Partitioning Polygons via Graph Augmentation

Jan-Henrik Haunert[1,⋆] and Wouter Meulemans[2]

[1] Institute of Computer Science, University of Osnabrück, Germany.
janhhaunert@uni-osnabrueck.de
[2] giCentre, City University London, United Kingdom.
wouter.meulemans@city.ac.uk

Abstract. We study graph augmentation under the dilation criterion. In our case, we consider a plane geometric graph $G = (V, E)$ and a set C of edges. We aim to add to G a minimal number of nonintersecting edges from C to bound the ratio between the graph-based distance and the Euclidean distance for all pairs of vertices described by C. Motivated by the problem of decomposing a polygon into natural subregions, we present an optimal linear-time algorithm for the case that P is a simple polygon and C models an internal triangulation of P. The algorithm admits some straightforward extensions. Most importantly, in pseudopolynomial time, it can approximate a solution of minimum total length or, if C is weighted, compute a solution of minimum total weight. We show that minimizing the total length or the total weight is weakly NP-hard.
Finally, we show how our algorithm can be used for two well-known problems in GIS: generating variable-scale maps and area aggregation.

1 Introduction

Polygons representing geographic objects can contain millions of vertices and thus can be difficult to handle. Often, they consist of multiple regions that are connected only via narrow bottlenecks, such as isthmuses in the case of land or straits in the case of water areas. To ease the handling of such polygons and to identify natural subregions, such as the Iberian Peninsula as a part of Europe, one often seeks a partition of a polygon into multiple smaller polygons of a certain type (e.g., into convex polygons). A triangulation of a polygon is the most common type of a polygon partition, yet often one is interested in larger (non-triangular) subregions. We present new algorithms for partitioning a polygon based on an internal triangulation of it: every output region is the union of a set of triangles of that triangulation. We consider our algorithm a useful tool for shape manipulation and demonstrate its effectiveness on two use cases: the generation of variable-scale maps and the aggregation of areas.

Our basic idea is to consider the polygon partitioning problem as a special *graph augmentation problem*. The vertices and edges of the input polygon P define a geometric graph G, which we augment with a selection of edges from a set C of candidate edges (that is, diagonals of P) to split P into multiple pieces.

⋆ Corresponding author

After the augmentation, the graph shall be well connected. More precisely, for each candidate edge $\{u, v\} \in C$ we require that the *dilation* for u and v in the augmented graph is bounded by a user-set parameter. For any two vertices u, v of a geometric graph G, the dilation (sometimes also called stretch factor or detour factor) is defined as the ratio between the shortest u-v path via G and the Euclidean distance between u and v. By selecting a minimum number of edges from C we obtain a nice decomposition of the input polygon. As an alternative optimization objective we consider minimizing the total weight of the selected edges, assuming that for each edge in C a weight is given as part of the input.

Contributions. We introduce terminology and a general problem definition with three primary variants (unweighted, length-weighted and general weights) in Section 2. We review related work in Section 3. In Section 4 we consider the problem variants for the case that the graph to be augmented is a simple polygon without holes and the edges that can be added are an internal triangulation. We provide an optimal linear-time algorithm for the unweighted case, and present some extensions. We prove that both the general-weights case and the length-weighted case are weakly NP-hard, present a pseudopolynomial-time algorithm for the general-weights case, and show that it can provide a $(1+\varepsilon)$-approximation algorithm for the length-weighted case. We discuss our two use cases in Section 5.

2 Preliminaries

Graphs. Let $G = (V, E)$ denote a graph defined by its vertices V and edges $E \subseteq \{\{u, v\} \mid u, v \in V\}$. We call G a *geometric* graph if every vertex is assigned a position in $\mathbb{R}^2$ and each edge is represented by the line segment connecting its endpoints. A geometric graph is *plane* if vertices have unique positions and no two edges intersect, except at common endpoints.

Dilation. Let $G = (V, E)$ be a geometric graph and $u, v \in V$ be two vertices of G. We denote the Euclidean distance between u and v as $\|u - v\|$; we use $\|e\|$ to denote the length of edge e. The length of the shortest path in G between u and v is denoted by $d_G(u, v)$. We define the (vertex) *dilation* between u and v as $\Delta_G(u, v) = d_G(u, v)/\|u - v\|$; the dilation of the entire graph is $\Delta_G = \max_{u,v \in V, u \neq v} \Delta_G(u, v)$. If G is disconnected, its dilation is infinite.

Problem statement. In this paper, we consider graph augmentation problems, where the augmentation is constrained to a prescribed set of vertex pairs. We call such vertex pairs *candidate edges*. Hence, a *problem instance* comprises

- a plane geometric graph $G = (V, E)$,
- a set $C \subseteq \{\{u, v\} \mid u, v \in V\} \backslash E$ of candidate edges, and
- a real number $\tau \geq 1$.

Consider $S \subseteq C$ to be a subset of the candidate edges. We denote by $G_S = (V, E \cup S)$ the graph obtained by augmenting G with the candidate edges in S. We call a candidate edge $\{u, v\} \in C$ *satisfied* with respect to S if $\Delta_{G_S}(u, v) \leq \tau$. A

simple path in G_S whose length is sufficiently small to prove that $\Delta_{G_S}(u, v) \leq \tau$ is called a *witness* of $\{u, v\}$. Set S is a solution to the problem if all edges in C are satisfied (with respect to S). Note that we ask to satisfy only the pairs specified by the candidate edges; we do not guarantee that the dilation between all vertices is bounded by τ. This is a trade-off that we make to guarantee that solutions exist. In particular, $S = C$ is a solution for any problem instance.

However, we want to find a "good" solution. A primary criterion, in the context of polygon partitioning, is that the edges in S do not intersect each other or existing edges of G. Furthermore, we consider optimizing three different objective functions, resulting in the following problems:

- MinSize: minimize $|S|$.
- MinLength: minimize $\sum_{e \in S} \|e\|$.
- MinWeight: minimize $\sum_{e \in S} w(e)$, given weights $w \colon C \to \mathbb{R}^+$.

