Area-preserving simplification and schematization of polygonal subdivisions

Citation for published version (APA):

Buchin, K., Meulemans, W., Van Renssen, A., & Speckmann, B. (2016). Area-preserving simplification and schematization of polygonal subdivisions. *ACM Transactions on Spatial Algorithms and Systems*, 2(1), 1-36. [2]. <https://doi.org/10.1145/2818373>

DOI:

[10.1145/2818373](https://doi.org/10.1145/2818373)

Document status and date:

Published: 08/04/2016

Document Version:

Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Area-Preserving Simplification and Schematization of Polygonal Subdivisions

KEVIN BUCHIN, TU Eindhoven, The Netherlands
WOUTER MEULEMANS, City University London, United Kingdom
ANDRÉ VAN RENSSSEN, National Institute of Informatics, Japan
BETTINA SPECKMANN, TU Eindhoven, The Netherlands

In this paper, we study automated simplification and schematization of territorial outlines. We present a quadratic-time simplification algorithm based on an operation called an edge-move. We prove that the number of edges of any nonconvex simple polygon can be reduced with this operation. Moreover, edge-moves preserve area and topology and do not introduce new orientations. The latter property in particular makes the algorithm highly suitable for schematization in which all resulting lines are required to be parallel to one of a given set of lines (orientations). To obtain such a result, we need only to preprocess the input to use only lines that are parallel to one of the given set. We present an algorithm to enforce such orientation restrictions, again without changing area or topology. Experiments show that our algorithms obtain results of high visual quality.

Categories and Subject Descriptors: H.3.8 [Database Management]: Database Applications—*Spatial Databases and GIS*; I.3.5 [Computer Graphics]: Computational Geometry

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Simplification, schematization, orientation restriction, area preservation, territorial outlines

ACM Reference Format:

Kevin Buchin, Wouter Meulemans, André van Renssen, Bettina Speckmann, 2014. Area-Preserving Simplification and Schematization of Polygonal Subdivisions. *ACM Trans. Spatial Algorithms Syst.* V, N, Article (April 2016), 35 pages.

DOI: <http://dx.doi.org/10.1145/2818373>

1. INTRODUCTION

Simplification is the process of reducing the complexity of a geometric shape by eliminating detail. A prominent area of application is automated cartography, where it is one of the main generalization operators [Regnauld et al. 1999]. It is applied to various geographic elements, such as roads, country borders, and buildings, to make a map more legible for a given scale. Two variants can be distinguished: vertex-restricted and non-vertex-restricted simplification. In the former, a subset of the vertices of the input is selected to obtain a simplification. In the latter, vertices may be placed in ‘new’ locations. In terms of the generalization operators defined by Regnauld et al. [1999], non-vertex-restricted simplification combines simplification and smoothing.

Corresponding author: Wouter Meulemans, wouter.meulemans@city.ac.uk.

Preliminary versions of some material in this paper have appeared in [Buchin et al. 2011b; 2011a; Meulemans et al. 2010]. In particular, the definitions in Section 3.1 originate from [Buchin et al. 2011b; 2011a]. Parts of this work have also been included in W. Meulemans’ PhD thesis [Meulemans 2014].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. 2374-0353/2016/04-ART \$15.00

DOI: <http://dx.doi.org/10.1145/2818373>

Schematization is an extreme variant of simplification. Geometry is heavily distorted in order to obtain a simplistic representation, typically to support a certain task. This is often applied to the visualization of networks, for example, a train or metro network (see Figure 1). Such a schematic map conveys clearly the primary information, the stations and the connections. The actual geometry is heavily distorted to rid the map of detail that would only distract a user. Often, all lines adhere to a certain set $\mathcal{C}$ of orientations: it is a $\mathcal{C}$ -oriented schematization. In this example, the four main orientations are used, but other options include axis-parallel or hexilinear orientation sets. As illustrated in Figure 1, this is not only useful for the network itself. The country outline is also schematized to lower the visual complexity. The presence of this outline helps users to locate stations.

Fig. 1. Map of the rail network of the Netherlands [www.trein-kaart.nl, accessed November 2013].

Therefore, it is desirable to be able to automatically produce such schematized country (or territorial) outlines from a detailed geographic outline. However, most previous efforts are concentrated on the schematization of networks. In this paper we focus on the schematization of territorial outlines, such as country or province borders. Such schematizations can be applied to support transit maps or display fare zone boundaries. Chorematic diagrams provide another application for schematized outlines. These diagrams are highly abstract and typically accompany texts that describe geoproceses.¹ Generally, if exact boundaries are not needed, it is preferable to replace them by schematic ones. This reduces visual clutter and indicates that the purpose of the map is not a (purely) geographic one.

A schematized map of high visual quality satisfies at least the following criteria. Regions are approximated using few links and few orientations.² Boundaries do not intersect and the topology of the original map is maintained. Finally, the output visually resembles the input, that is, region shapes and sizes are preserved as well as possible. It is comparatively easy to avoid self-intersections and to ensure proper adjacencies. However, it is less clear how to create regions of the ‘best’ shape.

