On Romeo and Juliet Problems: Minimizing Distance-to-Sight[*]

Hee-Kap Ahn[1], Eunjin Oh[2], Lena Schlipf[3], Fabian Stehn[4], and Darren Strash[5]

1 **Department of Computer Science and Engineering, POSTECH, South Korea**
`heekap@postech.ac.kr`

2 **Department of Computer Science and Engineering, POSTECH, South Korea**
`jin9082@postech.ac.kr`

3 **Theoretische Informatik, FernUniversität in Hagen, Germany**
`lena.schlipf@fernuni-hagen.de`

4 **Institut für Informatik, Universität Bayreuth**
`fabian.stehn@uni-bayreuth.de`

5 **Department of Computer Science, Colgate University, US.**
`dstrash@cs.colgate.edu`

Abstract

We introduce a variant of the watchman route problem, which we call the *quickest pair-visibility* problem. Given two persons standing at points s and t in a simple polygon P with no holes, we want to minimize the distance these persons travel in order to see each other in P. We solve two variants of this problem, one minimizing the longer distance the two persons travel (min-max) and one minimizing the total travel distance (min-sum), optimally in linear time.

1 Introduction

In the watchman route problem, a watchman takes a route to *guard* a given region—that is, any point in the region is visible from at least one point on the route. It is desirable to make the route as short as possible so that the entire area can be guarded as quickly as possible. The problem was first introduced in 1986 by Chin and Ntafos [4] and has been extensively studied in computational geometry [3, 10]. Though the problem is NP-hard for polygons with holes [4, 5, 7], an optimal route can be computed in time $O(n^3 \log n)$ for simple n-gons [6] when the tour must pass through a specified point, and $O(n^4 \log n)$ time otherwise.

In this paper, we study a variant we call the *quickest pair-visibility* problem, which can be stated as follows.

▶ **Problem** (quickest pair-visibility problem). *Given two points s and t in a simple polygon P, compute the minimum distance that s and t must travel in order to see each other in P.*

This problem may sound similar to the shortest path problem between s and t, in which the objective is to compute the shortest path for s to *reach* t. However, they differ even for a simple case: for any two points lying in a convex polygon, the distance in the quickest pair-visibility problem is zero while in the shortest path problem it is their Euclidean distance.

The quickest pair-visibility problem occurs in optimization tasks. For example, mobile robots that use a line-of-sight communication model are required to move to mutually-visible

[*] This work by Ahn and Oh was supported by the MSIT(Ministry of Science and ICT), Korea, under the SW Starlab support program(IITP-2017-0-00905) supervised by the IITP (Institute for Information & communications Technology Promotion.).

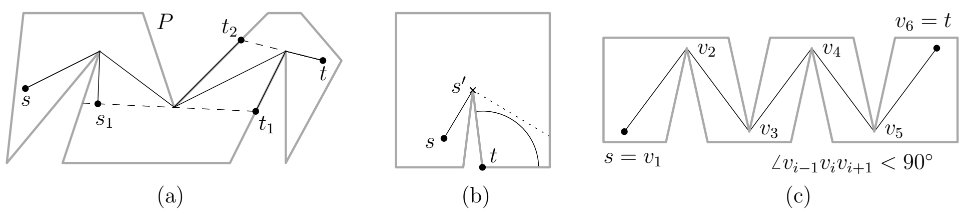

Figure 1 (a) The quickest pair-visibility problem finds two paths $\pi(s, s_1)$ and $\pi(t, t_1)$ such that $\overline{s_1t_1} \subset P$ and $\max\{|\pi(s, s_1)|, |\pi(t, t_1)|\}$ or $|\pi(s, s_1)| + |\pi(t, t_1)|$ is minimized. The quickest visibility problem for query point t finds a shortest $\pi(s, t_2)$ with $\overline{tt_2} \subset P$. (b) **min-max:** Every pair (s', t^*), where t^* is some point within the geodesic disk centered in t with radius $\pi(s, s')$, is an optimal solution to the *min-max* problem. (c) **min-sum:** Every pair (v_i, v_{i+1}) for $1 \leq i < 6$ is an optimal solution to this instance.

positions to establish communication [8]. An optimization task here is to find shortest paths for the robots to meet the visibility requirement for establishing communication among them.

