New Results on Stabbing Segments with a Polygon

José Miguel Díaz-Báñez[1,⋆,§], Matias Korman[2,⋆⋆,§], Pablo Pérez-Lantero[3,⋆⋆⋆], Alexander Pilz[4,†], Carlos Seara[2,‡,§], and Rodrigo I. Silveira[2,‡,§]

[1] Dept. Matemática Aplicada II, Universidad de Sevilla, Spain
[2] Dept. Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain
[3] Escuela de Ingeniería Civil en Informática, Universidad de Valparaíso, Chile
[4] Institute for Software Technology, Graz University of Technology, Austria

Abstract. We consider a natural variation of the concept of *stabbing* a segment by a simple polygon: a segment is stabbed by a simple polygon $\mathcal{P}$ if at least one of its two endpoints is contained in $\mathcal{P}$. A segment set S is stabbed by $\mathcal{P}$ if every segment of S is stabbed by $\mathcal{P}$. We show that if S is a set of pairwise disjoint segments, the problem of computing the minimum perimeter polygon stabbing S can be solved in polynomial time. We also prove that for general segments the problem is NP-hard. Further, an adaptation of our polynomial-time algorithm solves an open problem posed by Löffler and van Kreveld [Algorithmica 56(2), 236–269 (2010)] about finding a maximum perimeter convex hull for a set of imprecise points modeled as line segments.

1 Introduction

Let S be a set of n straight line segments (*segments* for short) in the plane. The problem of stabbing S with different types of stabbers (in the computer science literature) or transversals (in the mathematics literature) has been widely studied during the last two decades.

Rappaport [14] considered the case in which the stabber is a simple polygon. Specifically, he studied the following problem: a simple polygon $\mathcal{P}$ is a *polygon*

⋆ Partially supported by project MEC MTM2009-08652 and Excellence Project P09-TIC-4840 (Junta de Andalucía).
⋆⋆ With the support of the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the European Union.
⋆⋆⋆ Partially supported by grant CONICYT, FONDECYT/Iniciación 11110069 and MEC MTM2009-08652.
† Recipient of a DOC-fellowship of the Austrian Academy of Sciences at the Institute for Software Technology, Graz University of Technology, Austria.
‡ Partially supported by projects MINECO MTM2012-30951 and Gen. Cat. DGR2009GR1040. R.S. was also funded by FP7 Marie Curie Actions Individual Fellowship PIEF-GA-2009-251235.
§ Partially supported by the ESF EUROCORES program EuroGIGA -ComPoSe IP04-MICINN Project EUI-EURC-2011-4306.

P.G. Spirakis and M. Serna (Eds.): CIAC 2013, LNCS 7878, pp. 146–157, 2013.

transversal of S, if we have $\mathcal{P} \cap s \neq \emptyset$ for all $s \in S$; that is, every segment in S has at least one point in $\mathcal{P}$. A simple polygon $\mathcal{P}$ is a *minimum polygon transversal* of S if $\mathcal{P}$ is a polygon transversal of S and all other transversal polygons have equal or larger perimeter. Rappaport observed that such a polygon always exists, is convex, and may not be unique. He gave an $O(3^m n + n \log n)$ time algorithm for computing one, where m is the number of different segment directions. Several approximation algorithms are known [6,8], but determining if the general problem can be solved in polynomial time is still an intriguing open problem.

Arkin et al. [2] considered a similar problem: S is *stabbable* if there exists a convex polygon whose boundary $\mathcal{C}$ intersects every segment in S; the closed convex chain $\mathcal{C}$ is then called a (convex) *transversal* or *stabber* of S. Note that in this variation there is not always a solution. Arkin et al. [2] proved that deciding whether S is stabbable is NP-hard.

In this paper we also consider the problem of stabbing the set S by a simple polygon, but with a different criterion that is between the two criteria above. More concretely, we use the following definition:

Definition 1. *A segment $s \in S$ is stabbed by a simple polygon $\mathcal{P}$ if* at least *one of the two endpoints of s is contained in $\mathcal{P}$. The set S is stabbed by $\mathcal{P}$ if every segment of S is stabbed by $\mathcal{P}$.*

With this definition we study the Minimum Perimeter Stabbing Polygon (MPSP) problem, defined as finding a simple polygon $\mathcal{P}$ of minimum perimeter that stabs a given set S of segments. The MPSP problem is radically different from the two problems above, those studied by Rappaport [14] and Arkin et al. [2], because for the MPSP only the endpoints of the segments play a role in the solution. Indeed, an alternative way to describe the input to the MPSP problem is by saying that the input are *pairs of points* instead of segments. However, as we will show in this paper, the segments play an important role in establishing the difficulty of the problem, hence we stick to the original definition.

Moreover, the difference with the problem of Rappaport [14] is that in his definition $\mathcal{P}$ can have both endpoints of a segment of $s \in S$ not in $\mathcal{P}$ (provided that the interior of s is stabbed by $\mathcal{P}$), whereas we force one of the endpoints to be in $\mathcal{P}$. One of the common properties of both problems is that the optimal solution is a convex polygon and that it always exists (the convex hull of S is always a stabbing polygon).