In the above, we provide an upper bound on the allowed dilation and minimize the cost (size, length or weight) of the solution. The dual variants instead bound the allowed cost and ask to minimize the dilation. We focus on the stated variants; our algorithms can solve the dual variant by a binary search on τ. This is possible since the problem is monotonic: any solution for τ is also a solution for $\tau' > \tau$, and thus increasing the dilation can only reduce the minimal cost.

3 Related Work

Partitioning. Partitions of polygons into triangles, monotone polygons, or convex polygons are common in the context of GIS [19] and have intensively been studied in computational geometry. For example, for the case that no additional vertices (i.e., *Steiner points*) are allowed, Keil and Snoeying [13] have shown that a simple polygon with n vertices and r reflex vertices can be partitioned into a minimum number of convex polygons in $O(n + r^2 \min\{r^2, n\})$ time. In the case that Steiner points are allowed, the problem can be solved in $O(n + r^3)$ time [5]. For polygons with holes, the problem is NP-hard in both cases [16].

Often motivated by problems in computer vision and pattern recognition, researchers have developed methods for partitioning polygons into "natural and intuitive" [17], "simpler" [8], or "approximately convex" [15] pieces, which need not be convex. However, these methods do not provide any guarantee of optimality with respect to the number of output pieces or a different measure.

Dilation. Algorithmic work involving dilation is motivated mostly by applications in infrastructure design (e.g. road or electricity networks). Much research has been done without planar considerations, e.g. [2]. Considering our use cases, we focus here on results with such planar considerations; see [4] for a survey.

Giannopoulos *et al.* [9] prove that, given a point set Q, computing a graph $G = (Q, E)$ with $\Delta_G \leq 7$ is NP-hard, if $|E|$ is bounded to $O(|Q|)$. They also prove that adding $O(|E|)$ edges to a geometric graph to bound the dilation to 7 is NP-hard. Both claims hold with and without requiring planarity. This supports

the investigation of our variant, where we do not consider satisfying all pairs, but only those provided in a (constrained) candidate set.

Farshi *et al.* [7] show that it is possible to compute, for a given geometric graph, the edge that results in the largest dilation reduction in $O(n^4)$ time. This was later improved by Wulff-Nilsen [20] to $O(n^3 \log n)$ time. Note that repeatedly applying this greedy choice does not yield an optimal result. Aronov *et al.* [1] present algorithms for the following problem: given a point inside a polygon, compute a segment from the point to the boundary of the polygon such that the dilation from the given point to any point on the boundary is minimized.

If we measure dilation via the geodesic distance and only between vertices of which one is contained in a given small set, an FPTAS exists to compute a minimal-dilation triangulation of a simple polygon [14]. Klein *et al.* [14] attribute to folklore that a constrained Delaunay triangulation of a simple polygon has dilation at most $\pi(1+\sqrt{6})/2 < 5.09$. This readily implies that our algorithms—run with τ and using as C the constrained Delaunay triangulation—compute a small set of edges such that *all* vertex pairs have dilation less than 5.09τ (in the geodesic model). A similar result was proven by Bose and Keil [3], stating that a constrained Delaunay triangulation (not necessarily of a polygon) has dilation at most $4\pi\sqrt{3}/9 \approx 2.42$, though only between pairwise visible points.

4 Triangulated Polygons

Here we study the dilation problem restricted to instances where G is a simple polygon P and C is an inner triangulation of P. We denote the resulting problems by MinSizePoly, MinLengthPoly and MinWeightPoly.

We present a linear-time optimal algorithm for MinSizePoly in Section 4.1. In Section 4.2 we show how to deal with any nonintersecting set of internal diagonals as candidate edges; and in Section 4.3 we present a heuristic for dealing with holes. Finally, in Section 4.4 we prove that MinLengthPoly and MinWeightPoly are weakly NP-hard; we present a pseudopolynomial-time algorithm for MinWeightPoly with integer weights and, via rounding, obtain an approximation algorithm for MinLengthPoly.

4.1 Minimizing the number of selected edges

To solve MinSizePoly, we apply a recursive algorithm. Its recursion is structured using a rooted binary tree $\mathcal{T}$ on the edges of P and C. By maintaining three possible subsolutions for each node in $\mathcal{T}$, we show that we compute an optimal subsolution for each node based only on its children in $\mathcal{T}$.

Building a tree. We define a directed binary tree $\mathcal{T}$ with nodes corresponding to the edges $P \cup C$ as follows. First, we pick an arbitrary edge of polygon P as root r. Then, we add the two edges incident to the same unprocessed triangle as children to r and recurse on each child. The result is a tree on the edges and candidate edges, rooted at r; see Fig. 1. If the embedding is given—the cyclic order of candidate edges at each vertex—we can compute $\mathcal{T}$ in $O(n)$ time, where n is the number of polygon edges. Otherwise, $O(n \log n)$ time suffices.

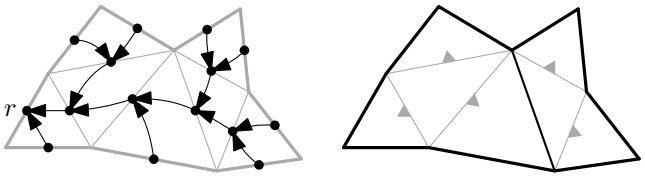


Fig. 1. (left) A binary tree $\mathcal{T}$ with root r. (right) A feasible role assignment for $\tau = 3$; the solid black diagonal is the only selected candidate edge in C, but allows a shorter path for another candidate edge.