Wynters et al. [12] studied this problem for two agents acting in a polygonal domain in the presence of polygonal obstacles and gave an $O(nm)$-time algorithm for the min-sum variant (where m is the number of edges of the visibility graph of all corners) and an $O(n^3 \log n)$-time algorithm for the min-max variant. A query version of the quickest visibility problem has also been studied [1, 9, 11]. In the query problem, a polygon and a source point lying in the polygon are given, and the goal is to preprocess them and construct a data structure that allows, for a given query point, to find the shortest path taken from the source point to see the query point efficiently. Khosravi and Ghodsi [9] considered the case for a simple n-gon and presented an algorithm to construct a data structure of $O(n^2)$ space so that given a query, it finds the shortest visibility path in $O(\log n)$ time. Later, Arkin et al. [1] improved the result and presented an algorithm for the problem in a polygonal domain. Very recently, Wang [11] presented an improved algorithm for this problem for the case that the number of the holes in the polygon is relatively small. Figure 1(a) illustrates differences in these problems for a simple polygon and two points, s and t, in the polygon.

1.1 Our results

In this paper, we consider two variants of the quickest pair-visibility problem for a simple polygon: either we want to minimize the maximum length of a traveled path (*min-max variant*) or we want to minimize the sum of the lengths of both traveled paths (*min-sum variant*). We give a sweep-line-like approach that “rotates” the lines-of-sight along vertices on the shortest path between the start positions, allowing us to evaluate a linear number of candidate solutions on these lines. Throughout the sweep, we encounter solutions to both variants of the problem. We further show that our technique can be implemented in linear time.

2 Preliminaries

Let P be a simple polygon and ∂P be its boundary. The vertices of P are given in counter-clockwise order along ∂P. We denote the shortest path within P between two points $p, q \in P$ by $\pi(p, q)$ and its length by $|\pi(p, q)|$. We say a point $p \in P$ is visible from another point $q \in P$ (and q is visible from p) if and only if line segment $\overline{pq}$ is completely contained in P.

For two starting points s and t, our task is to compute a pair (s', t') of points such that s'

and t' are visible to each other, where we wish to minimize the lengths of $\pi(s, s')$, and $\pi(t, t')$. In the *min-max* setting, we wish to minimize $\max\{|\pi(s, s')|, |\pi(t, t')|\}$. For the *min-sum* setting, we wish to minimize $|\pi(s, s')| + |\pi(t, t')|$. Note that, for both variants, the optimum is not necessarily unique; see Figure 1(b) and (c).

For our discussion, let (s^*, t^*) be an optimal solution for the instance at hand. Let $V(p)$ denote the visible region for a point p in P, that is, the portion of P that is visible from p. Clearly, $V(p)$ is a *star-shaped* polygon. Moreover, every boundary edge of $V(p)$ is either (part of) an edge of P or a segment $\overline{vq}$ that is contained in P and parallel to $\overline{pv}$, where v is a vertex of P visible from p and q is a point on the boundary of P. We call an edge of the latter type a *window edge* of the visibility region. The structure of $V(p)$ may change as p moves along a path contained in P. It is known that a change to the structure of $V(p)$ occurs if and only if two vertices of P become collinear with p [2].

▶ **Lemma 2.1.** *Unless s and t are visible to each other, the segment $\overline{s^*t^*}$ contains a vertex v of the shortest path $\pi(s, t)$ from s to t.*

It is easy to see by contradiction that $\overline{s^*t^*}$ must contain a vertex v of the boundary of P; using shortest path properties, one can show that v is a vertex of $\pi(s, t)$. The full proof is omitted due to space constraints.