Results. We focus on area-preserving simplification and $\mathcal{C}$ -oriented schematization of territorial outlines. That is, for a given subdivision (map with multiple outlines), we wish to find a result with low complexity (number of edges) such that: each region maintains its area, it has the correct topology, and the result visually resembles the input. For schematization, we require in addition that each line segment in the result adheres to a given set $\mathcal{C}$ of orientations.

In Section 3, we present a simplification algorithm based on an *edge-move* operation (see Figure 2). These operations preserve topology and do not introduce new orientations. By combining two edge-moves, they also preserve area. We prove that any

¹We refer to reader to the work of Reimer [2010] for an in-dept analysis of such diagrams.

²Note that this is no strict criterion. Rather, this restriction is a way to obtain schematization.

nonconvex polygon can be simplified with edge-moves. That is, at least one pair of edge-moves can be executed. We also show how to extend edge-moves to include vertices of higher degree such that the algorithm is also suitable for subdivisions.

Since edge-moves do not introduce new orientations, they are particularly suitable for schematization. If the input is C -oriented, the output is C -oriented as well. In Section 4, we describe an algorithm to convert any subdivision into an area-equivalent C -oriented one. We refer to this step as *orientation restriction*. Combining the two algorithms yields a C -oriented schematization algorithm that preserves area and topology.

In Section 5, we discuss results obtained by applying our algorithms to various territorial outlines. A preview of our results is given in Figure 3. We conclude that, with user-defined orientation sets and target complexities, the results are of high visual quality. Note that our algorithms are also applicable to simplify other types of shapes, such as buildings (see [Buchin et al. 2011b]).

For a given input there are many area-equivalent schematizations of equal complexity. It would seem natural to choose the best of those with respect to any standard distance metric, such as the symmetric difference, the Hausdorff distance, or the Fréchet distance. However, our algorithm applies edge-moves heuristically. To justify this approach, we argue in Appendix A that, for common distance measures, the schematization intuitively with the best shape is not necessarily the one with the lowest distance.

Related work. Numerous methods exist for the simplification of lines, polygons, and subdivisions. Popular examples include Douglas-Peucker [1973], Imai-Iri [1988], and Visvalingam-Whyatt [1993]. However, these methods do not necessarily preserve topology or area. Bose et al. [2006] study simplification with areal displacement criteria. They show that it is NP-hard to decide whether a (vertex-restricted) area-equivalent solution exist for monotone curves and give an approximation algorithm. These measures are also studied by Daneshpajouh et al. [2012] for general curves. Tutić and Lapaine [2009] give a line-simplification algorithm that produces area-equivalent results. However, their algorithm does not preserve topology.

Fig. 2. Combination of two edge-moves preserve the area and reduce the complexity; no new orientations are introduced. Multiple options may be possible.

Fig. 3. Results of our algorithm for a group of countries in southeast Asia, each with 130 edges. The area of each country is preserved. (a) Input. (b) Simplification. (c) Octilinear schematization. (d) Schematization with irregular orientations.

Unfortunately, if we formulate topology-preserving simplification as an optimization problem (e.g. minimize the complexity), then many variants are NP-hard [Guibas et al. 1993] and thus are unlikely to admit an efficient algorithm. Nonetheless, a number of methods exist that simplify based on heuristics or guarantee topological correctness by imposing additional constraints. For example, De Berg et al. [1998] describe a method that simplifies a polygonal subdivision without passing over special input points ('landmarks'). This problem, also referred to as homotopic line simplification, was later also studied by Abam et al. [2014]. Estkowski and Mitchell [2001] give a heuristic for simplifying parallel lines, such as elevation contours. Van de Kraats et al. [1995] discuss the special case where the subdivision to be simplified is a printed circuit board. Mustafa et al. [2001] obtain topology-preserving simplification by restricting results to Voronoi cells. Van der Poorten and Jones [2002] use the constrained Delaunay triangulation to find and simplify features without changing the input topology. However, the methods listed above are suitable only for simplification; they are not designed to obtain $\mathcal{C}$ -oriented results.

There is an ample body of work on path and network schematization. To schematize paths, Neyer [1999] presents an algorithm to compute a $\mathcal{C}$ -oriented representation within a given threshold Fréchet distance of an input path; the algorithm assumes that the vertices are in a sense 'well separated' with respect to the given threshold. Merrick and Gudmundsson [2006] describe an algorithm to compute $\mathcal{C}$ -oriented paths with a given Hausdorff distance of an input path and apply it to generate metro map layouts. Note that this result does not necessarily avoid self-intersections. Delling et al. [2010] describe a method to generate $\mathcal{C}$ -oriented route sketches with orthogonal order constraints for monotone paths. Gamsa et al. [2011] show that such route sketching problems are NP-hard in general and provide an integer linear program. For networks, Cabello et al. [2005] give a schematization algorithm which uses two or three links per path, if such a solution exists. Nöllenburg and Wolff [2011] use a method based on mixed-integer programming to generate metro maps using one edge per path. Swan et al. [2007] give an extensive overview of existing schematization algorithms and study their applicability to automated schematic map construction for web services. Algorithms for network schematization can be used to schematize subdivisions. However, they usually do not take criteria such as shape and size preservation into account. As such, these algorithms are unsuitable to schematize territorial outlines.