Components of $\mathcal{T}$. Every edge $e \in P \cup C$ (a node in the tree) partitions $\mathcal{T}$ into two components[†]. The component that contains r is referred to as $\mathcal{T}_e^{\text{ROOT}}$, the other as $\mathcal{T}_e^{\text{LEAF}}$. Both of these components exclude e itself. For root r we define $\mathcal{T}_r^{\text{ROOT}} = \emptyset$ and $\mathcal{T}_r^{\text{LEAF}} = \mathcal{T} \setminus \{r\}$. For uniformity of presentation, we also define a component $\mathcal{T}_e^{\text{SELF}}$ containing only edge e.

In a solution $S \subseteq C$ for MinSizePoly, each candidate edge $e = \{u, v\} \in C$ must have a witness: a simple u-v path of length at most $\tau\|e\|$. A witness of e lies fully within one of the three components of $\mathcal{T}$ defined by e.

Role assignment. With our algorithm we compute a *role assignment* $\alpha\colon C \to \{\text{SELF}, \text{LEAF}, \text{ROOT}\}$ for all candidate edges. The role assignment indicates which component must contain a witness; we call α *feasible* if $\mathcal{T}_e^{\alpha(e)}$ indeed contains a witness for all $e \in C$. A role assignment α directly prescribes the set S^α of edges that are part of the solution: $S^\alpha = \{e \mid e \in C \wedge \alpha(e) = \text{SELF}\}$. Hence, we refer to $|S^\alpha|$ as the size of α, using $|\alpha|$ as a shorthand. For uniformity, we define $\alpha(e) = \text{SELF}$ for all edges $e \in P$, but these are not part of S^α.

Fig. 1 shows an instance with a role assignment. Every edge $e \in C$ is displayed according to its role: SELF-edges are black; ROOT- and LEAF-edges are gray with a small triangle indicating the direction of their shortest path.

As an edge can play three different roles, there are up to $3^3 = 27$ configurations of a role assignment for a triangle; see Fig. 2. We reduce this to 20 configurations as follows. Consider two edges e_1 and e_2. We call e_1 and e_2 *conflicting* in α if either: e_1 is the parent of e_2 in $\mathcal{T}$, $\alpha(e_1) = \text{LEAF}$ and $\alpha(e_2) = \text{ROOT}$; or e_1 and e_2 are siblings in $\mathcal{T}$ and $\alpha(e_1) = \alpha(e_2) = \text{ROOT}$. The following lemma implies that we may indeed discard the bracketed configurations in Fig. 2.

Lemma 1. *There exists a feasible role assignment with minimal size that does not contain any conflict.*

Proof. Consider a solution S with minimal size. Let α be the role assignment obtained by assigning SELF to $e \in S$ and ROOT or LEAF to the remaining edges, depending on which component of $\mathcal{T}$ contains the shortest path between the endpoints of e. To derive a contradiction, assume α contains a conflict between e_1 and e_2. This implies that $e_2 \in \mathcal{T}_{e_1}^{\alpha(e_1)}$ and vice versa. By construction, the

[†] In this paper "edge" always indicates an element of $P \cup C$—a node in $\mathcal{T}$—and never an edge between nodes (parent-child relation) in $\mathcal{T}$.

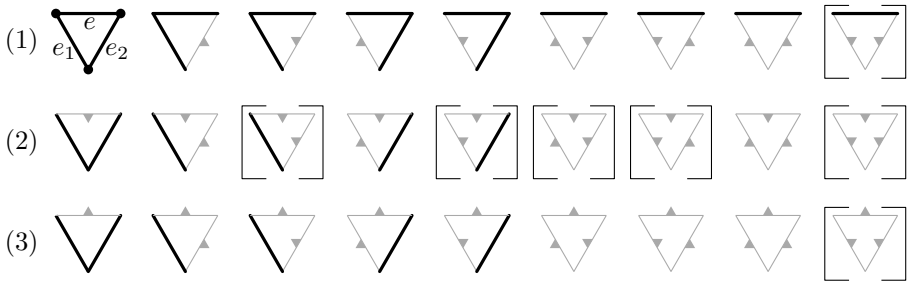


Fig. 2. The 27 configurations of roles for a triangle of an edge e and its children e_1 and e_2 in $\mathcal{T}$. The bracketed roles are not needed for an optimal solution.

shortest path π_1 for e_1 is contained in $\mathcal{T}_{e_1}^{\alpha(e_1)}$. Hence, π_1 must pass through the endpoints of e_2. However, this implies that the shortest path for e_2 is a subpath of π_1, and thus not in $\mathcal{T}_{e_2}^{\alpha(e_2)}$ as this component contains e_1. This is a contradiction, thus α cannot contain a conflict. □

Partial assignments. Our algorithm computes role assignments for subtrees of $\mathcal{T}$. A *partial* role assignment α_e is an assignment on $\{e\} \cup \mathcal{T}_e^{\text{LEAF}}$. Its partial solution S^{α_e} is defined as $\{e' \mid e' \in C \cap (\{e\} \cup \mathcal{T}_e^{\text{LEAF}}) \wedge \alpha_e(e') = \text{SELF}\}$; again we use $|\alpha_e|$ as a shorthand for the size of S^{α_e}. A partial assignment for the root r corresponds to a (full) role assignment. Assignment α_e is feasible if one of the following holds for all $e' \in \{e\} \cup \mathcal{T}_e^{\text{LEAF}}$:

1. $\alpha_e(e') = \text{SELF}$; or
2. $\alpha_e(e') = \text{LEAF}$ and $(S^{\alpha_e} \cup P) \cap \mathcal{T}_{e'}^{\text{LEAF}}$ contains a witness for e'; or
3. $\alpha_e(e') = \text{ROOT}$ and either:
   a) $(S^{\alpha_e} \cup P) \cap \mathcal{T}_{e'}^{\text{ROOT}} \cap (\{e\} \cup \mathcal{T}_e^{\text{LEAF}})$ contains a witness for e';
   b) the combined length of the two shortest paths in $S^{\alpha_e} \cup P$ from the endpoints of e' to the endpoints of e is at most $\tau \cdot \|e'\| - \|e\|$.

