Tight Rectilinear Hulls of Simple Polygons

Annika Bonerath\(^1\), Jan-Henrik Haunert\(^1\), and Benjamin Niedermann\(^1\)

\(^1\) University of Bonn
lastname@igg.uni-bonn.de

Abstract

A polygon is called \(C\)-oriented if the orientations of all its edges stem from a pre-defined set \(C\). The schematization of a polygon is then a \(C\)-oriented polygon that describes and simplifies the shape of the input polygon with respect to given hard and soft constraints. We study the case that the \(C\)-oriented polygon needs to contain the input polygon such that it is tight in the sense that it cannot be shrunk without starting to overlap with the input polygon; we call this a tight \(C\)-hull of the polygon. We restrict the tight \(C\)-hull to be a simple polygon. We aim at a tight \(C\)-hull that optimally balances the number of bends, the total edge length and the enclosed area. For the case that both polygons are rectilinear, we present a dynamic-programming approach that yields such a tight hull in polynomial time. For arbitrary simple polygons we can use the same approach to obtain approximate tight rectilinear hulls.

1 Introduction

Schematization has become a common tool for creating simplified visualizations of geometric objects such as paths, networks and regions. The purpose of this technique is to reduce the visual complexity of an object by describing its geometry based on a restricted and pre-defined set \(C\) of orientations. Most prominently, it is used for drawing maps of metro systems [10, 12], in which each edge is drawn either vertically, horizontally or diagonally; those maps became known as octilinear maps. An important core problem is the simplification of a polyline such that the result is \(C\)-oriented, i.e., each edge of the resulting polyline has an orientation that stems from \(C\). Finding \(C\)-oriented paths between two points in a polygon [1, 6, 9] or homotopic \(C\)-oriented paths between obstacles [11] is closely related.

In this paper, we study the schematization of simple polygons, i.e., for a given simple polygon \(P\) we aim for a \(C\)-oriented simple polygon \(Q\) that describes the shape of \(P\) with respect to pre-defined hard and soft constraints. For constructing \(C\)-oriented polygons several approaches have been presented, e.g., [2, 3, 4, 5, 7, 8].

We present a novel approach for schematizing a given simple polygon \(P\) by a \(C\)-oriented simple polygon \(Q\). In contrast to previous work, we construct \(Q\) such that it encloses \(P\). Further, \(Q\) should mimic the shape of \(P\) without having too many bends and without using unnecessarily much space; see Fig. 1. As application we have the schematization of plane graph drawings in mind whose outer faces we want to roughly sketch. We plan to use our approach for travel-time maps visualizing the reachable part within a road network (see Fig. 2) as well as for schematic representations of point sets. In the latter case the idea is to compute a planar graph representing a geometric spanner of the points and then to schematize the graph drawing.

We formalize the constraint that the original polygon \(P\) must be contained in the schematized polygon \(Q\) and mimics the shape of \(P\) in such a way that \(Q\) cannot be shrunk without intersecting \(P\). More specifically, let \(Q\) and \(Q'\) be two simple polygons with edges \(e_1, \ldots, e_n\) and \(e'_1, \ldots, e'_n\), respectively. Further, let \(\vec{v}_1, \ldots, \vec{v}_n\) and \(\vec{v}'_1, \ldots, \vec{v}'_n\) be the vectors that describe the directions and lengths of \(e_1, \ldots, e_n\) and \(e'_1, \ldots, e'_n\), respectively. The

This is an extended abstract of a presentation given at EuroCG’20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more formal form at a conference with formal proceedings and/or in a journal.
Figure 1 A rectilinear polygon $P$ (blue) and a tight rectilinear hull of $P$ (lilac).

Figure 2 A sketch of tight hulls enclosing the road network reachable from $s$ within a given time. The input polygon is the outer face of the reachable sub-graph. We note that we can adapt the definition of tight hulls to also respect the non-reachable part.

