Non-crossing matchings of points with geometric objects
Non-crossing Matchings of Points with Geometric Objects

Greg Aloupis\textsuperscript{a}, Jean Cardinal\textsuperscript{a}, Sébastien Collette\textsuperscript{a,2}, Erik D. Demaine\textsuperscript{b}, Martin L. Demaine\textsuperscript{b}, Muriel Dulieu\textsuperscript{c}, Ruy Fabila-Monroy\textsuperscript{d}, Vi Hart\textsuperscript{e}, Ferran Hurtado\textsuperscript{f}, Stefan Langerman\textsuperscript{a,3}, Maria Saumell\textsuperscript{f}, Carlos Seara\textsuperscript{f}, Perouz Taslakian\textsuperscript{a}

\textsuperscript{a}Université Libre de Bruxelles, CP212, Bld. du Triomphe, 1050 Brussels, Belgium.
\textsuperscript{b}MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA.
\textsuperscript{c}Polytechnic Institute of NYU, USA.
\textsuperscript{d}Departamento de Matemáticas, CINVESTAV, México DF, México
\textsuperscript{e}Stony Brook University, Stony Brook, NY 11794, USA.
\textsuperscript{f}Universitat Politècnica de Catalunya, Jordi Girona 1–3, E-08034 Barcelona, Spain.

Abstract

Given an ordered set of points and an ordered set of geometric objects in the plane, we are interested in finding a non-crossing matching between point-object pairs. In this paper, we address the algorithmic problem of determining whether a non-crossing matching exists between a given point-object pair. We show that when the objects we match the points to are finite point sets, the problem is NP-complete in general, and polynomial when the ob-

\textsuperscript{1}This work was partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040; the Communauté française de Belgique - ARC; and the ESF EUROCORES programme EuroGIGA, CRP ComPoSe: Fonds National de la Recherche Scientifique (F.R.S.-FNRS) - EUROGIGA NR 13604, for Belgium, and MICINN Project EUI-EURC-2011-4306, for Spain.

\textsuperscript{2}Chargé de recherches du F.R.S.-FNRS.

\textsuperscript{3}Maître de recherches du F.R.S.-FNRS.
jects are on a line or when their size is at most 2. When the objects are line segments, we show that the problem is NP-complete in general, and polynomial when the segments form a convex polygon or are all on a line. Finally, for objects that are straight lines, we show that the problem of finding a min-max non-crossing matching is NP-complete.

1. Introduction

Finding a matching between pairs of plane objects that connects these objects by a set of non-crossing line segments is a natural problem that has been frequently studied in computational geometry. It is well known, for instance, that given two sets of \( n \) points in the plane, say \( n \) red points and \( n \) blue points, there always exists a non-crossing perfect matching between red and blue points. In particular, it is not difficult to show that the minimum Euclidean length matching, if it exists, is non-crossing. Kaneko and Kano [26] survey a number of related results. Algorithms for finding minimum sum and minimum bottleneck distance red-blue matchings are given in [18, 32].

In this paper, we investigate related questions for general plane objects instead of points. Again, matchings are represented by line segments, but here the endpoints can be placed anywhere inside the corresponding matched objects. Note that as a consequence of the aforementioned result on points, there always exists a non-crossing matching between two sets of objects. Here we consider the problem where we are given object pairs (i.e. a point and the geometric object it must be matched to) and need to find a set of non-crossing matching edges, if one exists. This can be seen as a 1-regular graph drawing problem with constraints on the location of vertices.

Related work. Problems on matchings have an important role in combinatorial graph theory, both for theoretical and applied aspects; hence a lot of research is devoted to the study of these problems (for example, see [29]). Suppose we are given an embedding of a graph in the Euclidean plane, where the vertices are points in the plane, edges are rectilinear line segments, and weights on these edges represent the Euclidean distance between the vertices they connect. Elementary geometry tells us that the sum of any pair of opposite sides of a convex quadrilateral is strictly smaller than the sum of the diagonals. Remarkably, this implies that the minimum weight matching in any straight line embedding of the complete graphs \( K_{2n} \) and \( K_{n,n} \) consists of pairwise non-crossing segments. These geometric graph problems can be
solved using generic algorithms for weighted graphs. However, in the planar case just mentioned, Vaidya [32] proved that it is possible to obtain specialized algorithms with better running times (the title of his paper is especially suggestive: *Geometry helps in matching*). In particular, in [32] the running time of the generic algorithm for the bipartite case was reduced from $O(n^3)$ to $O(n^{2.5}\log n)$. This was later improved to $O(n^{2+\varepsilon})$ by Agarwal et al. [2]. Similar results have been obtained for other matching variations, such as *bottleneck matching* or *uniform matching*, in the work of Efrat, Itai and Katz [18]. The authors consider matchings as an approach for the problem of matching a point set $A$ with a point set $B$, where $A$ must be moved in some way to coincide as much as possible with $B$ or one of its subsets. This is a fundamental problem in pattern recognition [7, 10, 11, 13, 14, 15, 23, 24, 25]. Another matching variation is *C-matching* as described by ´Abrego et al. [1]. Here the authors consider the problem of matching a given set of points with a set of geometric objects such that every geometric object contains exactly two points. The objects they consider are circles and isothetic squares, and show the existence and properties of such matchings. Bereg et al. [9] consider *C*-matchings for axis-aligned squares and rectangles.

The non-crossing requirement in our problems is quite natural in geometric scenarios (see for example [3, 4, 31]), and the family of geometric problems that we consider has several applications; these applications include geometric shape matching [6, 16, 21, 22] (see also the references we give for geometric pattern recognition), colour-based image retrieval [16], and computational biology [17, 20].
Our results. Throughout the paper, we let $P := \{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane and $T := \{t_1, t_2, \ldots, t_n\}$ be a set of plane objects. A matching for a pair $(P, T)$ consists of a set of line segments, called edges, of the form $\{p_1m_1, p_2m_2, \ldots, p_nm_n\}$, where $m_i \in t_i$. A matching is said to be non-crossing if no pair of matching edges properly cross. This is illustrated in Figure 1.

We consider the problem of deciding whether a non-crossing matching exists for a given pair $(P, T)$. In cases where a non-crossing matching always exists, we consider the problem of finding the matching that minimizes either the length of the longest edge, or the sum of the lengths of all the edges.