3 Computing All Events for a Sweep-Line-Like Approach

For each vertex v on $\pi(s, t)$ we compute a finite collection of lines through v, each being a configuration at which the combinatorial structure of the shortest paths $\pi(s, s^*)$ and/or $\pi(t, t^*)$ changes. To be more precise, at these lines either the vertices of $\pi(s, s^*)$ or $\pi(t, t^*)$ (except for s^* and t^*) change or the edge of ∂P changes that is intersected by the extension of $\overline{s^*t^*}$. To explain how to compute these lines, we introduce the concept of a *line-of-sight*.

▶ **Definition 3.1** (line-of-sight). We call a segment ℓ a *line-of-sight* if (i) $\ell \subset P$, (ii) both endpoints of ℓ lie on ∂P, and (iii) ℓ is tangent to $\pi(s, t)$ at a vertex $v \in \pi(s, t)$.

We say a segment g is tangent to a path π at a vertex v if $v \in g \cap \pi$ and the local neighborhood of π at all intersections $g \cap \pi$ is on the same side of g. The algorithm we present is in many aspects similar to a sweep-line strategy, except that we do not sweep over the scene in a standard fashion but rotate a *line-of-sight* ℓ in P around the vertices of the shortest path $\pi(s, t) := (s = v_0), v_1, \ldots, v_{k-1}, (t = v_k)$. The process will be initialized with a line-of-sight that contains s and v_1 and is then rotated around v_1 (while remaining tangent to v_1) until it hits v_2, see Figure 2(a). In general, the current line-of-sight is rotated around v_i in a way so that it remains tangent to v_i (it is rotated in the interior of P) until the line-of-sight contains v_i and v_{i+1}, then the process is iterated with v_{i+1} as the new rotation center. The process terminates as soon as the line-of-sight contains v_{k-1} and t.

While performing these rotations around the shortest path vertices, we encounter all combinatorially different lines-of-sight. As for a standard sweep-line approach, we will compute and consider events at which the structure of a solution changes: this is either because the interior vertices of $\pi(s, s^*)$ or $\pi(t, t^*)$ change or because the line-of-sight starts or ends at a different edge of ∂P. These events will be represented by points on ∂P (actually, we introduce the events as vertices on ∂P unless they are already vertices). Between two consecutive lines-of-sight, we compute the local minima of the relevant distances for the variant at hand in constant time and hence encounter all global minima eventually.

There are three event-types to distinguish:

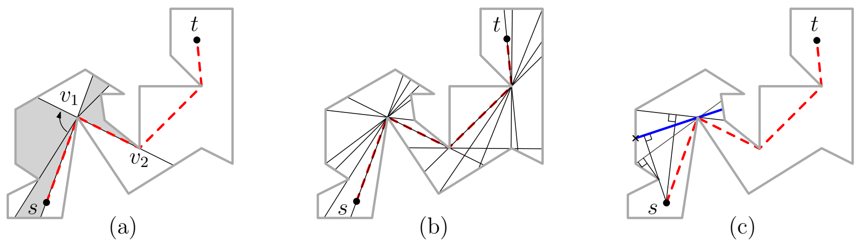

Figure 2 Path- and boundary-events. (a) The first path-event is the line-of-sight through $\overline{sv_1}$. The line-of-sight rotates until it hits the next path-event: the segment through $\overline{v_1v_2}$. (b) All path- and boundary-events: the event-queue is initialized with these events. (c) A bend-event (marked with a cross) occurs between the two boundary-events. The shortest path from s to these segments changes at the bend-event.

1. **Path-Events** are endpoints of lines-of-sight that contain two consecutive vertices of the shortest path $\pi(s,t)$. See Figure 2(a).
2. **Boundary-Events** are endpoints of lines-of-sight that are tangent at a vertex of $\pi(s,t)$ and contain at least one vertex of $P \setminus \pi(s,t)$ (potentially as an endpoint). See Figure 2(b).
3. **Bend-Events** are encountered when, the shortest path of s (or t) to the line-of-sight gains or loses a vertex while rotating the line-of-sight around a vertex v. See Figure 2(c). Note that bend-events can coincide with path- or boundary-events.