A polygon $Q'$ is a linear distortion of $Q$ if there are positive constants $c_1, \ldots, c_n$ such that $\vec{v}_1' = c_1 \cdot \vec{v}_1, \ldots, \vec{v}_n' = c_n \cdot \vec{v}_n$, i.e., each edge of $Q$ can be scaled and translated such that the polygon $Q'$ results; see Fig. 3a. A simple polygon $Q$ is a tight hull of another polygon $P$ if $Q$ contains $P$ and there is no linear distortion of $Q$ that lies in $Q$ and contains $P$. We emphasize that a tight hull has no self-intersections. In case that edges of $Q$ only use orientations from $C$ we call $Q$ a tight $C$-hull of $P$. Altogether, we formalize the schematization problem as finding a tight $C$-hull of $P$. In the special case that $C$ only contains diagonal, vertical and horizontal orientations, we call $Q$ a tight octilinear hull of $P$; see Fig. 3b. If it only contains vertical and horizontal orientations, we call $Q$ a tight rectilinear hull of $P$; see Fig. 3c.

We aim at a tight $C$-hull $Q$ of $P$ that is a good compromise between its edge length, its area and its number of bends, where a vertex is counted as bend if its incident edges have different orientations. More formally, for $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ with $\alpha_i \geq 0$ we define the cost of $Q$ as $\text{cost}(Q) = \alpha_1 \cdot \text{length}(Q) + \alpha_2 \cdot \text{area}(Q) + \alpha_3 \cdot \text{bends}(Q)$, where length($Q$) is the total edge length of $Q$, area($Q$) is the area of $Q$ and bends($Q$) is the number of bends of $Q$. We call a tight $C$-hull $Q$ of $P \alpha$-optimal if for any other tight $C$-hull $Q'$ of $P$ we have $\text{cost}(Q') \geq \text{cost}(Q)$. Throughout the rest of this paper we study the special case in which we aim for a tight rectilinear hull $Q$ of a rectilinear polygon $P$; see Fig. 3c. We use this fairly strong restriction to conduct a proof of concept for schematized tight hulls of polygons. Finally, we sketch how to use the approach for approximate tight hulls of not necessarily rectilinear polygons. We are currently extending our approach to more general settings, e.g., octilinear orientations as well as arbitrary polygons that are schematized.
Figure 3 (a) The polygon $Q$ is a linear distortion of the polygon $P$. For each edge of $Q$ the according scaling factor is shown. (b) $Q$ is a tight octilinear hull of $P$. The polygon $R$ is not a tight hull of $P$, as $Q$ is a linear distortion of $R$ contained in $R$. (c) $Q$ is a tight rectilinear hull of $P$.

Figure 4 Example of a maximally subdivided polygon.

2 Structural Properties of Tight Rectilinear Hulls

Let $P$ be a rectilinear polygon with $n$ vertices and let $Q$ be a tight rectilinear hull of $P$. We call a rectilinear polygon maximally subdivided if for each vertical and horizontal ray emanating from any vertex of $P$ into its exterior the first contact point with $P$ is again a vertex; see Fig. 4. In the remainder, we assume the input polygon $P$ is maximally subdivided.

Lemma 2.1. Every vertex of $Q$ on $P$ is also a vertex of $P$.

In the proof of Lemma 2.1 we assume that there is a vertex $v$ of $Q$ on $P$ that is not a vertex of $P$; see Fig. 5. We show that this contradicts the assumptions that $P$ is maximally subdivided (Fig. 5a) and $Q$ is tight (see Fig. 5b–5c). Thus, Lemma 2.1 shows that we can build the solution based on the vertices of $P$. The next lemma shows that $Q$ lies in the bounding box of $P$. The proof uses similar arguments as the proof of Lemma 2.1.

Lemma 2.2. The bounding box $B$ of $P$ is a tight rectilinear hull and any other tight rectilinear hull of $P$ is contained in $B$.

In the following we describe how any tight rectilinear hull $Q$ can be successively derived from the bounding box $B$. Figuratively, this process can be understood as carving $Q$ out of $B$. More precisely, we obtain $Q$ from $B$ by successively refining the edges of $B$ by replacing them with more and more complex polylines. As basic building block for this replacement procedure we use L-shaped polylines, which we call bridges. More specifically, a rectilinear polyline $B$ is a bridge of $P$ if $B$ starts and ends at vertices of $P$ and $B$ can be partitioned into a prefix and a (possibly empty) suffix such that the edges of the prefix have the same orientation and the edges of the suffix have the same orientation. Hence, each bridge corresponds to a line segment or two incident line segments forming an “L”; see Fig. 6. The region enclosed by $B$ and the polyline of $P$ connecting the same vertices as $B$ is the bag of $B$. We observe that $B$ may consist of multiple regions and have multiple edges with $P$ in common; see Fig. 6c.
Lemma 2.3. Every tight rectilinear hull of $P$ can be partitioned into a sequence of bridges.