In Section 2, we study the case where the objects $t_i$ are finite point sets. We prove that the decision problem is NP-complete in general, but becomes polynomial when every $t_i$ has size at most two, or when all the $t_i$ are on a line. In Section 3 we consider $T$ to be a set of line segments and prove that the $(P, T)$ matching problem is NP-complete. We also consider special cases, such as the case when the line segments form a convex polygon surrounding all points in $P$ (Section 4), or the case when segments belong to a single line (Section 5). We show that these special cases have polynomial solutions. Finally, in Section 6, we consider the problem of matching points with lines. In this variation, a non-crossing matching always exists; but we show that the optimization problems are NP-hard.

2. Matching points with finite point sets

We first prove that if the objects $t_i$ are pairs of points, then we can decide whether there exists a non-crossing matching in polynomial time. On the other hand, if the sets $t_i$ may contain three points or more, the problem becomes NP-complete. This situation is similar to that of the $k$-satisfiability problem ($k$-SAT). In $k$-SAT we are given a boolean formula $f$ of the form $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ (where each $C_i$ is an OR clause of $k$ variables), and we are required to find a truth assignment of its variables that satisfy the formula. It is well-known that 2-SAT has a polynomial-time solution whereas $k$-SAT is NP-complete for $k \geq 3$. The 2-SAT problem can be solved in polynomial time by exploiting the fact that, if in a clause a variable is set to false, it forces the other variable to be set to true.
Theorem 1. Given an ordered set \( P \) of points and an ordered set \( T \) of pairs of points, there is an algorithm that decides in \( O(n^2) \) time whether \((P,T)\) has a non-crossing matching.

Proof. We will prove the theorem by showing that the given matching problem reduces to 2-SAT, which is known to have an \( O(n^2) \) running time. Assume that the elements of each \( t_i \) are labeled arbitrarily “\( T_i \)” and “\( F_i \)” (thus \( t_i = \{T_i, F_i\} \)). We think of each \( p_i \) as a boolean variable, so that if we match \( p_i \) with \( T_i \) then \( p_i \) is set to “true”, and if \( p_i \) is matched with \( F_i \), it is set to “false”. Let \( X_i \) equal to \( T_i \) or \( F_i \), and \( Y_j \) equal to \( T_j \) or \( F_j \). In \( O(n^2) \) time, we construct a 2-SAT instance having variables \( x_0, x_1, \ldots, x_{n-1} \) as follows: Consider the segments \( p_i, X_i \) for all \( i = 0, 1, \ldots, n-1 \). For each pair of intersecting segments \( p_i, X_i \) and \( p_j, Y_j \), we construct the 2-SAT clause

- \((x_i, x_j)\) if \( X_i = T_i \) and \( Y_j = F_j \),
- \((x_i, \neg x_j)\) if \( X_i = F_i \) and \( Y_j = T_j \),
- \((\neg x_i, x_j)\) if \( X_i = T_i \) and \( Y_j = F_j \), or
- \((\neg x_i, \neg x_j)\) if \( X_i = T_i \) and \( Y_j = T_j \).

With this construction is it easy to see that if there is a solution for \((P,T)\) where the two vertices \( p_i \) and \( p_j \) have a valid non-crossing perfect matching \( p_i, X_i \) and \( p_j, Y_j \), then the corresponding 2-SAT clause has a valid truth assignment if we set \( x_i \) to \( X_i \) and \( x_j \) to \( X_j \). Conversely, if there exists a truth assignment that sets a 2-SAT clause \((x_i, x_j)\) to “true” then there exists a matching for \( p_i \) and \( p_j \). Therefore, the matching instance \((P,T)\) has a non-crossing perfect matching if and only if the corresponding 2-SAT instance has a valid truth assignment. Since the 2-SAT instance is constructed in \( O(n^2) \) time and solving 2-SAT is known to be possible in \( O(n^2) \) time, the overall complexity of the matching algorithm is \( O(n^2) \). \( \square \)

2.1. Matching points with triples

Theorem 2. Given an ordered set \( P \) of points and an ordered set \( T \) of triples of points, it is NP-complete to decide whether \((P,T)\) has a non-crossing matching. The problem remains NP-complete even if each triple of points is horizontally collinear.

---

\(^4\)Here, and throughout the rest of the paper, by an ordered set we mean a totally ordered set.
Proof. First we argue that the problem is in NP. Our input is a set of point-triple pairs. A matching can be specified combinatorially by listing which point in each triple \( t_i \) gets matched with the corresponding point \( p_i \). In time polynomial in the length of the input, we can check whether such a matching is non-crossing. Hence the problem is in NP.

It remains to show that the problem is NP-hard. We reduce from the planar 3-SAT problem, which is a version of 3-SAT whose implication graph (the bipartite graph having the variables on one side, the clauses on the other, and an edge between a variable \( x \) and a clause \( C \) if and only if \( x \) appears in \( C \)) is planar. Planar 3-SAT is known to be NP-hard [28]. Given an instance \( SAT \) of the planar 3-SAT, we will construct an instance \( (P,T) \) of the problem of matching a set of points \( P \) to a set of triples \( T \) such that \( SAT \) has a valid truth assignment if and only if \( (P,T) \) has a non-crossing matching between point-triple pairs. Every boolean variable in \( SAT \) is represented in \( (P,T) \) by points (or a triple in which two points are identical). Thus every point \( v_i \) can be matched in exactly two ways (see Figure 2). To each variable, we associate a wire gadget that is composed of a set of pairs \((v_i,q_i)\) (see Figure 3). These pairs are chosen so that once the edge for one of the points is selected, all the others are determined (given that we require a non-crossing matching). Hence in a non-crossing matching, the wire can only be in one of two distinct states, corresponding to the value of the variable. Such a wire can be split using the gadget shown in Figure 4.

Finally, we associate a pair \((p_j,t_j)\), \( p_j \in P \), \( t_j \in T \) to the \( j \)-th clause of the given 3-SAT formula, where \( t_j \) is a triple of points. The three possible edges connecting \( p_j \) to \( t_j \) correspond to the choice of the literal that will satisfy the clause. The three line segments between \( p_j \) and the three points of \( t_j \) interfere with the wires corresponding to the three variables used in the clause. Using the layout of Figure 5, a matching edge for the clause crosses an edge of the wire if and only if the value of the literal encoded in the edge is not compatible with the value of the variable encoded in the wire. In other words, \( p_j \) connects to point \( a \) of \( t_j \) (representing some variable \( v_i \) of the \( j \)-th
Figure 3: Wire gadget.

Figure 4: Splitter gadget.
clause of the 3-SAT formula) if and only if matching the variable gadget $v_i$ as either true or false sets the literal representing $a$ to true.

![Clause gadget](image)

Figure 5: Clause gadget.