On the other hand, a difference with the definition used by Arkin et al. is that in the MPSP problem a segment of S can be fully contained in $\mathcal{P}$, with both endpoints in the interior of $\mathcal{P}$, while this is not allowed in the problem studied by Arkin et al. Therefore, we can say that our problem is *between* the two mentioned ones.

Related Work. Prior to the paper by Rappaport [14], Meijer and Rappaport [12] solved the same problem for a set of n parallel segments in optimal $\Theta(n \log n)$ time. Mukhopadhyay et al. [13] considered a similar problem in which the segments are all vertical, and proposed an $O(n \log n)$ time algorithm to find a minimum-area convex polygon transversal of S. For parallel segments, Goodrich and Snoeyink [7] gave an $O(n \log n)$ time algorithm that decides whether a convex transversal exists.

Several similar problems have been considered in the context of *data imprecision* by Löffler and van Kreveld [10,11]. Their input is a set of imprecise points, where each point is specified by a region in which the point may lie. The output is the smallest and the largest possible convex hulls, measured by perimeter and by area. Among the results obtained in [10], we cite those where regions are segments. For maximum-area convex hulls, the problem can be solved in $O(n^3)$ time if the segments are parallel, or when they are pairwise disjoint with endpoints in convex position. The problem is NP-hard for general segments.

The minimum-perimeter and minimum-area convex hulls problems for parallel segments coincide with the problems studied by Meijer and Rappaport [12] and Mukhopadhyay et al. [13], respectively. Notice also that the setting we consider is in fact a constrained version of the problems studied by Löffler and van Kreveld [10], in which each imprecise point is specified by a pair of points.

Pairs of points are also the input to the problems studied by Arkin et al. [1], who studied the 1-center and 2-center problems for pairs of points. In the former problem, the goal is to find a disk of smallest radius containing at least one point from each pair. The latter one aims at finding two disks of smallest size such that each pair has one point in each disk. Arkin et al. [1] presented algorithms for these problems that run in $O(n^2 \text{polylog}\, n)$ and $O(n^3 \log^2 n)$ time, respectively.

In a more general setting, Daescu et al. [4] studied the complexity of the problem of given a k-colored point set, finding a convex polygon of minimum perimeter containing at least one point from each color. Note that the MPSP problem is the special case in which $2n$ points are colored with n colors and each color is used twice. They proved that their problem is NP-hard if k is part of the input of the problem.

Our Results. We show in Section 2 that if S is a set of pairwise disjoint segments, the MPSP problem for S can be solved in polynomial time. We then show how the algorithm can be adapted to solve the following maximization problem: Select exactly one point on each segment in S such that the perimeter (or area) of the convex hull of the selected points is maximized. This problem was stated as open [10], and is also the solution to the maximization variant of the transversal problem [10]. In Section 3 we show that for general segments the MPSP problem is NP-hard. We complement the NP-hardness by showing that the MPSP problem is Fixed Parameter Tractable (FPT).

Note throughout the paper that optimization on the perimeter requires comparing sums of radicals (specifically, the sum of Euclidean distances). It is not known whether this problem is in NP [3], and therefore the NP-hardness result does not imply NP-completeness for the decision version of the problem. For the same reason, we assume the real RAM as the underlying computational model in our algorithms. Since our algorithms are combinatorial and only the cost function depends on the geometry of the problem instance, the methods in Section 2 are also applicable for optimizing the area (which is in NP).

Due to lack of space, several proofs have been deferred to the full version [5].

2 Solving the Problem for Pairwise Disjoint Segments

In this section we show that if the segments in S are pairwise disjoint, then the MPSP problem can be solved in polynomial time. Given any two points p and q in the plane, let pq denote the segment joining p and q. For any simple polygon $\mathcal{P}$ let $\partial\mathcal{P}$ denote the boundary of $\mathcal{P}$. Consider all possible bitangents of S, i.e., let B be the set of all segments not contained in S spanned by two endpoints of segments in S. Note that the elements of B might cross each other and might also cross the segments in S. A polygon C^* with minimum perimeter that contains at least one endpoint of every segment of S is spanned by endpoints of segments in S, and its edges are elements of B.

Arkin et al. [2] describe a dynamic programming approach to decide whether a set of pairwise disjoint segments admits a convex transversal (the vertices of the transversing polygon are restricted to a given set of candidate points). They use constant-size polygonal chains that separate subproblems and are not crossed by segments; therefore the subproblems are independent. We adapt their approach to produce an algorithm for the MPSP problem. The main difference (apart from the fact that no candidate points are needed) is that segments actually *can* cross the separating chains. However, we show below that they can be handled in a way that leads to polynomial running time. Afterwards, we discuss how to adapt this approach for the maximization variation.