The bounding box $B$ of $P$ can be partitioned into four bridges $B_1$, $B_2$, $B_3$ and $B_4$ such that they contain the top-left, top-right, bottom-right and bottom-left vertices of $B$, respectively; see Fig. 7. The starting and end points of the four bridges lie on $Q$ such that they split $Q$ into four polylines $Q_1$, $Q_2$, $Q_3$ and $Q_4$ that are contained in the bags of $B_1$, $B_2$, $B_3$ and $B_4$, respectively. Our approach is based on the idea that each bridge $B_i$ defines a sub-instance $I_i$ that is solved independently from the others. The sub-instance $I_i$ is defined by $B_i$ and its bag; see Fig. 7c.

We now sketch a recursive procedure that creates $Q_i$ from $B_i$. In general we can describe this setting by a bridge $B$ that contains a subpath $H$ of $Q_i$; when the recursion starts we have $B = B_i$ and $H = Q_i$. In the base case of the recursion the bridge $B$ equals $H$. In the general case we recursively describe $H$ by bridges; see Fig. 8. More specifically for $B$ we can find up to three connected bridges $C_1$, $C_2$, and $C_3$ in the bag of $B$ such that the polyline that is defined by these bridges connects the start and end point of $B$. Each bridge $C_j$ forms a geometrically independent instance, i.e., the bridges $C_1$, $C_2$, and $C_3$ have pairwise disjoint bags. Further, the end points of $C_1$, $C_2$, and $C_3$ partition $H$ into three subpaths $H_1$, $H_2$ and $H_3$ that lie in the bags of $C_1$, $C_2$, and $C_3$, respectively. Hence, the three bridges $C_1$, $C_2$, and $C_3$ partition the bag of $B$ into smaller sub-instances defined by $C_1$, $C_2$, and $C_3$ containing the paths $H_1$, $H_2$ and $H_3$, respectively.

This provides us with the possibility of recursively describing $Q_i$; Figure 9 shows the recursion tree $T$ for $B_1$ and $Q_1$ of the polygon presented in Fig. 7. We call $T$ the derivation tree of $B_1$ and $Q_1$ the derivative path of $B_1$. In general we show the following theorem.

Theorem 2.4. For every bridge $B$ and every path $H$ of bridges that is contained in the bag of $B$ and connects the start and end point of $B$, there is a derivation tree $T_B$ such that $H$ is the derivative path of $T_B$. 

---

**Figure 5** Illustration of the proof of Lemma 2.1. (a) At the end point of $e_1$ the polygon $P$ has a vertex. (b)–(c) The edges $e_1$ and $e_3$ can be scaled such that $Q$ shrinks but contains $P$.

**Figure 6** Examples of rectilinear polylines (green) forming bridges of $P$ (blue).
To prove Theorem 2.4 we distinguish nineteen geometrical settings of the bridge $B$ and the path $H$. We use six different methods for the construction of the child nodes $C_1, \ldots, C_k$ with $1 \leq k \leq 3$; see Fig. 10. We can show for each construction that the path $H$ can be split into subpaths $H_1, \ldots, H_k$ so that each $H_j$ with $1 \leq j \leq k$ is contained in the bag of $C_j$. For example, we use Construction M in the case that $B$ and $P$ share more than two vertices; see Fig. 10. In that case, we insert two child nodes for $B$ in $T_B$ containing the bridges $C_1$ and $C_2$, where $C_1$ is the path of $B$ from the beginning to the first shared vertex $u$ with $P$ and $C_2$ contains the remaining part. We show that if we split $H$ at $u$ into subpaths $H_1$ and $H_2$, the path $H_1$ is contained in the bag of $C_1$ and the path $H_2$ is contained in the bag of $C_2$. The Constructions A-E assume that $B$ shares exactly two vertices with $P$, and yield bridges $C_1, \ldots, C_k$ that not only lie on $B$ but also in the interior of the bag of $B$ without crossing $H$.

The constructions are more general than necessary such that they also work for any rectilinear polygon $Q$ that consists of bridges of $P$. We conjecture that when exploiting the tightness only two children per node is sufficient, which later on would lead to an improvement of the running time by a linear factor. However, at latest when generalizing the result to the case that $P$ is not rectilinear, we can show that three children are necessary.