Using standard layout techniques for planar graphs (see [28]), we can represent the variable-clause incidence graph of the given 3-SAT formula using the wire and clause constructions above. This layout guarantees that there exists a satisfying assignment for the 3-SAT instance if and only if there exists a non-crossing matching for $(P,T)$. If the 3-SAT instance has a valid truth assignment, then every clause has at least one literal set to “true”. In the constructed matching instance, this is equivalent to connecting every $p_i$ to at least one of the points in $t_i$ by a segment that does not cross any other. On the other hand, assume that $(P,T)$ has a valid non-crossing matching. Then every variable gadget has a non-crossing matching that connects a point $v_i$ in either of two ways, “true” or “false”; moreover, this matching ensures that in every clause gadget, a vertex $p_j$ has a non-crossing matching to at least one of the three points of $t_j$. If we now assign the values of the variable gadgets in $(P,T)$ to the variables of the 3-SAT instance $SAT$, then every clause in $SAT$ will have at least one literal set to “true”. To conclude the proof, we note that the number of points created in $(P,T)$ for every variable and clause in $SAT$ is a polynomial function of the input to the problem; hence, our reduction is polynomial in the size of the input to the 3-SAT instance.

Finally, observe that our wire and clause layout may be constructed such
that the points in a triple are collinear. Thus the problem remains NP-complete even in this restricted version.

2.2. Matching points with \(k\)-tuples

**Theorem 3.** Given an ordered set \(P\) of points and an ordered set \(T\) of \(k\)-tuples (where each \(t_i\) is a set \(\{t_{i1}, t_{i2}, \ldots, t_{ik}\}\) of \(k\) points), if every edge \([p_i t_{ij}]\) crosses at most \(c < 0.183k\) other edges of the form \([p_i t_{ij'}]\), then there exists a non-crossing matching between \(P\) and \(T\).

**Proof.** We apply the probabilistic method [5], and match every point \(p_i\) with \(t_{ij}\), where \(j\) is chosen randomly in \(\{1, \ldots, k\}\). We need to show that there is a positive probability that the resulting matching is non-crossing. Let \(M\) denote the random matching.

We define a bad event as two edges of \(M\) of the form \([p_i t_{ij}]\) and \([p_i t_{ij'}]\) that cross. A bad event has probability either equal to 0 (if the edges are not crossing) or to exactly \(q := 1/k^2\). Two bad events are dependent whenever the two pairs of points of \(P\) involved intersect. Hence every bad event depends on at most \(d := 2ck\) other bad events (since there are \(k\) possible edges for each of two points, and every such edge intersects at most \(c\) others). By Lovász’ Local Lemma [19], if

\[
eq d + 1 \leq 1
\]

(where \(e\) is Euler’s number), then there is a nonzero probability that no bad event occurs. This means that a non-crossing matching exists. This yields

\[
eq \frac{1}{k^2}(2ck + 1) \leq 1
\]

\[
c \leq \frac{k}{2e} - \frac{1}{2k} \approx 0.183k
\]

Note that our proof does not use geometry, so it is likely that the constant 0.183 can be improved. The proof can also be made constructive using a recent result from Moser [30].

2.3. Matching points with \(k\)-tuples on a line

**Theorem 4.** Given an ordered set \(P\) of points and an ordered set \(T\) of \(k\)-tuples of points on a line, we can decide in \(O(k^3n^2 + k^2n^3)\) time whether \((P, T)\) has a non-crossing matching.
Proof. Without loss of generality, assume all the tuples are on a horizontal line $L$. Assume also that all points are on one side of $L$; otherwise we may consider each problem separately as the matching edges on each side of $L$ do not interact. We now show how to build a dynamic programming table that solves the problem.

In any solution to the problem, if a matching edge $e$ is part of the solution, then there is no matching edge that intersects $e$. Therefore, we can consider the regions on each side of $e$ (sub-problems) separately and determine whether they in turn have a valid solution. To achieve this, we will consider the points of $P$ top-to-bottom – the points with largest $y$-coordinate first, – and based on possible matching edges, split the problem into independent sub-problems. A sub-problem $(P', T')$ is defined as follows (see Figure 6): given a trapezoid $A$ with one edge adjacent to $L$ and an edge parallel to $L$, we want to decide if it is possible to find a non-crossing matching completely contained in the region $A$ for all the points contained in $A$, i.e., we want to solve the problem with $P' = P \cap A$ and $T'$ containing the subsets of the tuples of $T$ contained in $A$. If $A$ does not contain at least one point of $P$ (sub-problem of size 0), it is trivially true that there is a non-crossing matching. Otherwise, to solve the sub-problem we consider the topmost point $p$ in $A$. It has at most $k$ possible matching edges. If it has no possible matching edge, i.e., if all points that $p$ could be matched to in $T$ are out of $A$, then there is no valid matching.

Each of the possible matching edges for $p$ defines two new independent sub-problems (see Figure 7) in the trapezoids $A_1$ and $A_2$, whose sizes are strictly smaller than that of the original problem, as there is one less point to match. Each of the trapezoids $A_1$ and $A_2$ is defined by a possible matching edge of $p$, an edge bounding $A$, the line $L$, and a line through $p$ parallel to $L$. Note that as $p$ is the topmost point of $A$, then the region $A \setminus (A_1 \cup A_2)$ contains no points of $P$; this implies that the union of the regions of the
sub-problems of $A$ will contain all the points in $A$, and hence no point of $P$
will be ignored in the process.

To decide whether a matching exists for the original sets $P$ and $T$, we solve
the sub-problem defined by the bounding box of both $P$ and $T$. Notice that
all the sub-problems correspond to trapezoids defined by a pair of possible
matching edges or by the edges of the bounding box.

The dynamic programming table has $kn + 2$ rows and $kn + 2$ columns,
each of which corresponds to a possible matching edge or one of the left and
right edges of the bounding box; the cells correspond to sub-problems (a pair
of non-adjacent edges defines a trapezoid), and we fill them with true or false
values depending on whether or not a matching exists for the considered sub-
problem. Filling a cell of the table corresponds to first finding the topmost
point within the sub-problem in linear time, and then solving at most $k$ pairs
of sub-problems, which implies at most $2k$ lookups in the table for each of
the $O(k^2n^2)$ cells. Therefore, the total time and space required to solve the
problem is $O(k^2n^2(k + n)) = O(k^3n^2 + k^2n^3)$.

\begin{corollary}
Given an ordered set $P$ of points and an ordered set $T$ of
triples of points on a line, we can decide in $O(n^3)$ time whether $(P, T)$ has a
non-crossing matching.
\end{corollary}

This corollary shows that the additional restriction of having points on
a line greatly simplifies the problem, because the problem is NP-hard in the
general case, but is polynomial for points on a line.