The rationale for case 3 is that either the edge is already satisfied (3a) or it is to be satisfied by what has yet to come (3b). However, the latter must ensure that there is still some length "to be spent" in order to complete the solution.

Lemma 1 and the triangle inequality imply that, for a feasible α_e with $\alpha_e(e) \in \{\text{SELF}, \text{LEAF}\}$, all edges in $\{e\} \cup \mathcal{T}_e^{\text{LEAF}}$ are satisfied. It presents a shortest path between the endpoints of e to future computations. The length of this path is the *front-length* of α_e, denoted by $L(\alpha_e)$. Moreover, if $\alpha_e(e) = \text{ROOT}$, then a contiguous subset of $\mathcal{T}_e^{\text{LEAF}}$ may all have this assignment. The *front-allowance* $R(\alpha_e)$ is the maximal allowed length on the root side of e, such that all these assignments are still satisfied. If $\alpha_e(e) \neq \text{ROOT}$, it is infinite.

In the following, all role assignments are feasible, unless mentioned otherwise.

Algorithm. The algorithm relies on a postorder recursive traversal of $\mathcal{T}$ to compute the partial assignment α_e for each edge e. Calling this with r hence results in the full role assignment α. However, to do the recursion correctly, we cannot simply compute a single partial assignment, but compute three instead:

Definition 1. *The following three partial role assignments are defined:*

- α_e^{SELF}*: the smallest partial role assignment with* $\alpha_e(e) = \text{SELF}$*.*
- α_e^{LEAF}*: the partial role assignment with minimal front-length among the smallest partial role assignments with* $\alpha_e(e) = \text{LEAF}$*.*
- α_e^{ROOT}*: the partial role assignment with maximal front-allowance, among the smallest partial role assignments with* $\alpha_e(e) = \text{ROOT}$ *and* $R(\alpha_e) \geq \|e\|$*.*

We compute these assignments based on the partial assignments of the child nodes. The base case, a leaf of $\mathcal{T}$, corresponds precisely to an edge of P. For these, we consider only α_e^{SELF} to be defined, with size 0 and front-length $L(\alpha_e) = \|e\|$. For root r, again corresponding to an edge of P, we are interested only in computing α_r^{SELF}, the size of which (not counting r) is the size of the solution. Any other node of $\mathcal{T}$ is a candidate edge e, with precisely two children in $\mathcal{T}$: e_1 and e_2. To compute the partial assignments in this case, we simply try the 20 cases of Fig. 2 and find those that satisfy Definition 1. By storing the size of the partial assignments, the size of a new partial assignment is simply the sum of the sizes of the children's partial assignments, increased by 1 if e is assigned SELF. However, not all cases may lead to feasible assignments. We therefore check the feasibility as follows, where row numbers refer to the labels in Fig. 2.

Cases in the first row correspond to computing α_e^{SELF}. For cases involving $\alpha_{e_1}^{\text{ROOT}}$ (and analogously for e_2), the front-allowance is met if $L + \|e\| \leq R(\alpha_{e_1}^{\text{ROOT}})$ holds, where L is the front-length provided by sibling.

Cases in the second row correspond to computing α_e^{LEAF}. Cases with a ROOT assignment for a child can be ignored by Lemma 1. We must ensure that the combined front-length of e_1 and e_2 is at most $\tau \cdot \|e\|$.

Cases in the third row correspond to computing α_e^{ROOT}. We check and compute front-allowances. Since e is not part of the solution, a front-allowance of a child is "propagated". For a child with a ROOT assignment, its propagated front-allowance is its front-allowance minus the front-length of its sibling. The minimum of this propagated front-allowance (if any) and $\tau \cdot \|e\|$ is the new front-allowance for e in this case and we check whether it is longer than $\|e\|$.

Note that α_e^{SELF} always exists, but α_e^{LEAF} and α_e^{ROOT} need not exist. Only cases for which both partial assignments for the children exist are computed.

Correctness. To prove the algorithm correct, we shall prove that the computed partial assignments, α_e^{SELF}, α_e^{LEAF} and α_e^{ROOT}, indeed are the smallest feasible partial assignments according to Definition 1. The lemma below is at the heart of this proof. Essentially, it states that we can always get a partial assignment with infinite front-allowance and minimal front-length by increasing the size of an assignment by at most one.

Lemma 2. *For any edge* e *in* $\mathcal{T}$*, we know that* $|\alpha_e^{\text{SELF}}| \leq 1 + \min\{|\alpha_e^{\text{LEAF}}|, |\alpha_e^{\text{ROOT}}|\}$*, where the size of a partial assignment is considered infinite if it does not exist.*

Proof. Consider α_e^{LEAF} or α_e^{ROOT}. If we change the assignment of e to SELF, we obtain again a feasible partial assignment. The lemma readily follows. □

Lemma 3. *The computed partial assignments correspond to Definition 1.*

Proof. We prove this lemma via structural induction. In the base case, e is a leaf of $\mathcal{T}$. Hence, it is an edge of P and the only partial role assignment is α_e^{SELF} with size zero (since e is not in C). Trivially, this has minimal size.

In the inductive case, e is not a leaf of $\mathcal{T}$. It has two children, e_1 and e_2. Let β_e be an optimal partial assignment, according to Definition 1. It implies partial assignments β_{e_1} and β_{e_2} for its two subtrees. Let $\alpha_{e_1} = \alpha_{e_1}^{\beta_e(e_1)}$ be a shorthand for the partial assignment computed by our algorithm, for the given case; α_{e_2} is defined analogously. We use $*$ to consistently indicate either e_1 or e_2.

If $|\beta_*| < |\alpha_*|$, we arrive at a contradiction with the induction hypothesis, which implies that α_* has minimal size.