Altogether, due to the construction of the decomposition tree its derivative path $H$ does not intersect itself. In particular, two bridges $B_1$ and $B_3$ that intersect as shown in Fig. 11 can not belong to the same decomposition tree as neither one bag contains the other nor their bags are disjoint.
Figure 9 A recursion tree for the top-left part of the polygon shown in Fig. 7. On each level the bags of the bridges (orange) form geometrically independent sub-instances that are solved independently. Composing the bridges of the child nodes yields a path that connects the starting with the end point of the bridge of the parent node. Collecting the bridges of the leaves in pre-order yields the path $Q_1$ (lilac), which is part of $Q$. 
3 Algorithm for Tight Rectilinear Hulls

We present an algorithm that consists of three steps. In the first step, we build an orthogonal grid \( G \) based on the vertices of \( P \) such that \( G \) lies in the interior of \( B \) and the exterior of \( P \); see Fig. 12a. In the second step, we create the set \( \mathcal{B} \) of all valid bridges based on \( G \) using depth-first searches; see Fig. 12b. In the third step, we compute an \( \alpha \)-optimal tight rectilinear hull \( Q \) of \( P \) as follows. We split the bounding box \( B \) into the four bridges \( B_1, B_2, B_3 \), and \( B_4 \) as described in Section 2. These bridges split \( Q \) into four paths \( Q_i \) contained in the bags of \( B_i \) (with \( 1 \leq i \leq n \)), respectively. We compute each \( Q_i \) by constructing its derivation tree \( T_i \) over \( \mathcal{B} \) using dynamic programming. We finally assemble \( Q_i \) to \( Q \). From a technical point of view we need to take special care about correctly accounting for the bends at the vertices connecting two sub-instances. We prove that the dynamic programming approach, which is the most time consuming part of the algorithm, needs \( O(n^4) \) time and \( O(n^2) \) space.

\[ \textbf{Theorem 3.1.} \] The \( \alpha \)-optimal tight rectilinear hull of a rectilinear polygon \( P \) can be computed in \( O(n^4) \) time and \( O(n^2) \) space.

We observe that our approach is only based on the bridges that we compute using the grid \( G \). On that account a simple approach to support arbitrary simple polygons is discretizing \( P \) by subdividing each edge of \( P \) with additional vertices; see Fig. 13. We then build \( G \) based on the new and old vertices of \( P \). As one can show, the result is a (not necessarily \( \alpha \)-optimal) tight hull of \( P \). Depending on the desired quality, we choose the degree of discretization.

4 Conclusion

We have introduced the concept of tight hulls of polygons. In contrast to previous schematization techniques, we require that the input polygon is contained in the schematization. We
Figure 11 Decompositions of a bridge $B$. (a) The bridge $B$. (b) A decomposition of $B$ into three bridges $B_1$, $B_2$ and $B_3$ such that $B_1$ and $B_3$ intersect. Such decompositions are excluded from the decomposition tree by construction. (c) A valid decomposition tree for $B$.

Figure 12 Step 1 and Step 2 of the algorithm. (a) The grid $G$ in the exterior of $P$ is created based on the vertices of $P$. (b) For each vertex of $P$ all possible bridges to its successors are created.

have undertaken a proof of concept for rectilinear polygons and tight rectilinear hulls sketching a generic algorithm based on a dynamic programming approach. For simple polygons our approach yields approximate tight hulls. We are currently extending the algorithm to tight octilinear hulls as well as to $\alpha$-optimal tight hulls of general simple polygons.
Figure 13 Tight rectilinear hulls of a simple maximal subdivided polygon \( P \) (vertices of \( P \) are black points). (a) Lemma 2.1 is not true any more as \( Q \) has fixed vertices (lilac squares) that are not vertices of \( P \). (b) The tight hull of \( P \) is based on the vertices of \( P \) and additional vertices (lilac squares) subdividing the edges of \( P \).

Acknowledgments. This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy – EXC 2070 – 390732324.

References


8. Wouter Meulemans, André van Renssen, and Bettina Speckmann. Area-preserving subdivision schematization. In Sara Irina Fabrikant, Tumasch Reichenbacher, Marc van Kreveld,
Tight Rectilinear Hulls of Simple Polygons


