Compatible Paths on Labelled Point Sets

Elena Arseneva* Yeganeh Bahoo† Ahmad Biniaz‡ Pilar Cano§ Farah Chanchary§ John Iacono¶
Kshitij Jain‡ Anna Lubiw‡ Debajyoti Mondal‖ Khadijeh Sheikhan** Csaba D. Tóth††

Abstract

Let P and Q be finite point sets of the same cardinality in $\mathbb{R}^2$, each labelled from 1 to n. Two noncrossing geometric graphs G_P and G_Q spanning P and Q, respectively, are called *compatible* if for every face f in G_P, there exists a corresponding face in G_Q with the same clockwise ordering of the vertices on its boundary as in f. In particular, G_P and G_Q must be straight-line embeddings of the same connected n-vertex graph G. No polynomial-time algorithm is known for deciding whether two labelled point sets admit compatible geometric graphs. The complexity of the problem is open, even when the graphs are constrained to be triangulations, trees, or simple paths.

We give polynomial-time algorithms to find compatible paths or report that none exist in three scenarios: $O(n)$ time for points in convex position; $O(n^2)$ time for two simple polygons, where the paths are restricted to remain inside the closed polygons; and $O(n^2 \log n)$ time for points in general position if the paths are restricted to be monotone.

1 Introduction

Computing noncrossing geometric graphs on finite point sets that are in some sense 'compatible' is an active area of research in computational geometry. The study of compatible graphs is motivated by applications to shape animation and simultaneous graph drawing [4, 12].

Let P and Q be finite point sets, each containing n points in the plane labelled from 1 to n. Let G_P and G_Q be two noncrossing geometric graphs spanning P and Q, respectively. G_P and G_Q are called *compatible*, if for every face f in G_P, there exists a corresponding face in G_Q with the same clockwise ordering of the vertices on its boundary as in f. It is necessary, but not sufficient, that G_P and G_Q represent the same connected n-vertex graph G. Given a pair of labelled point sets, it is natural to ask whether they have compatible graphs, and if so, to produce one such pair, G_P, G_Q. The question can also be restricted to specific graph classes such as paths, trees, triangulations, and so on; previous work (described below) has concentrated on compatible triangulations. Compatible triangulations of polygons are also of interest, which motivated us to examine compatible paths inside simple polygons.

In this paper we examine the problem of computing compatible paths on labelled point sets. Equivalently, we seek a permutation of the labels $1, 2, \ldots, n$ that corresponds to a noncrossing (plane) path in P and in Q. Figures 1(a)–(b) show a positive instance of this problem, and Figures 1(c)–(d) depict an affirmative answer.

Our results. We describe a quadratic-time dynamic programming algorithm that either finds compatible paths for two simple polygons, where the paths are restricted to remain inside the closed polygons, or reports that no such path exists. For the more limited case of two point sets in convex position, we give a linear time algorithm to find compatible paths (if they exist). For two general point sets, we give an $O(n^2 \log n)$-time algorithm to find compatible monotone paths (if they exist). Finding (unrestricted) compatible paths of point sets remains open.

1.1 Background

Saalfeld [11] first introduced compatible triangulations of labelled point sets, which he called "joint" triangulations. In Saalfeld's problem, each point set is enclosed inside an axis-aligned rectangle, and the goal is to compute compatible triangulations (possibly using Steiner points). Although not every pair of labelled point sets admit compatible triangulations, Saalfeld showed that one can always construct compatible triangulations using (possibly an exponential number of) Steiner points.

Aronov et al. [2] proved that $O(n^2)$ Steiner points are always sufficient and sometimes necessary to compatibly

*Université libre de Bruxelles (ULB), Belgium. ea.arseneva@gmail.com

†Department of Computer Science, University of Manitoba, Canada. bahoo@cs.umanitoba.ca

‡Cheriton School of Computer Science, University of Waterloo, Canada. ahmad.biniaz@gmail.com, {k22jain,alubiw}@uwaterloo.ca

§Department of Computer Science, Carleton University, Canada. pilukno@gmail.com, farah.chanchary@carleton.ca

¶Université libre de Bruxelles, Belgium & NYU, USA. jiacono@ac.ulb.be

‖Department of Computer Science, University of Saskatchewan, Canada. dmondal@cs.usask.ca

**NYU Tandon School of Engineering, Brooklyn, USA. khadijeh@nyu.edu

††Department of Mathematics, California State University Northridge, Los Angeles, CA, USA. csaba.toth@csun.edu