To argue about the case that $|\beta_*| \geq |\alpha_*|$ holds for both children, we first make the following observations. If $|\beta_*| = |\alpha_*|$, then we can replace β_* with α_* without making the solution worse: by the induction hypothesis, α_* cannot have a greater front-length or a lower front-allowance. If $|\beta_*| > |\alpha_*|$, we cannot make this replacement as α_* may have a greater front-allowance or lower front-length. However, by Lemma 2, we now know that $|\alpha_*^{\text{SELF}}| \leq |\beta_*|$ and this assignment has overall minimal front-length and infinite front-allowance. Hence, replacing β_* with α_*^{SELF} does not make the solution worse.

When we carry out both replacements as described above, we obtain a partial assignment that is not worse than β_e and thus adheres to Definition 1. Due to exhaustive case analysis, our algorithm computes this partial assignment. □

The computed partial assignment α_r^{SELF} corresponds to a full role assignment; it is minimal by Lemma 3. This readily implies the following theorem.

Theorem 1. *The algorithm computes an optimal solution to* MinSizePoly.

Complexity. After building $\mathcal{T}$, the straightforward implementation of this algorithm runs in optimal $O(n)$ time, for a polygon with n edges. Keeping track of which cases give the best result in the computation of each partial assignment, allows the recovery of the optimal solution in $O(n)$ time as well.

4.2 Fewer diagonals

Suppose we require only that C is a nonintersecting set of diagonals inside P. Our algorithm can be modified to also deal with such a case. The most significant change is that $\mathcal{T}$ is no longer binary: nodes may have higher degree. Lemma 1 and Lemma 2 straightforwardly generalize to this case. Hence, we may conclude that an optimal partial assignment can be obtained by using LEAF assignments of those children of e that have the smallest front-length. Thus, we sort the children according to front-length of their LEAF assignment. Testing every child with a ROOT assignment, we can do a binary search to find the best selection of other children to use a LEAF assignment, the rest using SELF. Hence, processing a single edge e takes $O(d_e \log d_e)$ time, where d_e is the degree in $\mathcal{T}$. The total execution time is $O(n \log d)$ where d is the maximal degree in $\mathcal{T}$.

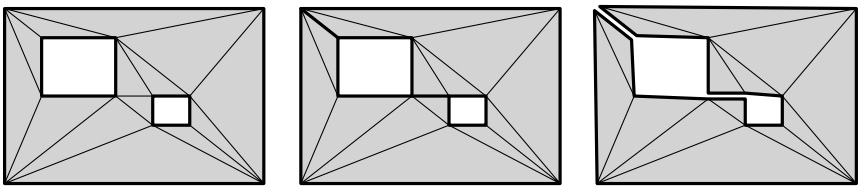


Fig. 3. (left) A polygon with two holes. (middle) Two edges are used as T, to connect the boundaries. (right) T is used to define a single polygon without holes.

4.3 Polygons with holes

Let us consider a simple polygon P *with* holes; C is an inner triangulation of P. To bound the dilation, we need at least some edges to connect the outer boundary of P and each hole. We thus proceed as follows, similar to [8]. First, we compute a minimal-length set $T \subseteq C$ that connects these boundaries, i.e., a minimal spanning tree on the boundary components of P. We use these edges to carve open P into a polygon P_T without holes (see Fig. 3). We then run our algorithm on P_T; let S_T denote its solution. The solution S to P is given by $S_T \cup T$. This heuristic does not provide an approximation guarantee, since the distance along the boundary of P_T can be higher than the distance in the graph $P \cup T$; this may result in adding edges to the solution unnecessarily.

4.4 Minimizing the total weight or length of the selected edges

We now analyze the computational complexity of MinLengthPoly and MinWeightPoly and, thereafter, present algorithms for their solution.

Theorem 2. MinLengthPoly *is weakly NP-hard.*

Proof. Our proof is by reduction from the weakly NP-complete problem Partition, defined as follows: let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a set of positive integers and let $A = \sum_{a_i \in \mathcal{A}} a_i$; is there a set $I \subseteq \mathcal{A}$ such that $\sum_{a_i \in I} a_i = A/2$? For a Partition instance, we construct a MinLengthPoly instance $\mathcal{M}$ with $\tau = 3$ and the polygon P and triangulation C as shown in Fig. 4, using one last point at distance $7A$ to the right of v_{n+1}. We prove that $\mathcal{M}$ admits a solution S of total length at most $3A/2$ if and only if $\mathcal{A}$ is a yes-instance of Partition.

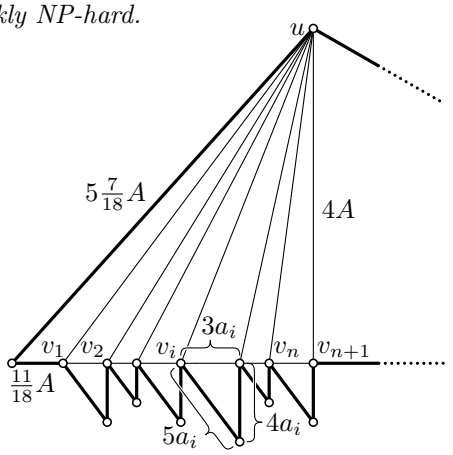


Fig. 4. MinLength instance constructed for instance $\{a_1, a_2, \ldots, a_n\}$ of Partition.

Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a yes-instance of Partition and let $I \subseteq \mathcal{A}$ be such that $\sum_{a_i \in I} a_i = A/2$. We show that $S = \{\{v_i, v_{i+1}\} \mid i \in I\}$ is a solution to

MinLength instance $\mathcal{M}$ with total length at most $3A/2$. Every edge $\{v_i, v_{i+1}\} \in C$ with $i \in \{1, \ldots, n\}$ (i.e., every *horizontal* edge) is trivially satisfied as P already contains a path of length $3a_i$. The vertical edge $\{u, v_{n+1}\}$ is exactly satisfied: walking in counter-clockwise direction along P yields a u-v_{n+1} path of length $15A$ and every horizontal edge $\{v_i, v_{i+1}\} \in S$ reduces the length of this path by $5a_i + 4a_i - 3a_i = 6a_i$; therefore, the shortest u-v_{n+1} path has total length $15A - 6A/2 = 12A = \tau 4A = \tau \|\{u, v_{n+1}\}\|$. Every other edge $\{u, v_i\}$ incident to u is satisfied, because it is longer than $\{u, v_{n+1}\}$, while at the same time the shortest u-v_i path is shorter than the shortest u-v_{n+1} path. By construction, the selected edges have total length $3A/2$.