Triangulating a Combination of Segments and a Polygon. The following way of triangulating a combination of segments and a polygon is crucial for the algorithm, and motivates the structure of the subproblems used in the dynamic programming algorithm.

Let $\mathcal{Q}$ be a simple polygon and let S_c be a set of pairwise disjoint segments of which each crosses $\partial\mathcal{Q}$ exactly once. Note that throughout this section we distinguish between a segment *intersecting* (having a point in common) and *crossing* (having an *interior* point in common with) another segment or set. Let X be the interior of $\mathcal{Q}$ and let X' denote the set we get after removing the 1-dimensional domains of S_c from X, i.e., $X' = X \setminus \bigcup_{s \in S_c} s$. Then X' is an open region whose closure is $\mathcal{Q}$. Note that the vertices of X' are the union of: (i) the vertices of $\mathcal{Q}$, (ii) the endpoints of edges in S_c that are in the interior of $\mathcal{Q}$, and (iii) the points where elements of S_c cross $\partial\mathcal{Q}$. Further, note that X' might not be connected if there is a segment of S_c that has one endpoint on $\partial\mathcal{Q}$ and the other one outside $\mathcal{Q}$ (e.g., the longest segment in Fig. 1, left).

We now triangulate X' (i.e., partition it into triangles that are spanned only by vertices of X', see Fig. 1). The triangulation T of X' behaves like the triangulation of a collection of simple polygons (imagine the 1-dimensional parts not in X' where the segments of S_c enter $\mathcal{Q}$, i.e., $X \setminus X'$, to be slightly "split", as in Fig. 1, center). Note that the vertices of T are exactly the vertices of X'. Each edge in T that is not part of $\partial\mathcal{Q}$ or part of a segment in S_c partitions X' into two sets (note that each set need not be connected). We call such edges *chords* (gray edges in Fig. 1, right). Chords are the equivalent of *diagonals* of simple polygons (interior edges that subdivide the polygon into two smaller polygons). Further,

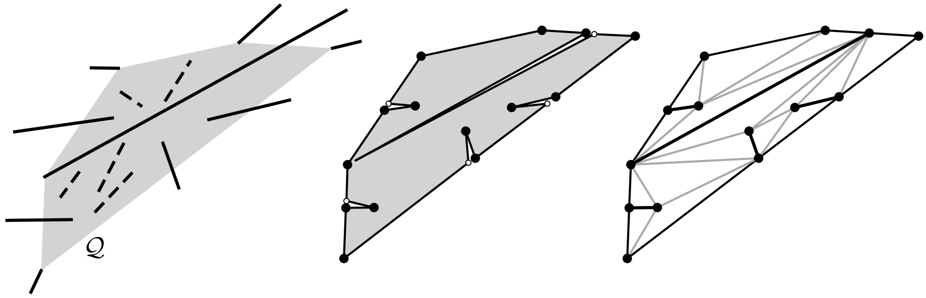

Fig. 1. Left: an optimal polygon $\mathcal{Q}$, only the solid edges are in S_c. Center: schematic view of X' as a collection of simple polygons. Right: a triangulation of X', gray edges are chords. The segments fully contained in the polygon (shown dashed) are ignored by the triangulation.

X' might also be separated by an edge that is part of a segment in S_c (like the longest edge in Fig. 1). We call such a segment a *separating segment*. Keep in mind that there are chords that have one or both of their endpoints not on the endpoint of a segment or at a vertex of $\mathcal{Q}$, but at the crossing of a segment with $\partial\mathcal{Q}$. In any case, a chord or a separating segment defines a polygonal path from one point on an edge of $\mathcal{Q}$ to another point on an edge of $\mathcal{Q}$. Following [2], we will use these polygonal paths of at most three edges, called *bridges*, to define our subproblems to obtain a solution when taking the MPSP C^* as $\mathcal{Q}$. One may think of the approach being similar to the classic dynamic programming algorithm for minimum weight triangulations of simple polygons (see, e.g., [9]), but with a major difference: we do not know the boundary of the triangulated region beforehand.

Subproblems. Every subproblem is defined by an ordered pair (a, b) of directed bitangents of B and a polygonal chain β of at most three edges, the *bridge*, which connects a and b. When evaluating a subproblem (a, b, β), we assume that a and b are edges of C^* and that C^* equals $\mathcal{Q}$ in the discussion above (for some choice of S_c to be defined later). Therefore, the bridge β is part of a triangulation of X' and separates X'; β is either a part of a separating segment or consists of a chord (called the *chord of* β) and at most two parts of segments of S_c. See Fig. 2 for examples of bridges. Note that a bridge might have a chord that is not a bitangent of B (like the second from the left in Fig. 2). Further, note that a bridge can only be crossed by a segment through the chord, since the segments are pairwise disjoint by definition.

Let the directed bitangents be $a = a_1a_2$ and $b = b_1b_2$. Given a directed bitangent $a = a_1a_2$ we write $\overline{a}$ for the directed bitangent a_2a_1. W.l.o.g. let a_1 and b_1 be on the x-axis and a_2 and b_2 be above it. Also, let b be to the left of the directed line through a_1 and a_2. See Fig. 3 for an illustration.