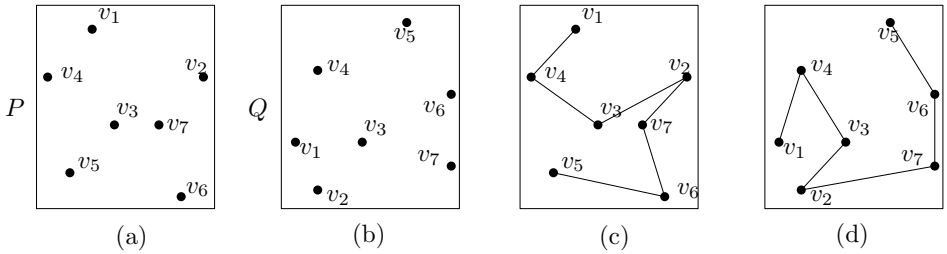

Figure 1: (a)–(b) A pair of labelled point sets P and Q. (c)–(d) A pair of compatible paths.

triangulate two polygons when the vertices of the polygons are labelled $1, 2, \ldots, n$ in clockwise order. Babikov et al. [3] extended the $O(n^2)$ bound to *polygonal regions* (i.e., polygons with holes), where the holes are also labelled 'compatibly' (with the same clockwise ordering of labels). The holes may be single points, so this includes Saalfeld's "joint triangulation" problem. Pach et al. [10] gave an $\Omega(n^2)$ lower bound on the number of Steiner points in such scenarios.

Lubiw and Mondal [9] proved that finding the minimum number of Steiner points is NP-hard for the case of polygonal regions. The complexity status is open for the case of polygons, and also for point sets. Testing for compatible triangulations without Steiner points may be an easier problem. Aronov et al. [2] gave a polynomial-time dynamic programming algorithm to test whether two polygons admit compatible triangulations without Steiner points. But testing whether there are compatible triangulations without Steiner points is open for polygonal regions, as well as for point sets.

The compatible triangulation problem seems challenging even for unlabelled point sets (i.e., when a bijection between P and Q can be chosen arbitrarily). Aichholzer et al. [1] conjectured that every pair of unlabelled point sets (with the same number of points on the convex hull) admit compatible triangulations without Steiner points. So far, the conjecture has been verified only for point sets with at most three interior points.

Let G_S be a complete geometric graph on a point set S. Let $H(S)$ be the *intersection graph* of the edges of G_S, i.e., each edge of G_S corresponds to a vertex in $H(S)$, and two vertices are adjacent in $H(S)$ if and only if the corresponding edges in G_S properly cross (i.e., the open line segments intersect). Every plane triangulation on S has $3n-3-h$ edges, where h is the number of points on the convex hull of S, and thus corresponds to a maximum independent set in $H(S)$. In fact, $H(S)$ belongs to the class of *well-covered graphs*. (A graph is well covered if every maximal independent set of the graph has the same cardinality). A rich body of research attempts to characterize well-covered graphs [6, 13]. Deciding whether two point sets, P and Q, admit compatible triangulations is equivalent to testing whether $H(P)$ and $H(Q)$ have a common independent set of size $3n-3-h$.

2 Paths in Polygons and Convex Point Sets

In this section we describe algorithms to find compatible paths on simple polygons and convex point sets. By compatible paths on polygons, we mean: given two polygons, find two compatible paths on the vertices of the polygons that are constrained to be non-exterior to the polygons. (See Figures 2(a)–(b).) Note that convex point sets correspond to a special case, where the polygons are the convex hulls. Not every two convex point sets admit compatible paths, e.g., 5-point sets where the points are labelled (1,2,3,4,5) and (1,3,5,2,4), resp., in counterclockwise order (Appendix A).

We first give a quadratic-time dynamic programming algorithm for simple polygons, and then a linear time algorithm for convex point sets.

We begin with two properties of any noncrossing path that visits all vertices of a simple polygon. Let P be a simple polygon with vertices $p_1, p_2, \ldots, p_n$ in some order (so the vertices have labels $1, 2, \ldots, n$). Let σ be a label sequence corresponding to a noncrossing path that lies inside P and visits all vertices of P. Define an *interval* on P to be a sequence of labels that appear consecutively around the boundary of P (in clockwise or counterclockwise order). For example, in Figure 2(a), one interval is $(2, 1, 7, 6)$. Define an *interval set* on P to be the unordered set of elements of an interval.