3. Matching points with line segments: general case

In this section we show that deciding the existence of a non-crossing
matching between a set of points and a set of line segments is NP-complete,
even if the segments are all horizontal.
Theorem 5. Given an ordered set $P$ of points and an ordered set $T$ of line segments, it is NP-complete to decide whether $(P, T)$ has a non-crossing matching. The problem remains NP-complete even if all line segments in $T$ are horizontal.

Proof. First we argue that the problem is in NP. It suffices to show that only a polynomial number of points along segments of $T$ need to be considered for a non-crossing matching. We construct the arrangement of lines between all pairs of points among the union of $P$ and the endpoints of all segments in $T$. This arrangement divides each segment of $T$ into subsegments, with the property that all points in the relative interior of a subsegment are equivalent matching solutions. Thus we can choose the midpoint and the endpoints of each subsegment as the canonical points representing possible choices for a non-crossing matching of $(P, T)$. Any matching can be rounded to use only points in $P$ and canonical points of subsegments in $T$, without adding any additional crossings. Therefore a matching can be represented as a combinatorial object on a polynomial number of points. Given such a representation, we can test in polynomial time whether the matching is non-crossing. Hence the problem is in NP.

It remains to show that the problem is NP-hard. We reduce from the non-crossing matching problem for an ordered set $P'$ of points and an ordered set $T'$ of horizontally collinear triples of points, which is NP-hard by Theorem 2. For each point $p \in P'$ and corresponding triple $t \in T'$, we place three points $p_1, p_2, p_3$ in $P$ and three corresponding triples of segments $t_1, t_2, t_3$ in $T$; refer to Figure 8.

Suppose $t = (a, b, c)$ with $a$, $b$, and $c$ appearing left to right along a
horizontal line. Let \( p_1 = p \) and \( t_1 \) be the segment from \( a \) to \( c \). Next, we choose a small subsegment of \( t_1 \) containing \( a \), and similarly we choose small subsegments containing \( b \) and \( c \) (“small” means that the subsegments do not cross any lines of the arrangement described above). Connecting the endpoints of these subsegments to \( p \) gives us three narrow triangles. We place \( p_2 \) on the right edge of the triangle containing \( a \); we place two short horizontal segments \( t_2 \) and \( t_3 \) both having their left endpoint on the left edge of the triangle containing \( b \) and right endpoint on the right edge of the triangle containing \( b \); and we place \( p_3 \) on the left edge of the triangle containing \( c \). Any matching edges connecting \( p_2 \) to \( t_2 \) and \( p_3 \) to \( t_3 \) block the ranges between the narrow triangles. This forces the matching edge connecting \( p_1 \) to \( t_1 \) to lie in one of the three narrow triangles, effectively matching \( p \) with either \( a \), \( b \), or \( c \). Thus \( (P,T) \) has a non-crossing matching if and only if \( (P',T') \) has a non-crossing matching. Note that the proof holds when the segments \( t_i \) are horizontal. \( \square \)

4. Matching points with an enclosing convex polygon

In this special case of matching points with line segments, we assume the segments are the edges of a convex polygon and the points to be matched are inside the polygon.

We first describe some geometric properties of the input of this problem. We then describe an algorithm that finds a non-crossing matching (if one exists) between a given set of point-segment pairs where the line segments form a convex polygon enclosing the points. Our algorithm runs in \( O(n \log^2 n) \) time and allows a minimum-length and minimum max-edge-length matching to be extracted easily.

4.1. Structural properties

Let \( D^o = \{\Delta_1^o, \Delta_2^o, \ldots, \Delta_n^o\} \) be a set of triangles where each \( \Delta_i^o \) is the triangle with apex \( p_i \) and base \( t_i \). Any valid matching edge \( e_i \) must lie inside \( \Delta_i^o \). Depending on the positions of other triangles in \( D^o \), some candidate positions for \( e_i \) can be identified as invalid because they would always cross other matching edges. By identifying such cases, triangle \( \Delta_i^o \) can be reduced to a smaller triangle \( \Delta_i \). At any time, the reduced triangle \( \Delta_i \) has apex \( p_i \) but its opposite base is a subsegment of \( t_i \). Initially, \( \Delta_i = \Delta_i^o \).
There are four ways in which two triangles $\Delta_i$ and $\Delta_j$ interact. The second case leads to a reduction rule. We describe the four cases below (see Figure 9):

1. $\Delta_i, \Delta_j$ are disjoint. In this case there will never be a direct interaction between the two.
2. $p_j$ is in $\Delta_i$, but $p_i$ is not in $\Delta_j$. In this case $\Delta_i$ should be reduced so that the two triangles become tangent (so that $p_j$ is no longer in $\Delta_i$).
3. $p_i$ is in $\Delta_j$ and $p_j$ is in $\Delta_i$. We call $\Delta_i$ and $\Delta_j$ inverted triangles, and cannot immediately make a reduction.
4. Both edges incident to each of $p_i$ and $p_j$ pairwise intersect. Then no non-crossing matching exists.

Note that in case (2) there is no choice but to reduce. The matching edge $e_j$ that is finally chosen will block any candidate $e_i$ that is outside the newly reduced $\Delta_i$. In case (3) there are two combinatorially valid placements for $e_i, e_j$, with respect to the positions of $p_i, p_j$. There is no reason to choose arbitrarily before verifying that neither triangle will be reduced further.

![Figure 9](image)

**Figure 9:** Left: $\Delta_i^o$ is reduced to $\Delta_i$ (case 2). Middle: inverted triangles – no immediate reduction is possible (case 3). Right: no solution exists (case 4).

### 4.2. Properties of a reduced set of triangles

Here we describe certain properties that must hold after we exhaustively apply our reduction rule to a set of triangles.

Let two (three) pairwise inverted triangles be called an inverted pair (triple). An inverted triple is shown in Figure 10. Consider an inverted triple $\Delta_0, \Delta_1, \Delta_2$. The clockwise radial ordering of the triangles $\Delta_0, \Delta_1, \Delta_2$ with respect to a point $p$ is the circular ordering by angle around $p$ of the
bases of these triangles that are visible to \( p \). Note that since the bases of \( \Delta_0, \Delta_1, \Delta_2 \) do not cross (input segments are non-intersecting), then every triangle base appears exactly once in this ordering.