Now, let $S \subseteq C$ be a solution to $\mathcal{M}$ of total length at most $3A/2$. Because every non-horizontal edge has a length of at least $4A$, S contains only horizontal edges. The edge $\{u, v_{n+1}\}$ can be satisfied only if the total length of horizontal edges in S is at least $3A/2$: hence, the total length of S is exactly $3A/2$. Therefore, the numbers in $\mathcal{A}$ corresponding to the edges in S sum up to $A/2$.

The coordinates of the vertices of the input polygon are rationals—or integer if we scale by a factor of 18—and polynomial in the sum A of $\mathcal{A}$. Therefore, the reduction can be computed in pseudopolynomial time. □

Theorem 3. MinWeightPoly *is weakly NP-hard.*

Proof. We use the same reduction as in the proof of Theorem 2, except that we define the weights as a part of the MinWeightPoly instance: we set the weight of each horizontal edge to its length and of each other edge to $4A$. All weights are polynomial in A. With this the argument works as before. □

Since MinLengthPoly and MinWeightPoly are weakly NP-hard, the more general problems MinLength and MinWeight are weakly NP-hard too.

Furthermore, the polygon that we constructed for our reduction admits only one triangulation. Therefore, the problems do not become easier, if we restrict the triangulation implied by C, e.g. to a constrained Delaunay triangulation [6].

Exact solution of MinWeightPoly. The algorithm for MinSizePoly can be adapted to solve MinWeightPoly, assuming integer weights. Let $w\colon C \to \mathbb{N}$ denote the weight function. In the unweighted case, Lemma 2 implies that leaf or root assignments with size over $|\alpha_e^{\text{SELF}}| - 1$ are never needed. Its weighted variant states that, for an edge e, leaf or root assignments with size over $W(\alpha_e^{\text{SELF}}) - w(e)$ are never needed, where $W(\cdot)$ denotes the sum of weights over all edges with a self assignment. Thus, for each diagonal e and $i \in \{1, \ldots, w(e)\}$, we compute a leaf assignment with total weight exactly $w(\alpha_e^{\text{SELF}}) - i$ and minimal front-length. Analogously, we compute up to $w(e)$ root assignments, with maximal front-allowance. A straightforward implementation for computing the partial solutions for an edge from its children's solutions thus takes $O(w(e)^2)$ time. Therefore, this algorithm takes $O(\sum_{e \in P \cup C} w(e)^2) \subseteq O(w_{\max} \cdot w_{\text{sum}}) \subseteq O(n w_{\max}^2)$ time, where $w_{\max} = \max_{e \in C} w(e)$ and $w_{\text{sum}} = \sum_{e \in C} w(e)$.

Approximating MinLengthPoly. If edge lengths are integer or fixed-point numbers, the weighted algorithm can compute the solution in pseudopolynomial time. Otherwise, rounding yields an approximate solution, as detailed below.

Let λ denote a small constant and assume $1+\lambda \leq \min_{e \in C} \|e\|$. We define two weight functions: $w(e) = 2\lambda \cdot round(\|e\|/(2\lambda))$ and $w'(e) = round(\|e\|/(2\lambda))$. We run the weighted algorithm using w' as its integer weight function. However, w and w' are identical up to scaling and thus produce the same optimal results. The rounding in w implies $\|e\| - \lambda < w(e) \leq \|e\| + \lambda$ and $w(e) > 1$ by assumption.

Let S denote the result of the algorithm; it has weight $w(S) = \sum_{e \in S} w(e)$ and length $l(S) = \sum_{e \in S} \|e\|$. We find that $w(S) > l(S) - \lambda|S| > l(S) - \lambda w(S)$, implying $l(S) < (1+\lambda)w(S)$. Let S^* denote an optimal solution to MinLengthPoly; we find $w(S^*) \leq l(S^*) + \lambda|S^*| \leq l(S^*) + \lambda w(S^*)$ and thus $l(S^*) \geq (1-\lambda)w(S^*)$. The approximation ratio obtained by our algorithm is $l(S)/l(S^*) < (1+\lambda)w(S)/((1-\lambda)w(S^*))$. Since S is optimal in terms of weight, this simplifies to $(1+\lambda)/(1-\lambda)$.

The running time of this approach is $O(nW'^2)$ where $W' = \max_{e \in C} w'(e)$. As $w'(e) \leq \|e\|/(2\lambda) + \frac{1}{2}$, we find that this is $O(nL^2/\lambda^2)$ where $L = \max_{e \in C} \|e\|$. We thus get a (pseudo)PTAS to approximate MinLengthPoly, that computes a $(1+\varepsilon)$-approximation in $O(nL^2(\frac{2+\varepsilon}{\varepsilon})^2) = O(nL^2/\varepsilon^2)$ time.

5 Use Cases

Two vertices lying on opposite sides of a narrow part of a polygon typically have a very large dilation: a connection across the strait of Gibraltar, for example, is much shorter than a path along the coast, all around the Mediterranean Sea. Hence, our dilation-based method may find natural subregions of a polygon. This general hope is confirmed by the results that we obtained with implementations of our algorithms; see Fig. 5. Here we apply our method to two specific problems: computing distorted maps (Section 5.1) and aggregating areas (Section 5.2).