Claim 1 *The set of labels of every prefix of σ is an interval set on P. Furthermore, if the prefix does not contain all the labels, then the last label of the prefix corresponds to an endpoint of the interval.*

Proof. We proceed by induction on t, the length of the prefix, with the base case $t = 1$ being obvious. So assume the first $t-1$ labels form an interval set corresponding to interval I. Let ℓ be the t-th element of σ. Suppose vertex p_ℓ is not contiguous with the interval I on P. Let u and v be the two neighbors of p_ℓ around the polygon P. Then u and v do not belong to I, and so the path must visit both of them after p_ℓ. But then the subpath between u and v crosses the edge of the path that arrives at p_ℓ, contradicting the assumption that the path is noncrossing. Thus vertex p_ℓ must appear just

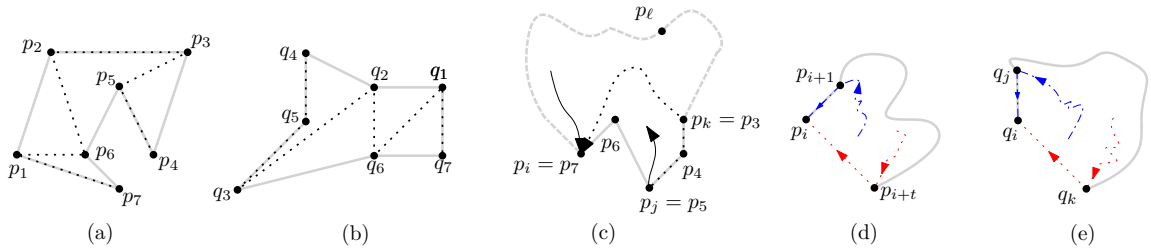

Figure 2: (a)–(b) Compatible paths on a pair of labelled polygons. The paths are drawn with dotted lines. (c) Illustration for Claim 2, where $I = (p_7, p_6, p_5, p_4, p_3)$. (d)–(e) Illustration for the dynamic programming algorithm.

before or after I, forming a longer interval with p_ℓ as an endpoint of the interval. □

Claim 2 *If I is an interval on P and σ does not start or end in I, then the labels of I appear in the same order in σ and in I (either clockwise or counterclockwise). Note that the labels need not appear consecutively in σ.*

Proof. Consider three labels i, j, k that appear in this order in I. Assume, for a contradiction, that these labels appear in a different order in σ and suppose, without loss of generality, that they appear in the order i, k, j in σ. Let ℓ be the last label of σ. Because ℓ does not lie in I, the order of vertices around P is p_i, p_j, p_k, p_ℓ. See, e.g., Figure 2(c) where $i, j, k = 7, 5, 3$. Then the subpath of σ from p_i to p_k crosses the subpath from p_k to p_ℓ, a contradiction. □

2.1 An $O(n^2)$-time dynamic programming algorithm

Let P, Q be two n-vertex simple polygons with labelled vertices. Let p_i (resp., q_i) be the vertex of P (resp., Q) with the label i.

Two vertices of a polygon are *visible* if the straight line segment connecting the vertices lies entirely inside the polygon. We precompute the visibility graph of each polygon in $O(n^2)$ time [8] such that later we can answer any visibility query in constant time.

Notation for our dynamic programming algorithm will be eased if we relabel so that polygon P has labels $1, 2, \ldots, n$ in clockwise order. For each label $i = 1, \ldots, n$ and each length $t = 1, \ldots, n$ let $I_Q(i, t, cw)$ denote the interval on Q of t vertices that starts at q_i and proceeds clockwise. Define $I_Q(i, t, ccw)$ similarly, but proceed counterclockwise from q_i. Define $I_P(i, t, cw)$ and $I_P(i, t, ccw)$ similarly. Note that $I_P(i, t, cw)$ goes from p_i to p_{i+t-1} (index addition modulo n).

We say that a path *traverses* interval $I_Q(i, t, d)$ (where $d = cw$ or ccw), if the path is noncrossing, lies inside Q, visits exactly the vertices of $I_Q(i, t, d)$ and ends at q_i. We make a similar definition for a path to traverse an interval $I_P(i, t, d)$.

Our algorithm will solve subproblems $A(i, t, d_P, d_Q)$ where i is a label from 1 to n, t is a length from 1 to n, and d_P and d_Q take on the values cw or ccw. This subproblem records whether there is a path that traverses $I_Q(i, t, d_Q)$ and a path with the same sequence of labels that traverses $I_P(i, t, d_P)$. If this is the case, we say that the two intervals are *compatible*. Observe that P and Q have compatible paths if and only if $A(i, n, d_P, d_Q)$ is true for some i, d_P, d_Q.