Work on schematizing shapes is relatively sparse. Cicerone and Cermignani [2012] describe a heuristic for rectilinear or octilinear schematization based on discretizing space near each vertex. However, this method does not guarantee area-equivalent results. Reimer and Meulemans [2011] investigate parallelism for schematization as a wider design space encompassing $\mathcal{C}$ -oriented schematization. They provide a heuristic approach to generate schematizations of low complexity that have high parallelism. Note that $\mathcal{C}$ -oriented schematization automatically results in parallel line segments. Barkowsky et al. [2000] describe an algorithm for schematization of shapes and networks. However, they take a more relaxed view on schematization, one without orientation restrictions.

Recently, another interpretation of schematization has gained increasing attention, one that relies on curves for representing shape rather than line segments. Manually drawn examples of such curved schematizations can be found, for example, in chorematic diagrams [Reimer 2010] and transit maps [Roberts 2012]. Van Goethem et al. [2013] describe a topologically safe framework to generate curved schematizations and illustrate this framework with Bézier curves and circular arcs. Circular arcs are chosen such that the result is area-equivalent to the input. However, the framework admits only vertex-restricted methods. Van Dijk et al. [2013] use this method to create circular-arc schematizations of 'focus maps', maps that have a certain area of interest

enlarged. Iterative methods using circular-arc replacements have also been explored [Van Goethem et al. 2014; Van Dijk et al. 2014]. Van Goethem et al. [2014] focus on polygons and present a user study about the effects of curved schematization; Van Dijk et al. [2014] focus on handling high-degree vertices in subdivisions and networks appropriately. Using a force-directed method, Fink et al. [2013] generate transit maps with Bézier curves. These methods provide an alternative view of schematization, but do not subsume $\mathcal{C}$ -oriented schematization. The choice of schematization style is likely to depend on a number of factors, including user, input shape, task, and context.

We describe an orientation-restriction algorithm for subdivisions. This problem is often encountered in computer graphics: a subdivision in $\mathbb{R}^2$ has to be converted to a similar subdivision in ‘pixel space’, $\mathbb{Z}^2$. It is studied for example by Christ et al. [2010]. However, such methods require the output to be on a grid; the vertices must have integer coordinates. Moreover, these methods do not guarantee that the area of the result is exactly equal to the area of the input.

2. PRELIMINARIES

Subdivisions. Throughout this paper, we refer to a set of territorial outlines as a *subdivision*. A subdivision is a planar straight-line embedding of a graph. As such, it consists of a set of *vertices* and a set of *edges*. The vertices are points in the plane; the edges are line segments connecting two distinct vertices. The *degree* of a vertex is the number of edges incident to the vertex. We refer to vertices of degree 3 or higher as *junctions*. The *complexity* of a subdivision is its number of edges; for a subdivision S , its complexity is denoted by $|S|$. A region enclosed by edges is called a *face*. In addition, there is always exactly one unbounded region called the *exterior face*.

For simplicity, we assume that all vertices have a degree of at least two and at most four. Since we design our algorithms for territorial outlines, these assumptions are valid on most inputs. If so desired, vertices of other degree can be supported as well. Degree-0 vertices could be used as ‘landmarks’ [De Berg et al. 1998]. Degree-1 vertices require an appropriate definition of edge-moves near such vertices. Junctions of degree higher than four pose a problem only for the orientation-restriction algorithm presented in Section 4. This algorithm may still be applied if a sufficient number of orientations is provided and the incident edges of junctions are spread reasonably well around it.

Preserving topology and area. The mental or cognitive map is a persons recollection of geographic entities and relations between these [Tversky 1993]. A basic relation is topological adjacency information: for example, “the Netherlands borders Belgium and Germany” or “Great Britain is an island”. A map with a change in topology interferes with the mental map, thus making it harder to recognize. Therefore, we require that our algorithms are *topologically safe*, that is, they should preserve the correct topology. This means that the faces of the input and output correspond one-to-one, that each face maintains the same neighbors, and that no boundary intersections are introduced. In addition, intersections should be avoided as they increase the visual complexity of a map and decrease its legibility.

When relative sizes between regions change (i.e. Luxembourg is larger than France instead of vice versa), this again interferes with the mental map and thus with recognizability. As a strict way of ensuring correct relative sizes, we require that our algorithms are *area-preserving* methods: the faces of the input and output must have the same area. Such a result is called *area-equivalent* to the input. Area preservation is also important when the area of a region is an integral aspect of the map design (i.e. in the context of equal-area map projections or value-by-area cartograms).

Orientations. The *orientation* of a line segment is its counterclockwise angle with respect to the horizontal axis. Given a set $\mathcal{C}$ of orientations (angles), a line segment is $\mathcal{C}$ -oriented if its orientation is contained in $\mathcal{C}$. A subdivision is $\mathcal{C}$ -oriented if all edges are $\mathcal{C}$ -oriented. We assume that $\mathcal{C}$ contains at least two distinct orientations.