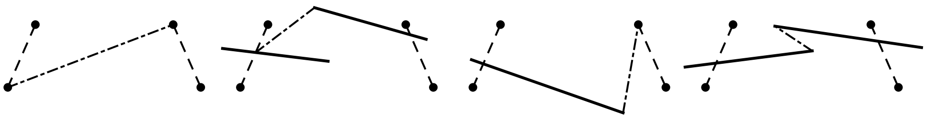

Fig. 2. Examples of bridges. The two bitangents defining the subproblem are shown dashed, chords are dash-dotted, and segments from S_c are shown solid.

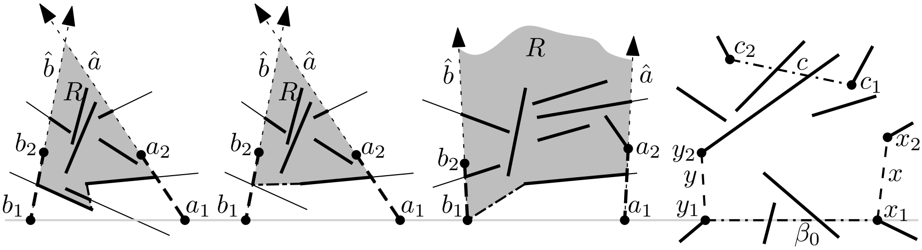

Fig. 3. Examples of subproblems. Rightmost: example for the initial pair.

Solution of a Subproblem. We define the solution of a subproblem as follows. Let $C^*_{a,b,\beta}$ be a polygon of minimum perimeter that: (i) contains a and b as two of its boundary edges, (ii) contains at least one endpoint of each segment in S, and (iii) contains both endpoints of every segment of S that properly crosses the chord of β. The importance of the third condition will become clear later.

Let $C_{a,b,\beta}$ be the polygonal chain on $\partial C^*_{a,b,\beta}$ starting at a_1, counterclockwise traversing $\partial C^*_{a,b,\beta}$ and ending at b_1. Note that $C_{a,b,\beta}$ is an open polygonal chain, as opposed to $C^*_{a,b,\beta}$, which is a simple polygon.

The solution of a subproblem (a, b, β) is $C_{a,b,\beta}$, and its cost is the length of that chain. The base case occurs when $a_2 = b_2$, and has cost equal to the sum of the lengths of a and b. Note throughout the construction that this is the only way a and b can intersect. In general, a and b form a quadrilateral $a_2a_1b_1b_2$. If the quadrilateral is not convex, we discard the subproblem (i.e., we assign it a cost of $+\infty$). The general case where it is convex is discussed next.

Outline of the Algorithm. From now on we assume that a and b define a convex quadrilateral. The outline of the algorithm is as follows. We guess a pair $x, y \in B$ such that $y_2y_1x_1x_2$ are four consecutive vertices of C^*. Hence, after $O(|S|^4)$ guesses we have found x and y such that $\partial C^* = C_{x,y,\beta_0} \cup y_1x_1$ with $\beta_0 = x_1y_1$. Suppose we are given the solution $\mathcal{Q} = C^*$. Let X' be defined as above, and let S_c be the set of segments in S that cross C_{x,y,β_0} (which does not include the ones that cross β_0). Let Δ_0 be the triangle of a triangulation T of X' that has $\beta_0 = y_1x_1$ as one side. The subproblem (x, y, β_0) will be solved by guessing the third endpoint of Δ_0 and the edge c of C_{x,y,β_0} that is incident to Δ_0 or that is crossed by a segment whose endpoint is incident to Δ_0. In the most general case, this gives two new subproblems $(x, \overline{c}, \beta_1)$ and (c, y, β_2), where each of β_1 and β_2 contains one side of Δ_0 that is not part of β_0 (we will consider the other cases in detail below). See Fig. 3, right.

Let $\hat{a}$ be the ray through a_2 starting at a_1. Let $\hat{b}$ be defined analogously. For every subproblem (a, b, β), only a part of the elements of S is relevant. Consider the (possibly unbounded) maximal region to the left of a and to the right of b (recall that a and b are directed). The bridge β disconnects that region into two parts. The *subproblem region* $R_{a,b,\beta}$ is the part "above" β (i.e., the part adjacent to $\hat{a} \setminus a$ and $\hat{b} \setminus b$; the bridge might not be x-monotone).

The subproblem region is marked gray in Fig. 3. Only the segments that have at least one endpoint in $R_{a,b,\beta}$ are relevant for finding $C_{a,b,\beta}$. We distinguish between three different types of such segments: (1) Segments that are entirely inside $R_{a,b,\beta}$ are *complete*. (2) Segments that share more than one point with $R_{a,b,\beta}$ but are not complete are *cut*. (3) A segment with infinitely many points on the bridge is neither cut nor complete. We say that a point is *inside* $C_{a,b,\beta}$ when it is contained in the closure of the region bounded by $C_{a,b,\beta}$ and β.