5.1 Computing distorted maps

Several methods exist to distort a map, for example, to resolve spatial conflicts or to emphasize certain information. Such methods often rely on constraints that

(a) MinSizePoly, $\tau = 2$ (b) MinSizePoly, $\tau = 5$ (c) MinLengthPoly, $\tau = 5$

Fig. 5. Results of our algorithms for a part of Europe. Especially the MinLengthPoly solution (c) nicely reflects the Iberian and Italian peninsulas.

are defined based on a geometric graph representing the map [10,11]. An edge in this graph may represent a line segment of a map object, but usually additional edges are needed to model the constraints for the output map. We consider our graph augmentation method as a useful tool for finding such relevant edges.

The method of Harrie and Sarjakoski [10] for the resolution of conflicts relies on a constrained Delaunay triangulation of the map objects. A constraint for the length of a triangle edge $e = \{u, v\}$ is introduced if e is shorter than a threshold ε and the map does not contain a u-v path of less than a number k of line segments. Similarly, our method selects edges of a triangulation based on a geometric distance and a graph-theoretical distance between two vertices of the map. However, while the method of Harrie and Sarjakoski measures the graph-theoretical distance in the input map, our method considers the graph-theoretical distance *after* augmenting the map with the selected edges. We consider our approach promising as it avoids redundant constraints.

The method of Haunert and Sering [11] enlarges a user-selected focus region in a map while minimizing the distortion, which is measured at the edges of a graph representing the map objects, for example, a network of roads or country borders. Additional edges are necessary if the relative position should be maintained for some pairs of vertices: e.g., vertices on opposite sides of a strait. To make a good selection of edges, Van Dijk et al. [18] have developed a greedy heuristic that iteratively augments the map with an edge of maximal dilation (among all edges of a constrained Delaunay triangulation) while the dilation of the graph exceeds a certain threshold. In contrast, our linear-time algorithm for polygons makes an optimal selection of multiple edges.

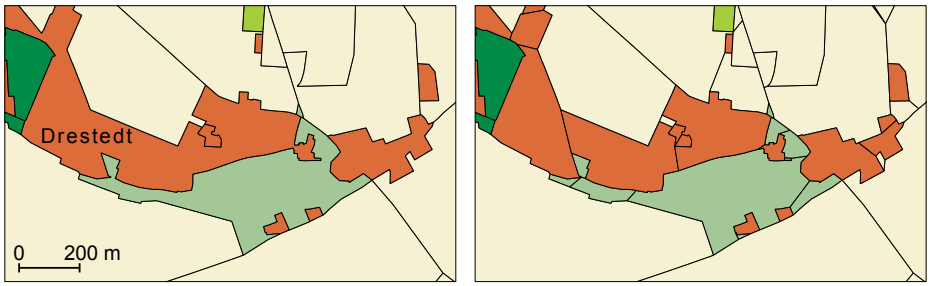
Fig. 6 shows results that we obtained with the method of Haunert and Sering [11] when enlarging Wales in a polygon representing Great Britain. For the result in Fig. 6(middle), only distortions of the edges of that polygon were taken into account, which almost caused a collision of England's east and west coast. A better result is obtained with the additional edges (see Fig. 6(right)): east-west relations are preserved more accurately, yielding a more "solid" deformation.

5.2 Area aggregation

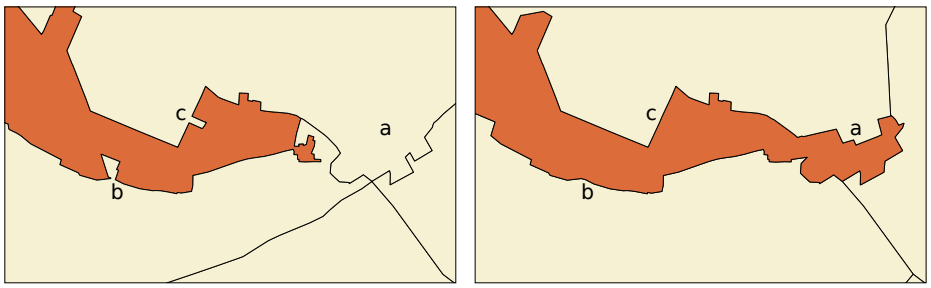
Information on land cover is often given as a planar subdivision that consists of regions of different classes (urban, rural, forest, etc.). To generalize such data, one often aggregates the areas into larger regions such that many-to-one relationships arise. Usually, every output area must have at least a certain minimal size. Subject to this requirement, Haunert and Wolff [12] suggested minimizing a cost function that combines two objectives: the overall weighted class change should be small and the resulting areas should be geometrically compact. They showed that the problem is NP-hard and developed an exact method based on integer linear programming and a heuristic method based on simulated annealing.

Fig. 7(a) shows a sample from the German digital landscape model ATKIS DLM 50, corresponding to a topographic map of scale 1:50 000. We processed this sample with the simulated-annealing-based aggregation method of Haunert and Wolff [12]; see Fig. 7(c). Each output polygon has at least $400\,000\,\mathrm{m}^2$, which is a

Fig. 6. (left) A polygon representing Great Britain with the edges of a MinSizePoly solution with $\tau = 2$. Variable-scale maps computed (middle) without and (right) with consideration of the selected diagonals. The method of Haunert and Sering [11] was used with a scale factor of 2 for Wales; the results were scaled to the same height.



(a) original landscape model (b) result of partitioning polygons in (a)



(c) result of aggregating polygons in (a) (d) result of aggregating polygons in (b)

Fig. 7. Results of the simulated-annealing-based aggregation method of Haunert and Wolff [12] when applied to an example from the landscape model ATKIS DLM 50 (c) without and (d) with application of our MinSizePoly algorithm with $\tau = 4$ for pre-processing. Three corresponding parts in the solutions are labeled with a, b, and c.

requirement for the scale 1:250 000. Observe that several settlement areas (red) are lost. To obtain a better solution, we apply our algorithm for MinSizePoly with $\tau = 4$ and use its result (Fig. 7(b)) as input for the aggregation method. The solution that we obtain (Fig. 7(d)) is clearly better with respect to the total class change: the relatively large settlement labeled with a is retained. Moreover, more compact shapes have been produced, for example, by filling small concavities in the polygons; see the labels b and c. Based on the objective function defined by Haunert and Wolff we can quantify this improvement: for a sample of $n_1 = 325$ polygons from ATKIS DLM 50 the aggregation method yielded a solution of 7.1% less total cost when using the polygon partitioning algorithm, which resulted in $n_2 = 881$ polygons. The cost for class change was reduced by 3.2% and the cost for non-compactness by 12.2%. The higher quality comes at the cost of an increased number of input polygons for the aggregation method. Hence, fast heuristics for aggregation are needed and it is reasonable to minimize the number of output polygons when using our polygon partitioning method. In our experiments, we ran simulated annealing with the same very large number ($8\,810\,000 = n_2 \cdot 10^4$) of iterations to produce near-optimal solutions; this took slightly more than half an hour on a desktop PC.