**Figure 10:** An inverted triple \((\Delta_1, \Delta_2, \Delta_3)\).

**Lemma 1.** Let \((\Delta_1, \Delta_2, \Delta_3)\) be an inverted triple, and let \( p_i \) be the apex of \( \Delta_i \) for \( i = 1, 2, 3 \). Then the clockwise order of \( p_1, p_2 \) and \( p_3 \) along their convex hull is identical to the clockwise radial order of \( \Delta_1, \Delta_2, \Delta_3 \) from any of the points \( p_1, p_2, \) or \( p_3 \).

**Proof.** Let \( c \) be the barycenter of \( p_1, p_2 \) and \( p_3 \), and consider the oriented line \( \ell \) through \( c \) rotating clockwise. Note that \( c \) is in the intersection of \( \Delta_1, \Delta_2 \) and \( \Delta_3 \). When \( \ell \) is incident to \( p_i \) with \( p_ic \) in the positive orientation of \( \ell \), the positive halfline of \( \ell \) from \( p_i \) intersects the base of \( \Delta_i \) (because \( c \in \Delta_i \)). Thus, this halfline visits the points \( p_i \) in the same clockwise order as the bases of the triangles \( \Delta_i \). \( \square \)

**Lemma 2.** Let \((\Delta_1, \Delta_2)\) and \((\Delta_1, \Delta_3)\) be two inverted pairs. If a solution exists, then applying the reduction rules to \( \Delta_1, \Delta_2, \Delta_3 \) will result in either \((\Delta_1, \Delta_2, \Delta_3)\) forming an inverted triple or becoming disjoint.

**Proof.** If \( \Delta_3 \) is also inverted with \( \Delta_2 \), then we have an inverted triple. Otherwise, note that \( \Delta_2 \) and \( \Delta_3 \) cannot be disjoint, since they both contain the apex of \( \Delta_1 \). Assuming case (4) does not apply to \( \Delta_2 \) and \( \Delta_3 \) (in which case no solution would exist), we are left with case (2). Assume without loss of generality that \( \Delta_2 \) contains \( p_3 \) and \( \Delta_3 \) does not contain \( p_2 \) (see Figure 11). Then \( \Delta_2 \) is reduced by \( \Delta_3 \), which implies that it is no longer inverted with \( \Delta_1 \). Thus \( \Delta_1 \) gets reduced, and then so does \( \Delta_3 \). All triangles end up disjoint. \( \square \)

**Lemma 3.** Let \((\Delta_1, \Delta_2, \Delta_3)\) be an inverted triple. If \( \Delta_1 \) also has the inverted property with some triangle \( \Delta_4 \), then if we apply the reduction rule to
\( \Delta_1, \Delta_2, \Delta_3, \Delta_4, \) either all four triangles become disjoint or no non-crossing matching exists.

**Proof.** Assuming a non-crossing matching exists, then for all \( i, j \in \{1, 2, 3\} \) and by Lemma 2, either \((\Delta_i, \Delta_j, \Delta_4)\) becomes an inverted triple, or \(\Delta_i, \Delta_j, \) and \(\Delta_4\) become disjoint after applying the reduction rule. The latter case implies, again by the same lemma, that all four triangles would be disjoint.

Otherwise, in the former case, every triple from \(\Delta_1, \Delta_2, \Delta_3, \Delta_4\) is inverted, and so by Lemma 1, every triple in \(p_1, p_2, p_3, p_4\) has the same clockwise orientation as the corresponding bases. This implies that \(p_1, p_2, p_3, p_4\) form a convex quadrilateral \(Q\). The angle of \(\Delta_i\) at \(p_i\) is larger than the interior angle of \(Q\) at \(p_i\) since \(\Delta_i\) contains the 3 other points. Therefore, the sum of the angles of \(\Delta_i\) at \(p_i\) is at least \(2\pi\). Let \(c\) be the barycenter of \(p_1, p_2, p_3, p_4\). The angle from \(c\) to the base of \(\Delta_i\) is strictly larger than the angle of \(\Delta_i\) at \(p_i\). The sum of the angles from \(c\) to the bases of \(\Delta_i\) is at most \(2\pi\) because these wedges from \(c\) do not overlap. Therefore, the sum of the angles of \(\Delta_i\) at \(p_i\) is strictly less than \(2\pi\), a contradiction. \(\square\)

Let a *unit* be a (possibly reduced) triangle, an inverted pair, or an inverted triple. Any time a triangle intersects a unit, the unit will be unaffected, or reduced according to Lemmas 3 and 2, or be “upgraded” to an inverted pair or triple (if it was a triangle or inverted pair, respectively). Two units are said to be *disjoint* if their interiors do not overlap. This establishes that units are the only possible structures that can remain after applying all possible deterministic reductions, if the decision problem has a positive answer. There can be an arbitrary number of any types of units in the final configuration.
Lemma 4. When a triangle $\Delta_p$ is added to a set of $n$ disjoint units not already containing $\Delta_p$, after possibly applying reductions we obtain a new set of disjoint units. This new set is either disjoint to $\Delta_p$, or $\Delta_p$ joins exactly one member of the set and becomes disjoint to all others. Furthermore, adding $\Delta_p$ to the existing disjoint units and applying (possible) reductions takes $O(n)$ time in the worst case.

Proof. Any triangle in a unit will be unaffected or reduced by interacting with $\Delta_p$, so the new set will end up disjoint. Since units are disjoint, $p$ can be inside at most one unit $u$. This means that for all triangles not belonging to $u$, the interaction of $\Delta_p$ will lead to case (4), or cause no change, or cause a reduction of $\Delta_p$. Furthermore, $\Delta_p$ will become disjoint to all such triangles.

Now consider the interaction between $\Delta_p$ and $u$. If $u$ is a triangle, we either get a reduction of $u$ or we obtain a new inverted pair. If $u$ is an inverted pair, by Lemma 2 we either obtain case (4), or an inverted triple, or three disjoint triangles, or $\Delta_p$ reduces $u$ (case (2)) without destroying the inverted pair of $u$. If $u$ is an inverted triple, by Lemma 3 we either obtain case (4), or we get four disjoint triangles, or $\Delta_p$ reduces $u$ (case (2)) without destroying the inverted triple of $u$.

In all cases, $\Delta_p$ either becomes part of an inverted unit or is left disjoint to all triangles. Since we only compare $\Delta_p$ to every triangle in the set, this procedure takes linear time. \qed

4.3. Algorithm

Theorem 6. Given an ordered set $P$ of points inside a convex polygon having an ordered set $T$ of line segments as edges, deciding whether $(P,T)$ admits a non-crossing matching can be done in $O(n \log^2 n)$ time.