We initialize by setting $A(i, 1, d_P, d_Q)$ to TRUE for all i, d_P, d_Q, and then solve subproblems in order of increasing t. In order for intervals $I_Q(i, t+1, d_Q)$ and $I_P(i, t+1, d_P)$ to be compatible, the intervals of length t formed by deleting the last label, i, must also be compatible, with an appropriate choice of direction (cw or ccw) on those intervals. There are two choices in P and two in Q. We try all four combinations. For a particular combination to 'work' (i.e., yield compatible paths for the original length $t+1$ intervals), we need the last labels of the length t intervals to match, and we need appropriate visibility edges in the polygons for the last edge of the paths.

We give complete details for $A(i, t+1, cw, cw)$. See Figure 2(d)-(e). (The other four possibilities are similar.) Deleting label i from $I_P(i, t+1, cw)$ gives $I_P(i+1, t, cw)$ and $I_P(i+t, t, ccw)$. Let q_j be the vertex following q_i in clockwise order around Q and let q_k be the other endpoint of $I_Q(i, t+1, cw)$ (in practice, for efficiency, we would store k with the subproblem). Deleting label i from $I_Q(i, t+1, cw)$ gives $I_Q(j, t, cw)$ and $I_Q(k, t, ccw)$. The two possibilities for P and Q are shown by blue dash-dotted and red dotted lines in Figures 2(d) and (e), respectively. We set $A(i, t+1, cw, cw)$ TRUE if any of the following four sets of conditions hold:

1. Conditions for $I_P(i+1, t, cw)$ and $I_Q(j, t, cw)$: $i+1 = j$ and $A(i+1, t, cw, cw)$.
2. Conditions for $I_P(i+1, t, cw)$ and $I_Q(k, t, ccw)$: $i+1 = k$ and q_k sees q_i in Q and $A(i+1, t, cw, ccw)$. Note that the last edge of the path in Q must be (q_k, q_i) which is why we impose the visibility condition.
3. Conditions for $I_P(i+t, t, ccw)$ and $I_Q(j, t, cw)$: $i+t = j$ and p_{i+t} sees p_i in P and $A(i+t, t, ccw, cw)$.

4. Conditions for $I_P(i+t, t, ccw)$ and $I_Q(k, t, ccw)$: $i+t = k$ and p_{i+t} sees p_i in P and q_k sees q_i in Q and $A(i+t, t, ccw, ccw)$.

Since there are a quadratic number of subproblems, each taking constant time to solve, this algorithm runs in time $O(n^2)$, which proves:

Theorem 1 *Given two n-vertex polygons, each with points labelled from 1 to n in some order, one can find a pair of compatible paths or determine that none exist in $O(n^2)$ time.*

2.2 A linear-time algorithm for convex point sets

In this section we assume that the input is a pair of convex point sets P, Q, along with their convex hulls.

Given a label x, we first define a *greedy construction* to compute compatible paths starting at x. The output of the construction is an ordered sequence σ_x of labels. Using Claim 1 we keep track of the intervals in P and Q corresponding to σ_x. Initially σ_x contains the label x. Each subsequent step attempts to add a new label to σ_x, maintaining intervals in P and Q. Suppose the intervals corresponding to the current σ_x are I_P and I_Q in P and Q respectively. Let a and b be the labels of the vertices just before and just after interval I_P on the boundary of P. Similarly, let c and d be the labels of the vertices just before and just after interval I_Q on the boundary of Q. If $\{a, b\} = \{c, d\}$, then we add a and b to σ_x in arbitrary order. Otherwise, if there is one label in common between the two sets, we add that label to σ_x. Finally, if there are no common labels, then the construction ends. Let σ_x be a maximal sequence constructed as above.

Lemma 2 *P and Q have compatible paths starting at label x if and only if σ_x includes all n labels.*

Proof. If P and Q have compatible paths with label sequence σ starting at label x then by Claim 1 every prefix of σ corresponds to an interval in P and in Q, and we can build σ_x in exactly the same order as σ.

For the other direction, we claim to construct noncrossing paths in P and Q corresponding to σ_x. Observe that when we add one or two labels to σ_x, we can add the corresponding vertices to our paths because the point sets are convex, so every edge is allowable. Furthermore, the paths constructed in this way are noncrossing because the greedy construction of σ_x always maintains intervals in P and Q. Hence the new edges are outside the convex hull of the paths so far. □

Lemma 2 allows us to find compatible paths (if they exist) in $O(n^2)$ time by trying each label x as the initial label of the path. In order to improve this to linear time, we first argue that when σ_x does not provide compatible paths, then we need not try any of its other labels as the initial label.

