$ for all $j \in J_{>i}$ in $O(\deg(v_i))$ time. This is a simple loop with two indices; details are provided in Figure 5. Here it is useful to treat $J_{<i}$ and $J_{>i}$ as lists where the elements are stored in increasing order. Finally, we use that, for each $j \in J_{>i}$

$$\textsc{Opt}[i,j] \;=\; \begin{cases} 4 & \text{if } m[j] \text{ is undefined,} \\ 1 + B[m[j]] & \text{if } m[j] \text{ is defined.} \end{cases}$$

Since the tables $B[\cdot]$ and $m[\cdot]$ can be computed in $O(\deg(v_i))$ time, the result follows. □

```
Algorithm FINDING m[j]
(* We treat J_{<i} and J_{>i} as lists storing the indices in increasing order *)
j ← first element in J_{>i};
ℓ ← first element in J_{<i};
while j ≠ NULL and ℓ ≠ NULL
        if v_ℓ v_j ∈ E(G) then
            m[j] ← ℓ;
            ℓ ← element after ℓ in J_{<i};
        else
            j ← element after j in J_{>i};
            m[j] ← element before ℓ in J_{<i};
```

Figure 5: Algorithm used in the proof of Lemma 7.

When the entire table $\text{OPT}[\cdot,\cdot]$ has been filled, we traverse it to find its maximum value. This maximum value is precisely $\text{LATERALMAXCLIQUE}(G, C, e)$. Notice that we can also find a largest clique containing e by augmenting each entry of the table OPT with extra information: for a given position $[i, j]$ we also store the value t such that $\text{OPT}[i, j] = \text{OPT}[t, i] + 1$. We conclude the following.

Lemma 8 *There is an algorithm* $\text{LATERALMAXCLIQUE}(G, C, e)$ *that, after* $O(|E(G)|^2/\omega(G))$ *preprocessing time, has the following property: for any given* $e \in E(G)$ *we can find in time* $O(|E(G)|)$ *a maximum-size clique in* G *that has* e *as lateral edge.*

Proof: We preprocess the graph as discussed in Lemma 5. After this, we compute the entries $\text{OPT}[i, j]$, $j \in J_{>i}$, for increasing values of $i \in J$. Using Lemma 7, we spend $O(\deg(v_i))$ per $i \in J$. So the total time to compute all entries $\text{OPT}[\cdot,\cdot]$ is $O\left(\sum_{i\in J} \deg(v_i)\right) = O(|E(G)|)$. Finally, we return $\max\{\text{OPT}[i, j] \mid i \in J,\ j \in J_{>i}\}$ and a clique of this size, which can be reconstructed from the extra information stored at each entry of the table OPT. Correctness follows from the definition of $\text{OPT}[\cdot,\cdot]$. □

3.2 General case

Calling $\text{LATERALMAXCLIQUE}(G, C, e)$ for each edge $e \in E(G)$ and returning the largest clique that is found, we obtain the following result.

Theorem 9 *Given the visibility graph* G *of a simple polygon* P *and a Hamiltonian cycle describing the order of the vertices of* P *along its boundary, we can compute a maximum clique in time* $O(|E(G)|^2)$. □

We now describe a randomized variant. In Figure 6 we give the algorithm RANDMAXCLIQUE.

Theorem 10 *Given the visibility graph* G *of a simple polygon* P*, a Hamiltonian cycle describing the order of the vertices of* P *along its boundary, and a parameter* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$*,* RANDMAXCLIQUE *computes a clique in* G *of maximum size in time* $O(\frac{|E(G)|^2}{\omega(G)} \log(1/\delta))$.