An orientation set $\mathcal{C}$ is described by a series of angles. For example, $\mathcal{C} = \{0, \frac{\pi}{4}, \frac{\pi}{2}\}$ describes a set with the horizontal, vertical, and one diagonal orientation (from bottomleft to topright). A regular orientation set contains orientations that are evenly spaced. Such a set is characterized by some initial angle β and the number of orientations c . The set of orientations is then $\{\beta + i \cdot \frac{\pi}{c} \mid 0 \leq i < c\}$. Typically, β is either zero (the first orientation is horizontal) or $\frac{\pi}{2}$ (the first orientation is vertical). We refer to such sets as horizontal and vertical respectively. Note that these sets are different only for odd values of c . We use $\mathcal{C}_c(\beta)$ to denote a regular set with c orientations and initial angle β . If β is 0, we abbreviate this to $\mathcal{C}_c$. Common orientation sets include rectilinear ($\mathcal{C}_2$), hexilinear ($\mathcal{C}_3$ or $\mathcal{C}_3(\frac{\pi}{2})$), and octilinear ($\mathcal{C}_4$). Note that these are often also referred to as orthogonal, hexagonal, and octagonal respectively. However, our algorithms are not restricted to regular orientation sets; irregular sets can also be used.

3. SIMPLIFICATION

In this section we describe an area-preserving simplification algorithm based on an operation called *edge-move*. We first introduce this operation. Then we describe a heuristic algorithm for iteratively selecting edge-moves in subdivisions. Finally, we show that any nonconvex simple polygon admits edge-moves that preserve topology.

3.1. Edge-moves

Below, we give a precise definition of the edge-move operation. We first introduce it for a simple polygon and afterwards extend it to subdivisions.

Definitions and notation. We are given a simple polygon $P = \langle p_0, \dots, p_{n-1} \rangle$ with n vertices. We treat the vertices circularly, that is, p_i is used as a shorthand for $p_{i \bmod n}$. The *exterior angle* of a vertex is defined as the angle between one edge and the extension of the other. The angle is negative if and only if the vertex is reflex. The sum of all exterior angles of a simple polygon is always equal to 2π .

The boundary of P , denoted by ∂P , consists of n edges: $e_0, \dots, e_{n-1}$. The directed edge e_i starts at vertex p_i and ends at vertex p_{i+1} . As with the vertices, we treat the edges modulo n . We call an edge *convex* or *reflex* if both its vertices are convex or reflex respectively (see Figure 4 for examples).

Configurations and edge-moves. Three consecutive edges $\langle e_{i-1}, e_i, e_{i+1} \rangle$ are called a *configuration*. An edge-move always operates on a configuration. We refer to its central edge e_i as its *inner edge*, the other edges are its *outer edges*.

Let $X = \langle e_{i-1}, e_i, e_{i+1} \rangle$ be a configuration of polygon P . The outer edges, e_{i-1} and e_{i+1} , define two *tracks*, infinite lines through the edges. An *edge-move* on X moves e_i such that its orientation is preserved and its vertices are on the tracks, making the outer edges longer or shorter. An edge-move is *valid* if at least one vertex of e_i remains on its original outer edge and e_i remains on the same side of or on the intersection point of the tracks (if it exists). A *contraction* is an edge-move that causes one of the

Fig. 4. A polygon with its convex edges (a) and reflex edges (b) indicated. Some edges are neither convex nor reflex (c).

Fig. 5. (a) A configuration with its positive (R^+ , light) and negative (R^- , dark) contraction region. (b) A positive edge-move. (c) A negative contraction. Hachures indicate polygon interior.

Fig. 6. Configurations (thick line) with negative and positive blocking points, labeled p and p' respectively. Negative and positive contraction regions are indicated by dark and light shading respectively.

edges of X to reach length zero. Contractions are ‘extremal edge-moves’ and reduce the complexity of P . An edge-move is *positive* if it adds area to P and *negative* if it removes area. Figure 5 shows some examples.

A configuration supports edge-moves, either positive, negative, or both. A *positive configuration* supports positive edge-moves; a *negative configuration* supports negative edge-moves. The *positive contraction region* $R^+(X)$ of a positive configuration $X = \langle e_{i-1}, e_i, e_{i+1} \rangle$ is the region enclosed by e_i , the tracks, and the position of e_i after a positive contraction. A *feasible positive configuration* is a configuration for which $R^+(X)$ does not contain any points on ∂P , except for those that belong to X . Similarly, we define the *negative contraction region* $R^-(X)$ and a *feasible negative configuration*. If a positive or negative configuration is feasible, then any valid positive or negative edge-move respectively is feasible. If a positive configuration is infeasible, then there is some point on $\partial P \setminus X$ in $R^+(X)$. A point in $\partial P \setminus X \cap R^+(X)$ that is closest to the line through e_i is called a *positive blocking point*. Analogously, X can have a *negative blocking point*. Examples of blocking points are given in Figure 6.