Lemma 3 *If σ_x has length less than n, then no label in σ_x can be the starting label for compatible paths of P and Q.*

Proof. Suppose that there are compatible paths with label sequence s_y starting at a label y in σ_x. Let z be the first label that appears in s_y but not in σ_x. Let I_P and I_Q be the intervals corresponding to σ_x in P and Q respectively. By Claim 1 the prefix of s_y before z corresponds to intervals, say I'_P and I'_Q on P and Q, respectively. Then $I'_P \subseteq I_P$ and $I'_Q \subseteq I_Q$ (by our assumption that z is the first label of s_y not in σ_x). Since the vertex with label z must be adjacent to I'_P on the boundary of P and to I'_Q on the boundary of Q, and z does not appear in σ_x, therefore the vertex with label z must be adjacent to I_P on the boundary of P and to I_Q on the boundary of Q. But then our construction would add label z to σ_x. □

We will use Lemma 3 to show that we can eliminate some labels entirely when σ_x is found to have length less than n. Suppose σ_x does not include all labels. Let I_P and I_Q be the intervals on P and Q, respectively, corresponding to the set of labels of σ_x. Let a and b be the labels that appear at the endpoints of I_P.

Suppose P and Q have compatible paths (of length n) with label sequence σ. Then by Lemma 2 the initial and final label of σ lie outside of σ_x. Furthermore, by Claim 2, the set of labels of σ_x must appear consecutively and in the same order around P and around Q (either clockwise or counterclockwise). Our algorithm checks whether I_P and I_Q have the same ordered lists of labels. If not, then there are no compatible paths.

So suppose that I_P and I_Q have the same ordered lists of labels. Then the endpoints of I_Q must have labels a and b. We will now reduce to a smaller problem by discarding all internal vertices of I_P and I_Q. Let P' and Q' be the point sets formed from P and Q, respectively, by deleting the vertices with labels in $\sigma_x - \{a, b\}$.

Lemma 4 *Suppose z is a label appearing in P'. P and Q have compatible paths with initial label z if and only if P' and Q' have compatible paths with initial label z.*

Proof. If P and Q have compatible paths (of length n) with initial label z, then we claim that deleting from those paths the vertices with labels in $\sigma_x - \{a, b\}$ yields compatible paths of P' and Q' with initial label z. It suffices to show that if we delete one vertex from a noncrossing path on points in convex position then the resulting path is still noncrossing. The two edges incident to the point to be deleted form a triangle, and the new path will use the third side of the triangle. Since the points are in convex position, the triangle is empty of

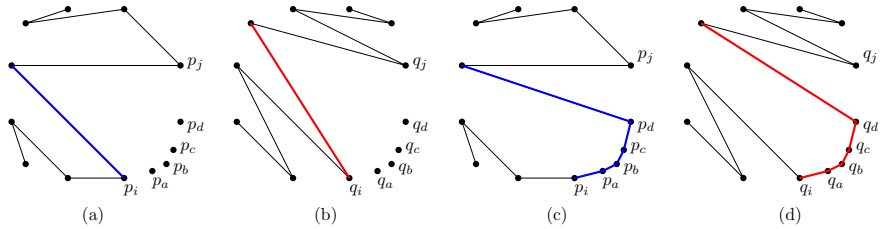

Figure 3: (a)–(b) Compatible paths on the point sets $P \setminus \{p_a, p_b, p_c, p_d\}$ and $Q \setminus \{q_a, q_b, q_c, q_d\}$. (c)–(d) Insertion of the deleted points keeps the paths compatible.

other points, and so the new edge does not cross any other edge of the path.

For the other direction, suppose that σ' is a label sequence of compatible paths of P' and Q' with initial label z. Suppose without loss of generality that label a comes before label b in σ'. Construct a sequence σ by adding the labels of $\sigma_x - \{a, b\}$ after a in σ' in the order that they appear in I_P. It remains to show that the corresponding paths in P and Q are noncrossing. This follows from the fact that in both P and Q the added points appear consecutively around the convex hull following the point with label a. □

We can now prove the main result of this section.

Theorem 5 *Given two sets of n points in convex position (along with their convex hulls) each with points labelled from 1 to n, one can find a pair of compatible paths or determine that none exist in linear time.*

Proof. The algorithm is as described above. At each stage we try some label x to be the initial label of compatible paths, by computing σ_x using the greedy construction. If σ_x has length n we are done. Otherwise if σ_x has length 1 or 2, then we have ruled out the labels in σ_x as initial labels. Finally, if σ_x has length less than n and at least 3 then we test whether the intervals corresponding to σ_x in P and Q have the same ordering, and if they do, then we apply the reduction described above and recurse on the smaller instance as justified by Lemma 4.