We will need to explicitly know both endpoints of the line-of-sight on ∂P at each event and the corresponding vertex of $\pi(s,t)$ on which we rotate.

▶ **Lemma 3.2** (Computing path- and boundary-events). *For a simple polygon P with n vertices and points $s,t \in P$, the queue $\mathcal{Q}$ of all path- and boundary-events of the rotational sweep process, ordered according to the sequence in which the sweeping line-of-sight encounters them, can be initialized in $O(n)$ time.*

Path events coincide with specific vertices of the shortest path map of s (or of t) in P, whereas boundary events are endpoints of specific edges of the shortest path tree of s (or of t) in P. These structures can be constructed and classified in linear time, a full proof is omitted due to space constraints.

Once we initialized the event queue $\mathcal{Q}$, we can now compute and process bend-events as we proceed in our line-of-sight rotations.

▶ **Lemma 3.3.** *All bend-events can be computed in $O(n)$ time, sorted in the order as they appear on the boundary of P.*

Due to space limitations, the proof of Lemma 3.3 is omitted.

4 Algorithm Based on a Sweep-Line-Like Approach

In this section, we present a linear-time algorithm for computing the minimum distance that two points s and t in a simple polygon P travel in order to see each order. We compute all events defined in Section 3 in linear time. The remaining task is to handle the lines-of-sight lying between two consecutive events.

▶ **Lemma 4.1.** *For any two consecutive events, the line-of-sight ℓ lying between them that minimizes the sum of the distances from s and t to ℓ can be found in constant time.*

Proof. Let $\mathcal{L}$ be the set of all lines-of-sights lying between the two consecutive events. Every line-of-sight in $\mathcal{L}$ contains a common vertex v of $\pi(s,t)$. We assume that $\mathcal{L}$ contains no vertical line-of-sight. Otherwise, we consider the set containing all lines-of-sight of $\mathcal{L}$ with positive slopes, and then the set containing all lines-of-sight of $\mathcal{L}$ with negative slopes.

By construction, the second to the last vertex u of $\pi(s,\ell)$ (and $\pi(t,\ell)$) for any $\ell \in \mathcal{L}$ remains the same. We already obtained v and u while computing the events. We will give an algebraic function for the length of $\pi(s,\ell)$ for $\ell \in \mathcal{L}$. An algebraic function for the length of $\pi(t,\ell)$ can be obtained by changing the roles of s and t.

Since the topology of $\pi(s,\ell)$ for every $\ell \in \mathcal{L}$ remains the same, we consider only the length of $\pi(u,\ell)$. Observe that $\pi(u,\ell)$ is a line segment for any $\ell \in \mathcal{L}$, and thus its length is the same as the Euclidean distance between u and ℓ. The length is either the Euclidean distance between u and the line containing ℓ, or the Euclidean distance between u and the endpoint of ℓ closest to u. We show how to handle the first case only because the second case can be handled analogously.

To use this observation, we use $\ell(\alpha)$ to denote the line of slope α passing through v for any $\alpha > 0$. There is an interval I such that $\ell(\alpha)$ contains a line-of-sight in $\mathcal{L}$ if and only if $\alpha \in I$. The Euclidean distance between u and $\ell(\alpha)$ is the same as the distance between u and the line-of-sight contained in $\ell(\alpha)$. Thus, in the following, we consider the distance between u and $\ell(\alpha)$ for every $\alpha \in I$.

Since $\ell(\alpha)$ passes through a common vertex, the line $\ell(\alpha)$ can be represented as the form of $y = \alpha x + f(\alpha)$, where $f(\alpha)$ is a function linear in α. Then, the distance between u and $\ell(\alpha)$ can be represented as the form of $|c_1\alpha + c_2|/\sqrt{\alpha^2+1}$, where c_1 and c_2 are constants depending only on v and u.