Complementary moves. Since we desire an approach that preserves the area of the polygon, we combine two *complementary* feasible configurations. Such a complementary pair consists of one positive and one negative configuration and we execute an edge-move on both simultaneously. The one with the smaller contraction region is contracted, while the other is moved just far enough to compensate for the area change. Two configurations *conflict* when they share an edge, unless they share only outer edges and one of these has a convex and a reflex vertex. In this special case the two edge-moves both either shorten or lengthen the shared edge. We call two nonconflicting complementary feasible configurations a *proper configuration pair*.

Subdivisions. We now extend the definition of edge-moves to subdivisions. To preserve the area of each face, it is important to combine only edge-moves of which the inner edges are incident to the same faces. In addition, we must define edge-moves in the presence of junctions (vertices of degree 3 or higher). A configuration with inner vertices of degree 2 supports edge-moves as defined for polygons. However, if an inner vertex of a configuration is a junction of degree 4 or higher, then performing edge-moves on this configuration could alter the topology or result in an area change. Hence, we do not allow moving such junctions; as a consequence any incident edge cannot be moved. For junctions of degree 3, there is some flexibility. The following three

Fig. 7. Three cases for degree-3 vertex v (white) when moving edge e . Copied vertices are indicated in black. From left to right: initial situation; moving e upward; moving e downward.

cases can occur for an edge-move with inner edge e incident to a vertex v of degree 3 (refer to Figure 7).

- (a) The other edges of v have the same orientation. These edges must lie on different sides of the line through e . When moving e in either direction, vertex v can slide along the other two edges.
- (b) The other edges of v do not have the same orientation, but lie on different sides of the line through e . When moving e in either direction, vertex v can slide along the edge on that side, but a copy of v must be introduced on its original position to preserve area and maintain correct topology.
- (c) The other edges of v do not have the same orientation, but lie on the same side of the line through e . When moving e in the direction of the other edges (upward in Figure 7(c)), vertex v can slide along the edge that makes the smallest angle to e , but a copy of v must be introduced on its original position to preserve area and maintain correct topology. When moving e in the other direction (downward in Figure 7(c)), vertex v can slide along the extension of either edge. We use the edge that has the largest angle to e , unless this edge has the same orientation as e (an angle of π). A copy of v must be introduced on its original position and this vertex has degree 3.

When combining edge-moves that introduce a new vertex, it is important to ensure that the complexity of the subdivision is still strictly reduced. Combinations of edge-moves that do not decrease the complexity are not permitted. Computing the total complexity reduction of a single contraction or edge-move is straightforward. Finally, to ensure correct topology, it is important that a junction is never removed, moved over or placed on another junction.

3.2. Simplification algorithm

In the previous section we introduced edge-moves for polygons and subdivisions. By performing these operations in proper configuration pairs, the complexity of the subdivision is reduced while preserving area and topology. Here we describe how to use these edge-moves to build an efficient simplification algorithm. Algorithm 1 provides an overview. The main results of this section are summarized in the theorem below.

ALGORITHM 1: Simplify(S, k)**Input:** a subdivision S and an integer k **Output:** a subdivision with at most k edges or without a proper configuration pair

```

1 Initialize blocking numbers for each edge
2 while  $|S| > k$  do
3   Determine minimal configuration pair  $(X_1, X_2)$  over all boundaries in  $S$ 
4   if no minimal configuration pair exists then
5     return  $S$ 
6   else
7     Decrease blocking numbers for configurations blocked by  $X_1$  or  $X_2$ 
8     Contract  $(X_1, X_2)$  to reduce  $|S|$ 
9     Increase blocking numbers for configurations blocked by  $X_1$  or  $X_2$ 
10    Reinitialize blocking numbers for configurations involving an edge of  $X_1$  or  $X_2$ 
11 return  $S$ 

```

THEOREM 3.1. *Given a subdivision S and an integer k , Algorithm 1 computes a subdivision that is area-equivalent and topologically equivalent to S , such that the resulting subdivision either has at most k edges or admits no proper configuration pair. Algorithm 1 runs in $O(|S|^2)$ time.*

The algorithm iteratively finds two complementary feasible configurations that incur minimal change. This minimal configuration pair is executed to reduce the complexity of the subdivision. This process is repeated until either the desired output complexity is reached or no minimal configuration pair exists. It makes use of ‘blocking numbers’ to keep track of the feasibility of configurations; these will be formally introduced later in this section. Below, we provide the details for the two crucial steps: how to compute a minimal configuration pair (Line 3) and the steps involved in performing the contraction (Line 8). The latter must ensure that the blocking numbers remain correct.

Choosing edge-moves. We need to choose a proper configuration pair to perform a contraction and an edge-move simultaneously. We search for a pair that incurs the smallest visual change; we refer to this pair as the *minimal configuration pair*. This minimality can be formalized in various ways. For our algorithm, we use the area of the contraction region of the smallest of the two involved configurations. This corresponds to half the symmetric difference of a face before and after the operation is performed. We found that minimizing the symmetric difference does not always yield satisfying results (see Appendix A). However, the problem of severely changing the shape does not occur here, since edge-moves make only local changes.