The running time of the algorithm is determined by the length of all the σ-sequences we compute. Define a σ-sequence to be 'long' or 'short' depending on whether it contains at least three labels or not. Every long sequence of length ℓ reduces the number of points by $(\ell-2)$ and requires $O(\ell)$ time. Thus, long sequences take $O(n)$ time in total. Computing any short sequence takes $O(1)$ time. Since for each label, we compute σ at most once, the short sequences also take $O(n)$ time in total. □

3 Monotone Paths in General Point Sets

In this section we examine arbitrary point sets in general position, but we restrict the type of path.

Let P be a point set in general position. An ordering σ of the points of P is called *monotone* if there exists some line ℓ such that the orthogonal projection of the points on ℓ yields the order σ. A *monotone path* is a path that corresponds to a monotone ordering. Note that every monotone path is noncrossing.

Two points sets P and Q each labelled $1, 2, \ldots, n$ have *compatible monotone paths* if there is an ordering of the labels that corresponds to a monotone path in P and a monotone path in Q. To decide whether compatible monotone paths exist, we can enumerate all the monotone orderings of P, and for each of them check in linear time whether it determines a monotone path in Q.

A method for enumerating all the monotone orderings of a point set P was developed by Goodman and Pollack:

Theorem 6 (Goodman and Pollack [7]) *Let ℓ_0 be a line not orthogonal to any line determined by two points of P. Starting with $\ell = \ell_0$, rotate the line ℓ through 360° counter-clockwise about a fixed point. Projecting the points onto ℓ as it rotates gives all the possible monotone orderings of P. There are $2\binom{n}{2} = n(n-1)$ orderings, and each successive ordering differs from the previous one by a swap of two elements adjacent in the ordering.*

Furthermore, the sequence of swaps that change each ordering to the next one can be found in $O(n^2 \log n)$ time by sorting the $O(n^2)$ lines (determined by all pairs of points) by their slopes.

This gives a straight-forward $O(n^3)$ time algorithm to find compatible monotone paths, since we can generate the $O(n^2)$ monotone orderings of P in constant time per ordering, and check each one for monotonicity in Q in linear time.

We now present a more efficient $O(n^2 \log n)$ time algorithm. For ease of notation, relabel the points so that the order of points P along ℓ_0 is $1, 2, \ldots, n$. As the line ℓ rotates, let $L_0^P, L_1^P, \ldots L_{t-1}^P$, where $t = n(n-1)$, be the monotone orderings of P, and let S_P be the corresponding swap sequence. Similarly, let $L_0^Q, L_1^Q, \ldots L_{t-1}^Q$ be the monotone orderings of Q and let S_Q be the corresponding swap sequence (Figure 4). We need to find whether there exist some i and j such that $L_i^P = L_j^Q$.

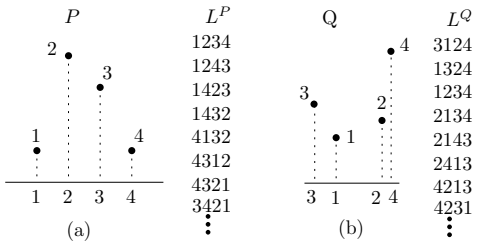

Figure 4: Illustration for computing compatible monotone paths.

As noted above, S_P and S_Q have size $O(n^2)$ and can be computed in time $O(n^2 \log n)$.

Recall that the *inversion number*, $I(L)$ of a permutation L is the number of pairs that are out of order. It is easy to see that the inversion numbers of the L_i^P's progress from 0 to $\binom{n}{2}$ and back again. In particular, $I(L_i^P) = i$ for $0 \leq i \leq \binom{n}{2}$. Our algorithm will compute the inversion numbers of the L_j^Q's, which also have some structure. Let I_j be the inversion number of L_j^Q. Note that we can compute I_0 in $O(n \log n)$ time—sorting algorithms can be modified to do this [5].