If there is a segment that is entirely to the right of a or to the left of b, then the choice of a and b cannot give a solution and such a subproblem is assigned $+\infty$ as cost. We also do this if a segment intersected by $\hat{a}$ or $\hat{b}$ does not have an endpoint inside the subproblem region.

Note that if a segment in a valid subproblem intersects $\hat{a}$ or $\hat{b}$, then we know which of its endpoints must be inside $C_{a,b,\beta}$, while we do not know that for the cut segments that intersect the chord of the bridge. However, we will choose our subproblems in a way such that all endpoints of cut segments in the subproblem region will be inside $C_{a,b,\beta}$; the reason for that will become clear in the proof of Lemma 3, but the reader should keep this in mind as an essential part of the method. For complete segments, we need to decide which endpoint to select.

Lemma 1. *Given a subproblem instance (a, b, β), let t be the chord of β, or its only edge if β is a single edge (which may be a chord itself, or part of a separating segment). Let X be the region bounded by $C_{a,b,\beta} \cup \beta$, and let $X' = X \setminus \bigcup_{s \in S_c} s$, for S_c the set of segments of S that are crossed by chain $C_{a,b,\beta}$. Then either t is an edge of $C_{a,b,\beta}$, or there exists a triangle Δ such that:*

1. *The interior of Δ is completely contained in X'.*
2. *The edge t is an edge of Δ.*
3. *The apex of Δ (i.e., the vertex not on t) is either (i) an endpoint of a segment in S_c inside X, (ii) an endpoint of a segment in S that is a vertex of $C_{a,b,\beta}$, or (iii) an intersection point between a segment in S_c and $C_{a,b,\beta}$.*

Proof. Arbitrarily triangulate X'. If t is not on the boundary, then the triangle Δ incident to t inside the subproblem region fulfills the properties. See Fig. 4. □

Lemma 2. *Let Δ be the triangle of Lemma 1. Any segment of S that has a non-empty intersection with the interior of Δ either has both its endpoints inside $C_{a,b,\beta}$ or crosses t; in the latter case the endpoint that is inside $R_{a,b,\beta}$ is also inside $C_{a,b,\beta}$.*

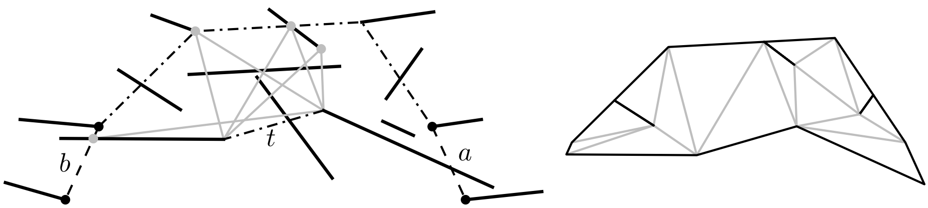

Fig. 4. Illustration of Lemma 1. Left: four possibilities for Δ shown in gray. $C_{a,b,\beta}$ is dash-dotted, with the defining bitangents dashed. Right: a triangulation of X'.

Proof. This follows from the properties of Δ in Lemma 1: A segment intersecting the interior of Δ is not part of S_{c} but has a non-empty intersection with X. Therefore, either both of its endpoints are inside $C_{a,b,\beta}$, or it enters X via t and therefore has its relevant endpoint inside $C_{a,b,\beta}$ by definition. See Fig. 4. $\square$

Getting Smaller Subproblems. Let A be the set of points that are either endpoints of S or crossing points of a segment and a bitangent (recall that no segment of S is an element of B). Hence, A contains all the points that are possible apices for a triangle Δ of Lemma 1. Note that one may construct subproblems where every possible apex of Δ is an endpoint of a segment in S_{c}, as well as subproblems where every possible apex is on a point where a segment crosses $C_{a,b,\beta}$. Further, note that $|A| \in O(|S|^3)$ since $|B| = 4\binom{|S|}{2}$.

Consider again the subproblem (a, b, β). As in Lemma 1, let t be the chord of β if a chord exists, or let t otherwise be the only edge of β. Let a_β be the intersection point of a with the bridge β; b_β is defined analogously. For each subproblem (a, b, β) that is not a base case (i.e., $a_2 \neq b_2$), one of the following cases applies, allowing to get one or two smaller subproblems. During the execution of the algorithm we will consider both cases.

Case 1: t is an Edge of the Solution, i.e., an Edge of $C_{a,b,\beta}$. This happens when t is a chord that does not intersect the interior of the quadrilateral defined by a and b. This case is only valid if no segment crosses t, as we require all the endpoints in $R_{a,b,\beta}$ of segments crossing t to be inside $C_{a,b,\beta}$. In that case we get at most two new subproblems $(a, \overline{t}, \beta_1)$ and (t, b, β_2), where β_1 is the edge $a_\beta t_1$ and β_2 is the edge $t_2 b_\beta$. However, note that one of $(a, \overline{t})$ or (t, b) (or both) might intersect at a_2 or b_2, respectively, and therefore form a base case.