Proof. We provide an algorithm where we employ a divide-and-conquer technique. Suppose that we have solved the problem separately on two consecutive convex chains (we can transform a chain into a polygon by adding 3 fake edges and points; thus, solving the problem on a chain is equivalent to solving the polygonal version).

We claim that we can merge the two solutions in $O(n \log n)$ time. Each solution is a set of disjoint triangles and inverted pairs or triples. Refer to Figure 12.
Let $A$ and $B$ be two solved sub-problems of size $k$. We construct a standard point-location data structure\(^5\) on each in $O(k)$ time \([27]\). Now, for every point $p_i$ in $B$, we locate $p_i$ in $A$ to determine if it is inside a unit in $A$. Note that $p_i$ can be in at most one unit. If it is, we determine if $\Delta_i$ reduces this unit by case (2). Likewise, for every point $p_j$ in $A$, we locate $p_j$ in $B$ to determine if it is inside a unit in $B$ and apply the appropriate reductions. Note that if at some moment $\Delta_i$ (belonging to $B$) gets reduced, this will not affect its corresponding unit in $A$; the same holds for all $\Delta_j$ in $A$ that get reduced.

Of course, it is possible that $\Delta_i$ will be inverted with a triangle in $A$. In this case we simply determine if there are reductions and, if applicable, we merge the two units. Therefore a constant number of reductions are applied per point, which means we spend $O(\log k)$ time per point for the point-location step.

The only unresolved issue is to detect if case (4) will occur between triangles of $A$ and $B$ (see the right diagram in Figure 12). For this we can use the Bentley-Ottmann line segment intersection algorithm and stop as soon as a bad intersection is found \([8]\). Given that all triangles have been reduced

---

\(^5\)To construct their point-location data structure, Kirkpatrick et al. \([27]\) triangulate each subdivision in $O(k \log k)$ time, and hence their algorithm requires $O(k \log k)$ time in the worst case. Using Chazelle’s linear-time triangulation algorithm \([12]\), we can reduce this running time to $O(n)$.
and merged into units, essentially we are verifying that no segments intersect. For \( k \) segments, such queries take \( O(k \log k + h \log k) \) time, where \( h \) is the number of intersections reported. As we stop as soon as we report an intersection, \( h = 1 \) and hence the total time is \( O(\log k) \) per point. Therefore, our merge procedure takes \( O(k \log k) \) time. By a simple recurrence analysis, we determine that the entire algorithm takes \( O(n \log^2 n) \) time.

The algorithm described in the proof of Theorem 6 either decides that no solution exists, or otherwise produces a final set of reduced triangles that represents all valid solutions to the problem. In the latter case, every resulting unit is disjoint and thus independent of all others. So in each triangle we can easily pick the shortest joining segment, and in each inverted pair/triple, we try out the two possible choices and take the best matching. Therefore, after the algorithm finds a solution, the min-max and min-sum optimization problems can be solved in linear time.

5. Matching points with segments on a line

As another special case of matching points to line segments, we now consider the case when the input line segments belong to one single line \( L \). Throughout this section we will assume, without loss of generality, that \( L \) is horizontal. As no matching edge will cross over \( L \), our problem is split into two disjoint sub-problems, and we focus on points above \( L \).

We consider two cases, depending on whether the segments are disjoint or not.

5.1. Matching points with disjoint segments on a line

**Theorem 7.** Given an ordered set \( P \) of points above a horizontal line \( L \) and an ordered set \( T \) of disjoint line segments belonging to \( L \) sorted in order of smallest \( x \)-coordinate, deciding whether \((P,T)\) admits a non-crossing matching can be done in linear time. In the affirmative, the matching that minimizes either the sum of the lengths of the edges or the maximum edge length can be found within the same time bound.

**Proof.** We denote by \([a_i,b_i]\) the interval corresponding to segment \( t_i \), for \( i = 1, \ldots, n \). Since the intervals are given in sorted order, we have \( a_1 \leq b_1 < a_2 \leq b_2 < \cdots < a_n \leq b_n \).
If \((P,T)\) admits some non-crossing matching \(\{p_1m_1, p_2m_2, \ldots, p_nm_n\}\), where \(a_i \leq m_i \leq b_i\) for all \(i = 1, 2, \ldots, n\), we can always slide the point \(m_i\) inside \(t_i\) to a position \(m_i^L\) as far to the left as possible (see Figure 13). This gives the unique \textit{leftmost non-crossing matching} for \((P,T)\), \(\{p_1m_1^L, p_2m_2^L, \ldots, p_nm_n^L\}\). Notice that either \(m_i^L = a_i\), or \(p_i\) and \(m_i^L\) are collinear with some \(p_j\) with \(j < i\).

Next we describe an algorithm for finding the leftmost non-crossing matching, if it exists. The algorithm considers points in a sequential greedy fashion, in the left-to-right order of the corresponding segments.

For \(p_1\), the leftmost matching is simply given by the segment \(p_1a_1\). We then consider the rays from the endpoints of this segment in the direction of the negative semiaxis of abscissae; their points at infinity can be symbolically described as \(q_0 = (-\infty, 0)\) and \(q_1 = (-\infty, y(p_1))\).

The \textit{forbidden region} is the (unbounded) region enclosed by an alternating sequence of horizontal line segments and subsegments of matched edges (see Figure 14). This region is updated at every step of the algorithm. Initially, it is described clockwise by its vertices, namely \(q_1p_1a_1q_0\). Observe that if \(p_2\) is inside the forbidden region, then a non-crossing matching \((P,T)\) would be impossible. If \(p_2\) is outside the forbidden region, a matching is possible if and only if there is some point \(m_2\) in the interval \(a_2b_2\) such that the segment \(p_2m_2\) does not cross the forbidden region. In the affirmative, we slide \(m_2\) to its leftmost possible position, and shoot a ray from \(p_2\) in the direction of the negative semiaxis of abscissae, which may go to infinity, or stop by hitting the segment \(p_1a_1\). The forbidden region is updated in each case, and is always defined by alternating horizontal edges with portions of segments from the matching. See Figure 14.

Assume that, in a generic step, we have obtained the leftmost matching \(\{p_1m_1^L, p_2m_2^L, \ldots, p_{j-1}m_{j-1}^L\}\) and we are processing \(p_j\). Let \(q_i, p_i, q_{i+1}, p_{i+1}, \ldots, q_k, p_k, m_k^L, q_0\) be the current forbidden region (refer to Figure 14). Observe that if
there is some $m_j \in [a_j, b_j]$ such that the segment $p_jm_j$ can be added to the
edges found so far, getting a non-crossing matching, the segment $p_jb_j$ is also
valid. We show next how to check the validity of $p_jb_j$.