Claim 3 *For all j, $1 \leq j \leq n(n-1)$, I_j differs from I_{j-1} by ± 1, and can be computed from I_{j-1} in constant time.*

Proof. L_j^Q is formed by swapping one pair of adjacent elements in L_{j-1}^Q. If this swap moves a smaller element after a larger one then $I_j = I_{j-1}+1$. Otherwise, it is $I_{j-1}-1$. □

The main idea of our algorithm is as follows. If $L_j^Q = L_i^P$, then they must have the same inversion number, I_j. There is one value of i in the range $0 \leq i < \binom{n}{2}$ that gives this inversion number, namely $i = I_j$. There is also one value of i in the second half of the range that gives this inversion number, but we can ignore the second half of the range based on the following:

Remark 1 *If there exist i, j such that $L_i^P = L_j^Q$, then there is such a pair with i in the first half of the index range, i.e., $0 \leq i < \binom{n}{2}$.*

Proof. The second half of each list of orderings contains the reversals of the orderings in the first half [7]. Thus if there is a match $L_i^P = L_j^Q$ then the reversals of the two orderings also provide a match, say $L_{i'}^P = L_{j'}^Q$, and either i or i' is in the first half of the index range. □

Our plan is to iterate through the orderings L_j^Q for $0 \leq j < n(n-1)$. Since each ordering differs from the previous one by a single swap, we can update from one to the next in constant time. For each j, we will check if L_j^Q is equal to $L_{I_j}^P$, i.e., for each j, $0 \leq j < n(n-1)$ we will compute the following four things:

- L_j^Q, I_j, $L_{I_j}^P$, and
- H_j, which is the *Hamming distance*—i.e., the number of mismatches—between L_j^Q and $L_{I_j}^P$

If we find a j with $H_j = 0$ then we output L_j^Q and $L_{I_j}^P$ as compatible monotone paths. Otherwise, we declare that no compatible monotone paths exist. Correctness of this algorithm follows from Remark 1 and the discussion above:

Claim 4 *P and Q have compatible monotone paths if and only if $H_j = 0$ for some j, $0 \leq j < n(n-1)$.*

We now give the details of how to perform the above computations. For $j = 0$ we will compute everything directly, and for each successive j, we will show how to update efficiently. We initialize the algorithm at $j = 0$ by computing L_0^Q and I_j in $O(n \log n)$ time, $L_{I_j}^P$ in $O(n^2)$ time, and H_j in linear time.

Now consider an update from $j-1$ to j. As already mentioned, L_j^Q differs from L_{j-1}^Q by one swap of adjacent elements, so we can update in constant time. By Lemma 3, I_j differs from I_{j-1} by ± 1 and we can compute it in constant time. This also means that $L_{I_j}^P$ differs from $L_{I_{j-1}}^P$ by one swap of adjacent elements, so we can update it in constant time.

Finally, we can update the Hamming distance in a two-step process as the two orderings change. When we update from L_{j-1}^Q to L_j^Q, two positions in the list change, and we can compare them to the same positions in $L_{I_{j-1}}^P$ to update from H_{j-1} to obtain the number of mismatches between L_j^Q and $L_{I_{j-1}}^P$. When we update to $L_{I_j}^P$, two positions in this list change, and we can compare them to the same positions in $L_{I_j}^Q$ to update to H_j. This two-step process takes constant time.

In total, we spend $O(n^2)$ time on initialization and constant time on each of $O(n^2)$ updates, for a total of $O(n^2)$ time. We thus obtain the following theorem.

Theorem 7 *Given two point sets, each containing n points labelled from 1 to n, one can find a pair of compatible monotone paths or determine that none exist in $O(n^2 \log n)$ time.*

Acknowledgement: We thank the organizers of the Fields Workshop on Discrete and Computational Geometry, held in July 2017 at Carleton University. E. Arseneva is supported in part by the SNF Early Postdoc Mobility grant P2TIP2-168563 and by F.R.S.-FNRS; A. Biniaz, K. Jain, A. Lubiw, and D. Mondal are supported in part by NSERC.

References

[1] O. Aichholzer, F. Aurenhammer, F. Hurtado, and H. Krasser. Towards compatible triangulations. *Theoretical Computer Science*, 296(1):3–13, 2003.

[2] B. Aronov, R. Seidel, and D. L. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry*, 3:27–35, 1993.

[3] M. Babikov, D. L. Souvaine, and R. Wenger. Constructing piecewise linear homeomorphisms of polygons with holes. In *Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG)*, 1997.

[4] W. V. Baxter III, P. Barla, and K. Anjyo. Compatible embedding for 2D shape animation. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):867–879, 2009.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[6] A. S. Finbow, B. L. Hartnell, and M. D. Plummer. On well-covered pentagonalizations of the plane. *Discrete Applied Mathematics*, 224:91–105, 2017.

[7] J. E. Goodman and R. Pollack. On the combinatorial classification of nondegenerate configurations in the plane. *Journal of Combinatorial Theory, Series A*, 29(2):220–235, 1980.