Finding pairs. There may be more than a single border between two adjacent faces; edge-moves in one may be combined with edge-moves in the other. To avoid the need to explicitly search for other borders, we assume that the subdivision is partitioned into *face-face boundaries*. That is, for every pair of adjacent faces, all borders that separate them are stored together. In a preprocessing step, these boundaries can be computed in $O(|S|^2)$ time and $O(|S|)$ space. Note that there are at most a linear number of such boundaries since S is a subdivision (a plane graph). The sum of the number of edges over all boundaries is $|S|$.

If a configuration conflicts a complementary one, then these configurations share at least one edge. As a configuration consists of three consecutive edges, a conflict implies that the configurations share either an outer edge, or an outer edge and an inner edge, or all three edges (one edge cannot move in two directions simultaneously). Since each

configuration has two outer edges and one inner edge, this yields five configurations that can conflict.

Knowing this property, we proceed as follows for each face-face boundary to find the minimal configuration pair. First, we find the six smallest positive and the six smallest negative feasible configurations on this boundary in $O(b)$ time, where b is the number of edges of the boundary. Here, ‘smallest’ refers to the minimality measure: the area of the contraction region. As at most five configurations may conflict, at least one of these smallest feasible configurations must be part of a proper configuration pair, if the boundary admits any feasible pair. We need to find only the minimal pair: we compute which of these twelve candidate configurations admits the smallest contraction that is part of a proper configuration pair. For each, we simply traverse the border in $O(b)$ time to search for a nonconflicting complementary feasible configuration. If there are multiple complementary moves, we use the one that is nearest, measured in the number of edges along the boundary. There are only $O(1)$ candidates for one boundary. Hence, computing the minimal pair of a single boundary takes $O(b)$ time. The minimal configuration pair is then simply the minimal pair over all boundaries; it is computed in $O(|S|)$ time.

If none of the boundaries has a proper configuration pair, no minimal configuration pair exists. In this case, the algorithm terminates before reaching the desired complexity k . By Theorem 3.2 (proven in Appendix B), we know that this does not occur for simple polygons (unless k is less than twice the number of orientations in the input).

Checking feasibility. In the analysis above, we assumed that we can determine in constant time whether a configuration is feasible. To this end, each edge e stores two *blocking numbers*. The positive blocking number indicates the number of edges that (partially) reside in the positive contraction region of configuration X with inner edge e ; the edges of X are excluded in this number. Analogously, the negative blocking number indicates the number of edges that overlap the negative contraction region. In the remainder, we refer to the negative and positive blocking numbers collectively as blocking numbers. To initialize these blocking numbers (Line 1), we can simply iterate over all pairs of edges and increment the values accordingly. Hence, this takes $O(n^2)$ time in total. However, we need to update the blocking numbers when contracting the minimal configuration pair. This is discussed in the upcoming paragraph.

Performing a contraction. Once the algorithm has computed a minimal configuration pair (X_1, X_2) , it performs the contraction to reduce the complexity of S . A contraction is performed in a sequence of steps to ensure that the blocking numbers have the correct values afterwards.

In the first step, we discard the contribution that the edges of X_1 and X_2 made to the blocking numbers. To this end, we iterate over all edges in S . For each edge, we decrement the positive (or negative) blocking number by one for each edge of X_1 or X_2 that overlaps the positive (or negative) contraction region. Since the number of edges of X_1 and X_2 is 6, we can perform this entire step in $O(|S|)$ time.

In the second step, we perform the edge-moves that constitute the contraction of (X_1, X_2) . The subdivision S is updated accordingly.

In the third step, we add the contribution to the blocking numbers for the edges that changed during the contraction (i.e., the remaining edges of X_1 and X_2). We repeat the same linear-time process as before, but now increment the numbers instead.

As a result of the contraction, the contraction region changes for a number of configurations. This implies that the blocking numbers may be incorrect for the inner edges of those configurations. Hence, in the fourth and last step, we ensure that these blocking numbers are correct as well. The contraction region of a configuration changes only if one or more of its edges were part of X_1 or X_2 . As we do not allow edge-moves for

junctions of degree 4 or higher, only a small constant number of configurations are affected. For each of the affected edges, we simply reinitialize the blocking number by iterating over all other edges of S . This takes $O(n)$ time per edge and thus also $O(n)$ time in total.

Analysis. To initialize the algorithm, we compute the boundaries of the subdivision and set the blocking numbers for the edges. Both steps take $O(n^2)$ time. As described above, each simplification step can be performed in $O(|S|)$ time using these numbers. Each contraction reduces the complexity of S by at least one: at most $O(|S|-k) = O(|S|)$ steps are performed. Thus, Algorithm 1 computes a simplification in $O(|S|^2)$ time.

This execution time may be prohibitive to process with large maps. Table I (Section 4.4) lists some computation times, though we note that this uses a proof-of-concept implementation, not designed for performance. However, as we showed in [Buchin et al. 2011b], fairly large building data sets can be processed in reasonable time with this algorithm: an input of roughly 15 000 edges was processed in under 4 seconds. Note, however, that a set of buildings is structurally very different from a set of, say, country outlines. For schematization purposes, we typically require a very low complexity for the result. It is therefore reasonable to use an input which has been pre-simplified by another algorithm to a complexity that still greatly exceeds the desired complexity.