Proof: Since $\hat{\omega}$ is smaller than or equal to $\omega(G)$, the condition to exit the repeat loop implies that there are at least

$$m(G, \delta) := \left\lceil \frac{|E(G)|}{\omega(G)} \ln(1/\delta) \right\rceil$$

Algorithm RANDMAXCLIQUE
Input: graph G, Hamiltonian cycle C, parameter $\delta \in (0,1)$
Output: maximum clique in G with probability at least $1-\delta$

```
1.  Preprocess G to solve LATERALMAXCLIQUE;   (* Lemma 8 *)
2.  ω̂ ← 0;   (* size largest found clique *)
3.  i ← 0;   (* counter number of iterations *)
4.  repeat
5.      choose edge e ∈ E(G) uniformly at random;
6.      ω̂ ← max{ω̂, LATERALMAXCLIQUE(G, C, e)}
7.      i ← i + 1;
8.  until ω̂ ≥ (|E(G)|/i) ln(1/δ)
9.  return ω̂
```

Figure 6: Algorithm RANDMAXCLIQUE.

iterations of the repeat loop.

Let A be the event that, in the first $m(G,\delta)$ iterations of the repeat loop, at least once we select an edge e that is lateral for a maximum-size clique. If the event A occurs, then:

- the algorithm returns $\omega(G)$, and
- there are exactly $m(G,\delta)$ iterations of the repeat loop, so the time complexity is

$$O\left(\frac{|E(G)|^2}{\omega(G)} + m(G,\delta)\cdot|E(G)|\right) = O\left(\frac{|E(G)|^2}{\omega(G)}\log(1/\delta)\right)$$

  because of Lemma 8.

We next show that A occurs with probability at least $1-\delta$, which finishes the proof. Consider any iteration of the repeat loop. Since a maximum-size clique has $\omega(G)$ lateral edges, with probability at least $\frac{\omega(G)}{|E(G)|}$ the selected edge e is lateral for some maximum-size clique. Let A^c be the complement of A: in each of the first $m(G,\delta)$ iterations, e is not a lateral edge for any maximum-size clique. The probability of A^c is at most

$$\left(1-\frac{\omega(G)}{|E(G)|}\right)^{m(G,\delta)} \leq \left(1-\frac{\omega(G)}{|E(G)|}\right)^{\frac{|E(G)|}{\omega(G)}\ln(1/\delta)} \leq e^{-\ln(1/\delta)} = \delta,$$

where we have used that $(1-1/x)^x \leq 1/e$ holds for any $x \in \mathbb{R}$. Thus, the event A occurs with probability at least $1-\delta$ and the proof is finished. $\square$

References

[AR85] David Avis and David Rappaport. Computing the largest empty convex subset of a set of points. In *Proc. 1st Symp. Computational Geometry*, pages 161–167. ACM, 1985.

[BSDBL+11] Crevel Bautista-Santiago, José Miguel Díaz-Báñez, Dolores Lara, Pablo Pérez-Lantero, Jorge Urrutia, and Inmaculada Ventura. Computing optimal islands. *Oper. Res. Lett.*, 39(4):246–251, 2011.

[Cha91] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6:485–524, 1991.

[Fis97] Paul Fischer. Sequential and parallel algorithms for finding a maximum convex polygon. *Comput. Geom.*, 7(3):187–200, 1997.

[GG13] Subir Kumar Ghosh and Partha Pratim Goswami. Unsolved problems in visibility graphs of points, segments, and polygons. *ACM Comput. Surv.*, 46(2):22, 2013.

[Gho97] Subir Kumar Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete Comput. Geom.*, 17(2):143–162, 1997.

[Gho07] Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

[GM91] Subir Kumar Ghosh and David M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM J. Comput.*, 20(5):888–910, 1991.

[GSBG07] Subir Kumar Ghosh, Thomas C. Shermer, Binay K. Bhattacharya, and Partha P. Goswami. Computing the maximum clique in the visibility graph of a simple polygon. *J. Discrete Algorithms*, 5(3):524–532, 2007.

[Her89] John Hershberger. An optimal visibility graph algorithm for triangulated simple polygons. *Algorithmica*, 4(1):141–155, 1989.