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

[9] A. Lubiw and D. Mondal. On compatible triangulations with a minimum number of Steiner points. In *Proceedings Canadian Conference on Computational Geometry (CCCG)*, pages 101–106, 2017. http://arxiv.org/abs/1706.09086.

[10] J. Pach, F. Shahrokhi, and M. Szegedy. Applications of the crossing number. *Algorithmica*, 16(1):111–117, 1996.

[11] A. Saalfeld. Joint triangulations and triangulation maps. In *Proceedings of the Third Annual Symposium on Computational Geometry (SoCG)*, pages 195–204. ACM, 1987.

[12] V. Surazhsky and C. Gotsman. High quality compatible triangulations. *Engineering with Computers*, 20(2):147–156, 2004.

[13] D. Tankus and M. Tarsi. The structure of well-covered graphs and the complexity of their recognition problems. *J. Comb. Theory, Ser. B*, 69(2):230–233, 1997.

Appendix A

In this section we show that for every $n \geq 5$, there exist two convex labelled point sets, each containing n points, that do not admit compatible trees. Note that this also rules out the existence of compatible paths.

Claim 5 *Let P and Q be point sets in convex position, each containing $n \geq 2$ points labelled by $\{1, 2, \ldots, n\}$. If they admit a compatible tree that is not a star, then there exists a partition $\{1, 2, \ldots, n\} = A \cup B$ such that $2 \leq |A| \leq |B| \leq n-2$ such that A and B are interval sets for both P and Q.*

Proof. Suppose that P and Q admit a compatible tree T, which is not a star. Then T has an edge e between two vertices of degree two or higher. The deletion of e decomposes T into two subtrees, say T_1 and T_2, each with at least two vertices. The vertex sets of T_1 and T_2, resp., correspond to an interval set in P and Q. □

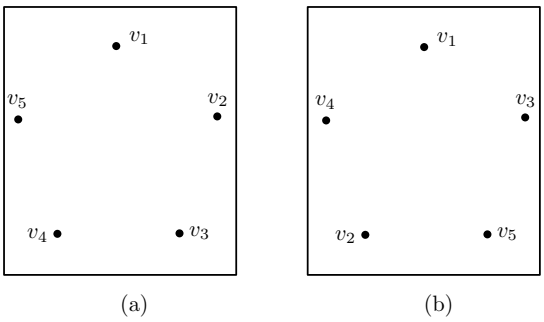

Figure 5: Illustration for Lemma 8.

Theorem 8 *For every integer $n \geq 5$, there exist two sets, P_n and Q_n, each of n labelled points in convex position, such that P_n and Q_n do not admit any compatible tree.*

Proof. For $n = 5$, let P_5 and Q_5 be point sets labelled $(1, 2, 3, 4, 5)$ and $(1, 3, 5, 2, 4)$, respectively, in counterclockwise order (Figure 5). If a compatible star exists, then the four leaves would appear in the same counterclockwise order in both P_5 and Q_5 (by the definition of compatibility). However, the two convex sets have distinct counterclockwise 4-tuples. If there is a compatible tree that is not a star, then by Claim 5, a 2-element set $A \subset \{1, 2, 3, 4, 5\}$ is an interval set for both P_5 and Q_5. However, all five consecutive pairs along the convex hull of P_5 are nonconsecutive in the convex hull of Q_5. Therefore, P_5 and Q_5 do not admit any compatible tree.

For $n > 5$, we can construct P_n and Q_n analogously. Let P_n be labelled $(1, 2 \ldots, n)$ in counterclockwise order. For $i = 0, 1, 2, 3, 4$, let N_i be the sequence of labels in $\{1, 2, \ldots, n\}$ congruent to i modulo 5 in increasing order. Now let Q_n be labelled by the concatenation of the sequences N_1, N_3, N_0, N_2, N_4 in counterclockwise order.

If a compatible star exists, then the $n-1$ leaves would appear in the same counterclockwise order in both P_n and Q_n (by the definition of compatibility). However, the both neighbors of a vertex in P_n are different from the two neighbors in Q_n, consequently P_n and Q_n do not share any counterclockwise $(n-1)$-tuple. If there is a compatible tree that is not a star, then by Claim 5, there is a partition $\{1, 2, \ldots, n\} = A \cup B$ into interval sets, where $|A|, |B| \geq 2$. However, A and B cannot partition any subset of 5 consecutive elements in sequence $(1, 2, \ldots, n)$, similarly to the case when $n = 5$. Consequently, P_n and Q_n do not admit any compatible tree. □