Case 2: t is Not an Edge of the Solution. Then there is a triangle adjacent to t as in Lemma 1. We will guess the apex of the triangle. For every point d in $A \cap R_{a,b,\beta}$ consider the triangle Δ_d that d forms with t. We only consider d if Δ_d is completely inside $R_{a,b,\beta}$, and where the interior of Δ_d does not intersect any segment that intersects a or b. It follows from Lemma 1 that one of the triangles tested leads to a subdivision of the optimal solution. We get the following two subcases, see Fig. 5.

Case 2.1: d is a Point Where a Bitangent and a Segment Cross. Let c be the bitangent that contains d. If c equals a or b, then we get one new subproblem (a, b, β'), with β' containing a side of Δ_d as a chord (Fig. 5a). Otherwise, we get

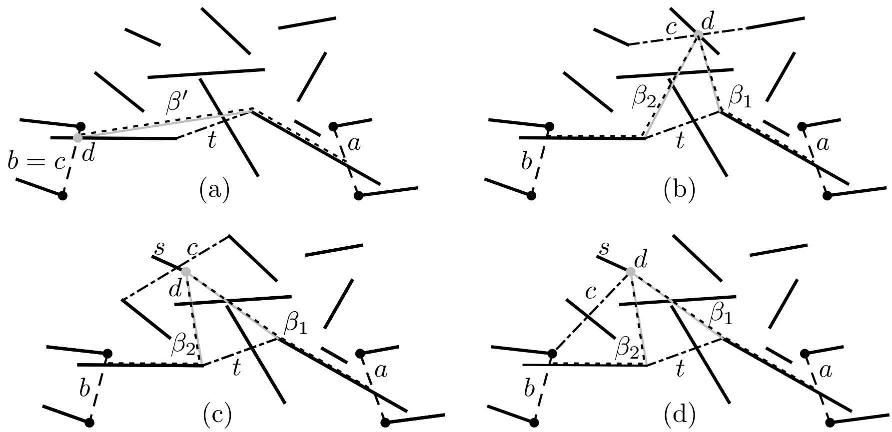

Fig. 5. Case 2. The new bridges are dotted. (a)-(b) Case 2.1. (c)-(d) Case 2.2.

two new subproblems, $(a, \overline{c}, \beta_1)$ and (c, b, β_2), where β_1 and β_2 both contain a side of Δ_d (Fig. 5b).

Case 2.2: d is an Endpoint of a Segment. Let s be the segment that has d as its endpoint. Choose a point x where s intersects some bitangent c. Then, for every possible choice of x (which implies the choices of c), we get two new subproblems $(a, \overline{c}, \beta_1)$ and (c, b, β_2), as in the previous case; note that for both new bridges, $x = d$ is possible. The degenerate case where c equals a or b can be handled as in the previous case. See Fig. 5c-d.

Lemma 3. *Given any valid subproblem (a, b, β), there is a pair of subproblems among the ones above such that the union of their solutions is equal to $C_{a,b,\beta}$.*

Proof. Consider the edge t of Lemma 1. If t is a chord and part of $C_{a,b,\beta}$, then it will be considered in Case 1. Otherwise, consider the triangle Δ inside $C_{a,b,\beta}$. All segments that are intersected by the interior of Δ are either completely contained in $C_{a,b,\beta}$ or enter through t (if it is a chord) and therefore have their relevant endpoint inside $C_{a,b,\beta}$ (cf. Lemma 2). Hence, when the choice of Δ_d coincides with Δ, the two subproblems can be combined into $C_{a,b,\beta}$; the only segments that are part of both subproblems intersect the interior of Δ, and we know that both endpoints will have to be inside the chain that results from the combination of the solutions of the subproblems. Since all possibilities of Δ_d are checked, the subproblem combination of minimum cost is guaranteed to be $C_{a,b,\beta}$. □

This last lemma now implies that we actually find the optimal solution. Note that it is easy to construct a pair of bitangents and a bridge (a, b, β) that is part of the optimal solution but for which $C_{a,b,\beta}$ is not part of C^*. However, as mentioned in the outline of the algorithm, we choose the initial problem (x, y, β_0) in a way that $\partial C^* = C_{x,y,\beta_0} \cup \beta_0$. All segments crossing $\beta_0 = x_1 y_1$ need to have their endpoint above β_0 inside the solution, and the algorithm actually produces a triangulation of X' when taking C^* as $\mathcal{Q}$ and S_c being the segments that cross ∂C^* but do not cross β_0.