Increasing flexibility. It is relatively straightforward to modify the algorithm such that infeasible configurations can be used to compensate for area change. That is, we may allow edge-moves on configurations that have a blocking point p , if we ensure that this edge-move does not move the inner edge to or beyond p . We need to quickly decide whether a configuration can compensate for a change in area. To this end, the algorithm should not maintain the blocking number but the actual blocking point of a configuration. We can then compute in $O(1)$ time the maximal area change that the configuration can compensate for. A simple implementation using lists of blocking points leads to a cubic-time algorithm. However, for territorial outlines, we expect edge-moves to be obstructed by relatively few edges. Hence, we may expect that the actual execution time does not differ much from the original algorithm. This modification is especially relevant for subdivisions as it gives more flexibility. We apply this modified algorithm to obtain the results presented in Section 5.

Alternative choices. We use the area of the contraction region to define the minimal configuration pair. As an alternative, we may use the Fréchet distance between the configuration before and after the edge-move. However, some operations may cause a relatively large Fréchet distance but only a small change in area. The visual change seems to be more strongly correlated to the area than the distance. This is similar to the use of area (rather than distance) to decide on simplification steps in algorithm by Visvalingam and Whyatt [1993].

To compute the minimal configuration pair, we select the nearest complementary edge-move, measured in the number of edges along the boundary. As an alternative, we could opt to use one of the smallest complementary moves which have already been found: as there are six, at least one of these must not conflict. Though it does not improve the asymptotic running time, it eliminates the need for extra traversals. However, this change reduces the ‘locality’ of the minimal configuration pair. To perform edge-moves that make small local changes, we opted to use the nearest complementary move instead.

We do not allow edge-moves that move junctions of degree 4 or higher as these either alter topology or area. By combining more than two edge-moves, we can move area along a ‘cycle’ of faces, allowing edge-moves involving these junctions. This does incur a higher computational cost. However, we consider such an extension to be unnecessary: in our experiments, the desired complexity could be reached without this addition.

3.3. Existence of feasible configurations for polygons

The algorithm presented in the previous section needs to find a proper configuration pair—a nonconflicting pair of complementary feasible configurations—on which to perform edge-moves. One of these configurations must admit a contraction to reduce the complexity and ensure progress. To guarantee a topologically equivalent result, we perform edge-moves only on feasible configurations, that is, on configurations without a blocking point. In a subdivision, such a pair of configurations may not exist and the desired complexity of the output may be unattainable. However, this does not occur for a simple polygon (i.e., a polygon that has no self-intersections), unless it is convex. This is formulated in the theorem below.

THEOREM 3.2. *Every nonconvex simple polygon has a nonconflicting pair of complementary configurations.*

The main implication of this theorem on Algorithm 1 is that it can always make progress on nonconvex polygons by combining two edge-moves, one of which is a contraction, thus reducing $|S|$. Using induction on subsequences of edges of a polygon, we prove that there are feasible negative configurations. By turning a nonconvex polygon ‘inside-out’, we prove that a feasible positive configuration exists as well. By strengthening the above statements and using case distinction, we prove that a nonconflicting pair indeed exists: this theorem indeed holds. The formal proof involves a host of definitions and mathematical detail and is deferred to Appendix B.

4. SCHEMATIZATION

In the previous section we described a topologically safe and area-preserving simplification algorithm. The edge-moves used by this algorithm do not introduce new orientations either. Hence, the orientation set of the result is equal to the orientation set of the input (or a subset thereof). Due to these properties, we obtain an area-preserving $\mathcal{C}$ -oriented schematization algorithm by applying an *orientation-restriction algorithm* beforehand. This algorithm turns a subdivision into an area-equivalent $\mathcal{C}$ -oriented subdivision for any orientation set $\mathcal{C}$.

The quality of the eventual result (i.e. after applying Algorithm 1) improves if the input subdivision does not contain ‘long’ edges. We use a small constant fraction $\lambda = 0.05$ of the diameter of the input as an upper bound for the edge length and split all edges which are longer. In doing so, we ensure that the orientation restriction provides a close approximation of the input subdivision, allowing more options in the simplification step. Otherwise, the orientation restriction may partially predetermine the eventual result.

An overview of our orientation-restriction algorithm is given in Algorithm 2. In this section, we shall provide the details of our method and prove the following theorem.