Then our problem reduces to the problem of finding a minimum of the function of the form of $(|c_1\alpha + c_2| + |c_1'\alpha + c_2'|)/\sqrt{\alpha^2+1}$ for four constants c_1, c_2, c_1' and c_2', and for all $\alpha \in I$. We can find a minimum in constant time using an elementary analysis. ◀

▶ **Lemma 4.2.** *For any two consecutive events, the line-of-sight ℓ lying between the them that minimizes the maximum of the distances from s and t to ℓ can be found in constant time.*

▶ **Theorem 4.3.** *Given a simple n-gon P with no holes and two points $s, t \in P$, a point-pair (s^*, t^*) such that* i) $\overline{s^*t^*} \subset P$ *and* ii) *either* $|\pi(s,s^*)| + \pi(t,t^*)|$ *or* $\max\{|\pi(s,s^*)|, |\pi(t,t^*)|\}$ *is minimized can be computed in $O(n)$ time.*

Proof. Our algorithm first computes all path- and boundary-events as described in Lemma 3.2. The number of events introduced during this phase is bounded by the number of vertices of the shortest path maps, M_s and M_t, respectively, which are $O(n)$. In the next step, it computes the bend-events on ∂P as described in Lemma 3.3, which can be done in $O(n)$ time. Finally, our algorithm traverses the sequence of events. Between any two consecutive events, it computes the respective local optimum in constant time by Lemma 4.1. It maintains the smallest one among the local optima computed so far, and return it once all events are processed. Therefore the running time of the algorithm is $O(n)$.

For the correctness, consider the combinatorial structure of a solution and how it changes. The path-events ensure that all vertices of $\pi(s,t)$ are considered as being the vertex lying on the segment connecting the solution (s^*, t^*). While the line-of-sight rotates around one fixed vertex of $\pi(s,t)$, either the endpoints of line-of-sight sweep over or become tangent to a vertex of ∂P. These are exactly the boundary-events. Or the combinatorial structure of $\pi(s,s^*)$ or $\pi(t,t^*)$ changes as interior vertices of $\pi(s,s^*)$ or $\pi(t,t^*)$ appear or disappear. These happen exactly at bend events. Therefore, our algorithm returns an optimal point-pair. ◀

Acknowledgments

This research was initiated at the 19th Korean Workshop on Computational Geometry in Würzburg, Germany.

References

1 E. M. Arkin, A. Efrat, C. Knauer, J. S. B. Mitchell, V. Polishchuk, G. Rote, L. Schlipf, and T. Talvitie. Shortest path to a segment and quickest visibility queries. *Journal of Computational Geometry*, 7(2):77–100, 2016.

2 B. Aronov, L. J. Guibas, M. Teichmann, and L. Zhang. Visibility queries and maintenance in simple polygons. *Discrete Comput. Geom.*, 27(4):461–483, 2002.

3 S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete Comput. Geom.*, 22(3):377–402, 1999.

4 W. Chin and S. Ntafos. Optimal watchman routes. In *Proceedings of the 2nd ACM Symposium on Computational Geometry*, pages 24–33, 1986.

5 W. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39–44, 1988.

6 M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell. Touring a sequence of polygons. In *Proc. 35th ACM Symposium on Theory of Computing*, pages 473–482, 2003.

7 A. Dumitrescu and C. D. Tóth. Watchman tours for polygons with holes. *Computational Geometry*, 45(7):326–333, 2012.

8 A. Ganguli, J. Cortes, and F. Bullo. Visibility-based multi-agent deployment in orthogonal environments. In *Proc. Am. Control Conf.*, pages 3426–3431, 2007.

9 R. Khosravi and M. Ghodsi. The fastest way to view a query point in simple polygons. In *Proc. European Workshop on Computational Geometry*, pages 187–190, 2005.

10 J. S. B. Mitchell. Approximating watchman routes. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 844–855, 2013.

11 H. Wang. Quickest visibility queries in polygonal domains. In *Proceedings of the 33rd International Symposium on Computational Geometry*, volume 77, pages 61:1–61:16, 2017.

12 E. L. Wynters and J. S. B. Mitchell. Shortest paths for a two-robot rendez-vous. In *Proc. 5th Canadian Conference on Computational Geometry*, pages 216–221, 1993.