We first check the $y$ coordinates of the points $m_{ik}, p_{ik}, p_{ik-1}, \ldots$, which form
an increasing sequence, until we find that $y(p_i) \geq y(p_j) \geq y(p_{i+1})$ (the
case in which $y(p_j)$ is a maximum is completely analogous). Then, we check
whether the segment $p_jb_j$ crosses the segments $m_{ik}p_{ik}, q_{ik-1}p_{ik-1}, \ldots, q_{it-1}p_{it-1}$.

In the affirmative, the algorithm is over, as no crossing-free matching is
possible. Otherwise, the segment $p_jb_j$ is valid. We slide the point matched
with $p_j$ as much to the left as possible (Figure 15), which can be done by
finding the angularly closest point among $p_{i+1}, p_{i+2}, \ldots, p_{ik}, a_j$.

If we shoot a ray from $p_j$ in the direction of the negative semiaxis of
abscissae, we hit the boundary of the forbidden region in a point $q_j$, possibly
at infinity, and the forbidden region is updated to be $q_0, p_i, q_{i+1}p_{i+2}, \ldots, p_ip_jm_jq_0$.

The cost of the step for $p_j$ is proportional to the size of the forbidden
polygonal region that disappears, and that will never be processed again.
Therefore, the amortized cost of one step is constant and the global cost
of the algorithm is $O(n)$. At the end we obtain the leftmost matching \{\ p_1m_1, p_2m_2, \ldots, p_nm_n \}, unless no matching is possible.

If \((P,T)\) admits a non-crossing matching, with a symmetric algorithm we can obtain the rightmost matching \{\ p_1m_1^R, p_2m_2^R, \ldots, p_nm_n^R \}. Then any points \(m_i\) in the intervals \([m_i^L, m_i^R]\) provide a non-crossing matching \{\ p_1m_1, p_2m_2, \ldots, p_nm_n \}. In particular, in each interval \([m_i^L, m_i^R]\) we can pick the matching point \(m_i\) which is closest to \(p_i\), and hence obtain the matching that minimizes the sum of the lengths of the edges or the maximum edge length in additional \(O(n)\) time.

\[Observation.\] If the input disjoint segments \(t_1, \ldots, t_n\) are not given in sorted order along the line, then, we can always sort them in \(O(n \log n)\) time as a preprocessing step. An \(\Omega(n \log n)\) lower bound holds for this problem of matching points with disjoint unsorted segments on a line, by reduction from the problem of integer uniqueness, which is known to have an \(\Omega(n \log n)\) lower bound in the algebraic decision tree model of computation.

Let \(x_1, \ldots, x_n\) be a set of given integers. We associate to them \(2n\) points and \(2n\) segments defining \(p_i = (x_i, 2i), p_i' = (x_i, 2i+1), t_i = [x_i-2/5, x_i-1/5], t_i' = [x_i-4/5, x_i-3/5],\) for \(i = 1, \ldots, n\). Points \(p_i\) and \(p_i'\) are to be matched with the segments \(t_i\) and \(t_i'\), respectively, for \(i = 1, \ldots, n\).

If a number \(x_i\) is unique, then the matching is possible (Figure 16, left). However, if two values are equal, \(x_i = x_j\), then a crossing is unavoidable (Figure 16, right). Therefore, a non-crossing matching exists if and only if the numbers \(x_1, \ldots, x_n\) are all different, which proves the claim.

\[\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure16.png}
\caption{There is a crossing when some integer is repeated.}
\end{figure}\]

5.2. Matching points with arbitrary segments on a line

In this section, we show that when the given segments are confined to a line and possibly intersect, we can determine the existence of a non-crossing matching in polynomial time. The proof first discretizes the problem, and
then uses the same approach as in the proof of Theorem 4 for \( k = O(n^2) \).

**Theorem 8.** Given an ordered set \( P \) of points above a horizontal line \( L \) and an ordered set \( T \) of line segments belonging to \( L \), deciding whether \( (P,T) \) admits a non-crossing matching can be done in \( O(n^8) \) time.

**Proof.** We solve the problem by discretizing it: we transform it into matching the set of points \( P \) with \( O(n^2) \)-tuples, corresponding to all combinatorially distinct matchings for each point.

Consider all lines through every pair of points in \( P \). These lines intersect the horizontal line \( L \). Let \( I \) be the set of all these intersection points and of all the endpoints of segments in \( T \). \( I \) has size \( O(n^2) \), as there are \( 2n \) endpoints and at most \( \binom{n}{2} \) intersections.

\( I \) splits \( L \) into \( O(n^2) \) regions. If any subset \( S \) of the points in \( P \) are matched with an edge incident to one region \( r \) of \( L \), we can pick an arbitrary point \( x \) inside \( r \) and match all the points of \( S \) with an edge incident to \( x \), still preserving the existence of a non-crossing matching. In other words, for each point, there are only \( O(n^2) \) combinatorially different matching edges, and we can thus apply our algorithm of Theorem 4 for matching with \( k \)-tuples. Therefore, the complexity of finding a matching is \( O((n^2)^3n^2 + (n^2)^2n^3) = O(n^8) \). \( \square \)

6. Matching points with lines

In the case where points are matched with lines, it is easy to see that a non-crossing matching always exists: choose an arbitrary direction, not parallel to any line, and project each point on its corresponding line in that direction. Here we show that the optimization problem of minimizing the maximum length over all matching edges in NP-complete. We consider the decision version of the min-max problem.

**Theorem 9.** Given an ordered set \( P \) of points, an ordered set \( T \) of lines, and a number \( y \), deciding whether there exists a non-crossing matching of \( (P,T) \) whose longest edge has length at most \( y \) is NP-complete.