6 Conclusion

We studied the algorithmic problem of augmenting a simple polygon P of n edges by adding edges from an internal triangulation to bound its dilation. We described an optimal linear-time algorithm to minimize the number of edges added. Moreover, we gave an $O(n \log d)$ algorithm for dealing with any crossing-free set C of candidates (d is the maximal number of neighbors of a region induced by P and C) and a heuristic for polygons with holes. Furthermore, we proved that the weighted case and the length-weighted case are weakly NP-hard. We gave an $O(nw_{\max}^2)$ algorithm for the former problem ($w_{\max}$ is the maximal weight of an edge) and a $(1+\varepsilon)$-approximation algorithm for the latter.

We evaluated the benefits of using augmentation in two use cases: distorting maps and area aggregation. When distorting a map to enlarge a focus region, the augmentation leads to a better preserved shape throughout the map. When aggregating areas, it yields 3.2% less class change and 12.2% better compactness.

Future work. Our results leave several interesting open algorithmic problems. E.g., can we construct an algorithm that can deal with a candidate set C that contains intersecting edges, but the solution must be planar? However, this may imply that no solution exists. What if we allow not only internal diagonals of a polygon, but any edge that does not cross the polygon boundary?

We plan to run extensive experiments to further explore graph augmentation for our use cases, to provide guidelines for parameter and weight selection and model the trade-offs between computation time and quality more explicitly.

Acknowledgments. The authors would like to thank Johannes Oehrlein for helpful discussions on the topic of this paper. W. Meulemans is supported by Marie Skłodowska-Curie Action MSCA-H2020-IF-2014 656741.

References

1. B. Aronov, K. Buchin, M. Buchin, B. Jansen, T. de Jong, M. van Kreveld, M. Löffler, J. Luo, R. I. Silveira, and B. Speckmann. Connect the dot: computing feed-links for network extension. *J. Spatial Inf. Sci.*, 3:3–31, 2011.
2. B. Aronov, M. de Berg, O. Cheong, J. Gudmundsson, H. Haverkort, M. Smid, and A. Vigneron. Sparse geometric graphs with small dilation. *Comp. Geom.*, 40:207–219, 2008.
3. P. Bose and J. Keil. On the stretch factor of the constrained Delaunay triangulation. In *Proc. 3rd Int. S. Vor. Diag. in Sci. & Eng.*, pages 25–31, 2006.
4. P. Bose and M. Smid. On plane geometric spanners: a survey and open problems. *Comp. Geom.*, 47(7):818–830, 2013.
5. B. Chazelle and D. Dobkin. Decomposing a polygon into its convex parts. In *Proc. 11th S. Theory of Comp.*, pages 38–48, 1979.
6. L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4(1–4):97–108, 1989.
7. M. Farshi, P. Giannopoulos, and J. Gudmundsson. Improving the stretch factor of a geometric network by edge augmentation. *SIAM J. Comp.*, 38(1):226–240, 2008.
8. H. Y. F. Feng and T. Pavlidis. Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition. *IEEE Trans. Comput.*, 24(6):636–650, 1975.
9. P. Giannopoulos, R. Klein, C. Knauer, M. Kutz, and D. Marx. Computing geometric minimum-dilation graphs is NP-hard. *Int. J. Comp. Geom. & Appl.*, 20(2):147–173, 2010.
10. L. Harrie and T. Sarjakoski. Simultaneous graphic generalization of vector data sets. *GeoInformatica*, 6(3):233–261, 2002.
11. J.-H. Haunert and L. Sering. Drawing road networks with focus regions. *IEEE Trans. Vis. & Comp. Graph.*, 17(12):2555–2562, 2011.
12. J.-H. Haunert and A. Wolff. Area aggregation in map generalisation by mixed-integer programming. *Int. J. Geogr. Inf. Sci.*, 24(12):1871–1897, 2010.
13. J. M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. *Int. J. Comp. Geom. & Appl.*, 12(3):181–192, 2002.
14. R. Klein, C. Levcopoulos, and A. Lingas. A PTAS for minimum vertex dilation triangulation of a simple polygon with a constant number of sources of dilation. *Comp. Geom.*, 34:28–34, 2006.
15. J.-M. Lien and N. M. Amato. Approximate convex decomposition of polygons. *Comp. Geom. Theory & Appl.*, 35(1):100–123, aug 2006.
16. A. Lingas. The power of non-rectilinear holes. In *Proc. 9th Colloquium on Automata, Languages and Programming*, pages 369–383, 1982.
17. K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(3):239–251, 1995.
18. T. C. van Dijk, A. van Goethem, J.-H. Haunert, W. Meulemans, and B. Speckmann. Accentuating focus maps via partial schematization. In *Proc. 21st ACM SIGSPATIAL Int. C. Advances Geogr. Inf. Syst.*, pages 418–421, 2013.
19. A. Voisard, M. O. Scholl, and P. Rigaux. *Spatial Databases: With Application to GIS*. Morgan Kaufmann, 2002.
20. C. Wulff-Nilsen. Computing the dilation of edge-augmented graphs in metric spaces. *Comp. Geom.*, 43(2):68–72, 2010.