Recall that we initialize the algorithm using a brute-force approach: that is, we consider all $O(|S|^4)$ possible choices for two defining bitangents and a bridge a_1b_1. Every subproblem contains less edges of the complete graph on all endpoints of S, and for every subproblem we need polynomial time. The number of subproblems can be bounded by the choices for c and d. Therefore, dynamic programming can be applied to obtain a polynomial-time algorithm.[1]

Theorem 1. *Given a set S of pairwise disjoint segments, a Minimum Perimeter Stabbing Polygon (MPSP)—i.e. a minimum perimeter polygon containing at least one endpoint of each segment in S—can be computed in polynomial time.*

Maximization for Pairwise Disjoint Segments. Our previous algorithm relies on the fact that the result has minimum perimeter: this automatically prevents two endpoints of the same segment from being vertices of the resulting polygon. However, making the algorithm slightly more sophisticated, we can solve in polynomial time a maximization version of the problem, stated open by Löffler and van Kreveld [10]: *select exactly one point on each segment in S such that the perimeter (or area) of the convex hull of the selected points is maximized.* This result is based on the fact that for the maximum area or perimeter transversal, one needs to consider only the endpoints of the segments [10, Lemmata 1 and 8]. The proof can be found in the full version [5].

Theorem 2. *There exists a polynomial-time algorithm that selects exactly one point on each segment in S such that the perimeter (or area) of the convex hull of the selected points is maximized over all possible selections.*

3 Hardness of the General Version

In this section we prove that the MPSP problem is NP-hard by reducing 3-SAT to it. Our reduction is similar to the ones used in [2,4,10].

Theorem 3. *The MPSP problem is NP-hard.*

Proof (Sketch). We only present here the main construction, the rest of the proof is given in the full version [5]. Let a 3-SAT instance consist of n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. We reduce this instance to the following one of the MPSP problem. We draw a circle and place variable gadgets in the left semicircle, clause gadgets in the right semicircle, and segment connectors joining variable gadgets with clause gadgets. See Fig. 6a.

For each variable x_i, $i \in [1..n]$, we put points T_i and F_i on the circle and place three segments: segment T_iF_i, and two zero-length segments a_i and b_i, so that

[1] A straightforward analysis of the running time results in $O(|S|^9)$, which probably can be improved. In any case, it is worth stressing that our main contribution is that the problem can be solved in polynomial time, more than the running time itself.

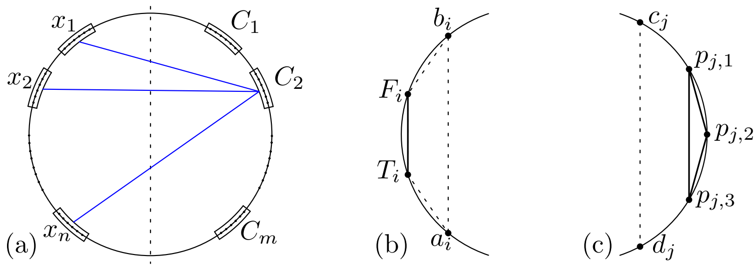

Fig. 6. (a) Overview of the reduction from 3-SAT. Variable gadgets (b) are to the left and clause gadgets (c) to the right.

T_iF_i is parallel to the line containing both a_i and b_i. Refer to Fig. 6b. Furthermore, trapezoids with vertices a_i, T_i, F_i, b_i, for all $i \in [1..n]$, are congruent. Let $P_v := |a_iT_i| + |T_ib_i| = |a_iF_i| + |F_ib_i|$ and $P'_v := |a_iT_i| + |T_iF_i| + |F_ib_i|$ (where $|pq|$ denotes the length of the segment pq).

For each clause C_j, $j \in [1..m]$, we first place two zero-length segments c_j and d_j. We select the three points $p_{j,1}$, $p_{j,2}$, and $p_{j,3}$, dividing evenly the smallest arc of the circle joining c_j and d_j into four arcs, and then we place three other segments: $p_{j,1}p_{j,2}$, $p_{j,2}p_{j,3}$, and $p_{j,3}p_{j,1}$. See Fig. 6c. The convex pentagons with vertices $d_j, c_j, p_{j,1}, p_{j,2}, p_{j,3}$, for all $j \in [1..m]$, are congruent. Let $P_c := |c_jp_{j,1}| + |p_{j,1}p_{j,2}| + |p_{j,2}d_j| = |c_jp_{j,1}| + |p_{j,1}p_{j,3}| + |p_{j,3}d_j| = |c_jp_{j,2}| + |p_{j,2}p_{j,3}| + |p_{j,3}d_j|$ and $P'_c := |c_jp_{j,1}| + |p_{j,1}p_{j,2}| + |p_{j,2}p_{j,3}| + |p_{j,3}d_j|$. We further ensure that $m(P'_c - P_c) < P'_v - P_v$. This condition will be necessary in the problem reduction.

For each clause C_j, $j \in [1..m]$, we add segments called *connectors* as follows. Let x_i be the variable involved in the first literal of C_j. If x_i appears in positive form then we add the segment $T_ip_{j,1}$. Otherwise the segment $F_ip_{j,1}$ is added. We proceed analogously with the variable in the second literal and point $p_{j,2}$, and with the variable in the third literal and point $p_{j,3}$.