**Proof.** We argue that the problem is in NP. We will first show that only a polynomial number of points along lines of \( T \) need to be considered for a non-crossing matching. We then show how, given a solution based on
these canonical points, we can determine in polynomial time whether the length of each matched edge is at most $y$. We construct the arrangement of lines between all pairs of points among the union of $P$ and the points of intersection of the lines in $T$. We then place a bounding box enclosing the union of $P$ and the points of intersection of the lines in $T$. Together with the bounding box, this arrangement divides each line of $T$ into subsegments, with the property that all points in the relative interior of a subsegment are equivalent non-crossing matching solutions. We can now choose the midpoint and the endpoints of each subsegment, as the canonical points representing possible choices for a non-crossing matching of $(P,T)$. Any matching can be rounded to use only points in $P$ and canonical points of subsegments in $T$, without adding any additional crossings. Therefore a matching can be represented as a combinatorial object on a polynomial number of points. Given a solution with such a representation, we can test in polynomial time whether the matching is non-crossing. Now, we still need to check whether the length of every matched edge is at most $y$. Let $m_i$ be the point on $t_i$ such that the distance between $p_i$ and the subsegment to which $p_i$ is matched is the shortest. For every $p_i$, we can check in polynomial time whether the distance between $p_i$ and $m_i$ is at most $y$. Note that matching $p_i$ to $m_i$ will not introduce any crossings: suppose two points $p_i$ and $p_j$ are matched to the same canonical point in the given solution, and suppose matching $p_i$ to $m_i$ and $p_j$ to $m_j$ will cause the segments $p_im_i$ and $p_jm_j$ to intersect. Then this would imply that segment $p_im_j$ is shorter than $p_im_i$, contradicting the fact that $p_im_i$ is the shortest segment. Therefore, in polynomial time it is possible to check whether a given solution is non-crossing with the distance of every matching edge equal to at most $y$. Hence the problem is in NP.

We reduce from the problem of deciding the existence of a non-crossing matching between a set of points and a set of segments. In Section 3 we proved that this problem is NP-complete. Given an instance $(Q,S)$ of the point-to-segment matching problem, we construct an instance $(P,T)$ of our min-max problem as follows. For each pair $(q_i, s_i)$ in $(Q,S)$, we include the point $q_i$ in $P$ and the line $t_i$ supporting the segment $s_i$ in $T$. We then include a number of pairs $(x,\ell)$, $x \in P$, $\ell \in T$, such that the edge matching $q_i$ with $t_i$ is forced to have its endpoint within the boundaries of $s_i$ in order not to create a long edge between a pair $(x,\ell)$. Thus any non-crossing matching of $(P,T)$ with maximum edge length $y$, when restricted to the pairs $(q_i, t_i)$, will also be a non-crossing matching of $(Q,S)$. The gadget is illustrated in Figure 17.
Figure 17: Illustration of the reduction for the point-to-lines problem.

Let a be one endpoint of the segment $s_i$. We include a point $x$ on the segment $q_i a$, at an arbitrarily small distance $\delta$ from $q_i$. The corresponding line $\ell$ is parallel to $q_i a$, at a distance $\varepsilon$ from $x$. Note that $\ell$ is positioned so that it does not intersect $s_i$. Then, for $\varepsilon$ sufficiently small, an edge connecting $q_i$ to $t_i$ on the right of $a$ will force the edge between $x$ and $\ell$ to be long. More precisely, let $a'$ be the point on $t_i$ such that the angle $\angle a q_i a'$ equals $\theta$ for some small positive value $\theta$. Then if $q_i$ is matched with $t_i$ at any point $p \in aa'$, the length of the edge matching $x$ with $\ell$ can be made arbitrarily close to $\varepsilon / \sin \theta$.

If we fix the value of $\theta$, we can reproduce the gadget at regular angular intervals around $q_i$, covering the whole range of possible edge angles with a constant number of pairs $(x, \ell)$ (see Figure 17). The same construction is used for the other endpoint of $s_i$.

Let $y = \max_i \{d(q_i, s_i) : q_i \in Q, s_i \in S\}$. We choose $\varepsilon$ and $\theta$ such that $\varepsilon / \sin \theta > y > \varepsilon$. If there exists a non-crossing matching for $(Q, S)$, then $q_i$ can be matched to $t_i$ within the boundaries of $s_i$, and every $x$ can be matched to the corresponding $\ell$ using an edge orthogonal to $\ell$, of length $\varepsilon$. Hence every edge has length at most $y$. On the other hand, if no non-crossing matching exists for $(Q, S)$, then a point $q_i \in P$ needs to be matched with $t_i$ outside of $s_i$, and one $(x, \ell)$ gadget is triggered, creating an edge of length $\varepsilon / \sin \theta > y$.

Note that we simultaneously require that $\varepsilon$ be a constant, and $\varepsilon / \sin \theta > y$, hence that $\theta < \arcsin(\varepsilon / y)$. So the value $\max_i \{d(q_i, s_i) : q_i \in Q, s_i \in S\}$ must be bounded by a constant. Also, the gadget pairs $(x, \ell)$ should not interfere with other edges of the matching. Since $\delta$ can be made arbitrarily small, we require the existence of a ball of radius strictly greater than $\varepsilon$, around every point of $Q$, that is never intersected by any edge in a non-crossing matching.
These two conditions (that the largest distance to a segment is bounded, and that there exists an empty ball of constant radius around each point) are satisfied by the hard instances constructed in the reduction of Theorem 5. This concludes the proof.

We observe that if the lines have only a bounded number \( k \) of distinct directions, then there is a simple approximation algorithm for the min-max matching problem. Consider the set of directions (that is, angles with respect to the horizontal axis) of the lines, and find the largest absolute difference between two consecutive angles. Let \( \alpha \) and \( \beta \) be the two consecutive angles maximizing the difference \( \gamma = |\alpha - \beta| \). We have that \( \gamma \geq \pi/k \). If we project the points \( p_i \) on their respective lines at an angle \( (\alpha + \beta)/2 \), then the length of the matching edge between \( p_i \) and \( t_i \) is at most \( 1/\sin(\gamma/2) \) times the distance between \( p_i \) and \( t_i \). Thus this directly yields an approximation factor of \( 1/\sin(\gamma/2) \).

7. Concluding remarks

Non-crossing matchings of points with geometric objects is part of a more general class of problems where non-crossing matchings between geometric sets are considered. In this latter class, the first interesting set of problems is that of matching points with sets of points/segments/lines, as the NP-hardness results for these problems apply to problems of finding non-crossing matchings between sets and sets of many different classes. One example would be all classes of objects that include all segments, such as convex sets.

However, it is still unclear whether our results imply anything about the general problem of finding non-crossing matchings between sets of geometric objects. It is not clear for example that our hardness result for finding a non-crossing matching between points and segments (Theorem 5) has any implication on the problem of finding non-crossing matchings between points and either orthogonal polygons with a fixed number of edges or fat convex objects. Could the algorithm for finding a non-crossing matching for segments in convex position (Theorem 6) be extended to also work for sets of segments that have the same radial ordering about every point \( p_i \)? Also, for matching points with lines, is the problem still NP-complete when the lines have a bounded number of directions? And is the condition of having lines with a bounded number of directions necessary for having an approximation algorithm? These and many similar questions raise interesting open
problems in the study of non-crossing matchings between sets of geometric objects.

References