Consider the set of segments added at variable gadgets, clause gadgets, and connectors as an instance of the MPSP problem. Observe that any optimal polygon $\mathcal{P}_{opt}$ for this instance satisfies the following conditions:

(a) $\mathcal{P}_{opt}$ contains as vertices points a_i and b_i for all variables x_i, $i \in [1..n]$, and points c_j and d_j for all clauses C_j, $j \in [1..m]$.
(b) For each variable x_i, $i \in [1..n]$, $\mathcal{P}_{opt}$ contains exactly one of T_i and F_i as vertex between a_i and b_i.
(c) In the clause gadget of each clause C_j, $j \in [1..m]$, if the selected endpoint of at least one connector is not in the gadget as a vertex of $\mathcal{P}_{opt}$, then exactly two points among $p_{j,1}$, $p_{j,2}$, and $p_{j,3}$ are vertices of $\mathcal{P}_{opt}$. Otherwise, all three are vertices of $\mathcal{P}_{opt}$.

In the full version [5], we show that any polygon satisfying conditions (a)-(c) induces a valid variable assignment that satisfies the formula if and only if the polygon has minimum perimeter. Further, we give an exact construction for the segment endpoints of the gadgets. □

Observe that the same reduction with minor modifications applies for the case of minimizing the area of the output polygon. Moreover, our proof shows that the problem remains NP-hard even if the endpoints of all the segments are in convex position. On the other hand, the $\sqrt{2}$-approximation algorithm of Daescu et al. [4] gives the same approximation ratio for our MPSP problem.

It is worth mentioning that the MPSP problem is FPT on the number k of segments that intersect other segments. Namely, let $S' \subseteq S$ be the set of segments of S that do not intersect any segment of S. Consider the 2^k instances of the MPSP problem such that each consists of the elements of S' joint with exactly one endpoint (i.e., a segment of length zero) of each element of $S \setminus S'$. All these instances can be solved in $O(2^k P(n))$ time, for the polynomial time $P(n)$ of Theorem 1, since each instance consists of pairwise disjoint segments. The optimal solution for S is among the $O(2^k)$ solutions found for those instances.

References

1. Arkin, E.M., Díaz-Báñez, J.M., Hurtado, F., Kumar, P., Mitchell, J.S.B., Palop, B., Pérez-Lantero, P., Saumell, M., Silveira, R.I.: Bichromatic 2-center of pairs of points. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 25–36. Springer, Heidelberg (2012)
2. Arkin, E.M., Dieckmann, C., Knauer, C., Mitchell, J.S., Polishchuk, V., Schlipf, L., Yang, S.: Convex transversals. Comput. Geom. (2012) (article in press)
3. Blömer, J.: Computing sums of radicals in polynomial time. In: FOCS 1991, pp. 670–677. IEEE Computer Society (1991)
4. Daescu, O., Ju, W., Luo, J.: NP-completeness of spreading colored points. In: Wu, W., Daescu, O. (eds.) COCOA 2010, Part I. LNCS, vol. 6508, pp. 41–50. Springer, Heidelberg (2010)
5. Díaz-Báñez, J.M., Korman, M., Pérez-Lantero, P., Pilz, A., Seara, C., Silveira, R.I.: New results on stabbing segments with a polygon. CoRR abs/1211.1490 (2012)
6. Dumitrescu, A., Jiang, M.: Minimum-perimeter intersecting polygons. Algorithmica 63(3), 602–615 (2012)
7. Goodrich, M.T., Snoeyink, J.: Stabbing parallel segments with a convex polygon. Computer Vision, Graphics, and Image Processing 49(2), 152–170 (1990)
8. Hassanzadeh, F., Rappaport, D.: Approximation algorithms for finding a minimum perimeter polygon intersecting a set of line segments. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 363–374. Springer, Heidelberg (2009)
9. Klincsek, G.T.: Minimal triangulations of polygonal domains. Ann. Discrete Math. 9, 121–123 (1980)
10. Löffler, M., van Kreveld, M.J.: Largest and smallest convex hulls for imprecise points. Algorithmica 56(2), 235–269 (2010)
11. Löffler, M., van Kreveld, M.J.: Largest bounding box, smallest diameter, and related problems on imprecise points. Comput. Geom. 43(4), 419–433 (2010)
12. Meijer, H., Rappaport, D.: Minimum polygon covers of parallel line segments. In: CCCG 1990, pp. 324–327 (1990)
13. Mukhopadhyay, A., Kumar, C., Greene, E., Bhattacharya, B.K.: On intersecting a set of parallel line segments with a convex polygon of minimum area. Inf. Proc. Lett. 105(2), 58–64 (2008)
14. Rappaport, D.: Minimum polygon transversals of line segments. Int. J. Comput. Geometry Appl. 5(3), 243–256 (1995)