THEOREM 4.1. *Given a subdivision S and an orientation set $\mathcal{C}$, Algorithm 2 computes a $\mathcal{C}$ -oriented subdivision R that is area-equivalent and topologically equivalent to subdivision S . Algorithm 2 runs in $O(|S|^2 + |R|)$ time.*

The algorithm consists of three high-level steps, each of which we describe in detail in this section. First, the algorithm assigns directions and, based on this assignment, classifies the vertices and edges (Line 1–2, see Section 4.1). Using this classification, it eventually constructs a $\mathcal{C}$ -oriented curve, a ‘staircase’, for each edge (Line 7–10, see Section 4.2). To this end, the required number of ‘steps’ in the staircase is computed such that the result is crossing-free (Line 3–6, see Section 4.3). Though the construction of staircases occurs afterwards, this is described before the details of computing

ALGORITHM 2: OrientationRestriction($S, \mathcal{C}$)**Input:** a subdivision S and an orientation set $\mathcal{C}$.**Output:** a $\mathcal{C}$ -oriented subdivision R that is area-equivalent and topologically equivalent to subdivision S .

```

1 Assign directions to each incident edge for each vertex in  $S$ 
2 Classify vertices and edges in  $S$ 
3 Determine staircase region for each edge in  $S$ 
4 for edges  $e$  in  $S$  do
5     Determine possible interfering edges using staircase regions
6     Compute required number of steps for  $e$ 
7 Initialize empty subdivision  $R$ 
8 for edges  $e$  in  $S$  do
9     Construct staircase for  $e$  with the computed number of steps
10    Add the staircase to  $R$ 
11 return  $R$ 

```

Fig. 8. Subdivision (a) and area-equivalent orientation-restricted results with $\lambda = 1$: (b) Rectilinear, $\mathcal{C}_2$; (c) Octilinear, $\mathcal{C}_4$; (d) Irregular, $\{\frac{\pi}{4}, \frac{11\pi}{12}\}$.

the number of steps: this depends heavily on the geometry of the various staircases. Possible results of this algorithm are illustrated in Figure 8.

4.1. Classification of vertices and edges

The first step in our algorithm is to assign directions and classify vertices and edges in the subdivision (Algorithm 2, Line 1–2). We call an edge *aligned* if it already adheres to an orientation in $\mathcal{C}$, and *unaligned* otherwise. The orientations of $\mathcal{C}$ partition the space around each vertex into $|\mathcal{C}|$ sectors. Each sector has two *associated directions*: these are the directions along the orientations that bound the sector. At both of its endpoints, an edge has one or two associated directions. If the edge is aligned, it has one associated direction, being the direction of the edge. If it is unaligned, the two associated directions correspond to the associated directions of the sector in which it lies. We call a vertex *insignificant*, if the associated directions of its incident edges are disjoint. This means that we can freely choose an associated direction for each incident edge without limiting the choice for other edges. A vertex is called *significant* otherwise, indicating that we are not free to choose directions. We assume that every edge has at most one significant vertex. This property is ensured by splitting an edge with two significant vertices, as the new vertex is insignificant.

For each edge, we now assign a $\mathcal{C}$ -oriented direction at its significant vertex. If an edge has two insignificant vertices, we randomly pick one to be treated as the significant vertex. Since we assume that each vertex has degree at most 4 (see Section 2), we can always find an assignment that ensures the following three properties:

- no two edges are assigned the same direction at a common vertex;
- the cyclic order of edges is preserved;
- the total angular deviation is minimized.

Fig. 9. Classification of edges: basic edges (solid); evading edges (thick solid); deviating edges (dotted). Significant vertices are marked with gray dots. (a–g) Rectilinear, C_2 . (h–n) Octilinear, C_4 .

The angular deviation of an edge at its significant vertex is always smaller than π . Based on this assignment, we classify the edges as follows (refer to Figure 9 for examples).

- *Aligned basic edge*: the edge is aligned and its assigned direction is its (only) associated direction. See Figure 9(b–c,i–j,n).
- *Unaligned basic edge*: the edge is unaligned, its assigned direction is one of its associated directions, and its significant vertex does not have another incident unaligned edge in the same sector that is assigned one of its associated directions. See Figure 9(a–c,g–n).
- *Evading edge*: the edge is unaligned, its assigned direction is one of its associated directions, and its significant vertex has another incident unaligned edge in the same sector that is assigned one of its associated directions. Evading edges always occur in pairs and aligned evading edges cannot occur. See Figure 9(d–g,l–n).
- *Aligned deviating edge*: the edge is aligned and its assigned direction is different from its associated direction. Note that its significant vertex must be a junction. See Figure 9(f,m).
- *Unaligned deviating edge*: the edge is unaligned and its assigned direction is not one of its associated directions. Note that its significant vertex must be a junction. See Figure 9(e–g,m).

4.2. Converting edges to staircases

To construct a C -oriented subdivision, we create a *staircase* for each edge (Algorithm 2, Line 7–10). A staircase is a sequence of C -oriented edges that starts and ends at the vertices of the edge. A *step* in a staircase is a combination of two C -oriented edges such that the step starts and ends on the edge that the staircase must represent. By increasing the number of steps in the staircase, intersections can be avoided. We describe in Section 4.3 how to obtain the correct number of steps. Once we know the appropriate number of steps, the edge is converted in isolation. Let $e = (v, w)$ denote the edge we wish to convert and s_e the number of steps it must use. Without loss of generality, we assume that v is the significant vertex. The construction of a staircase depends on the classification of edge e .

- If e is an aligned basic edge, we do not change it as it is already C -oriented.
- If e is an unaligned basic edge, we treat it as follows. Let d_1 denote its assigned direction and d_2 its other associated direction. Each step uses one edge parallel to d_1 , called the *assigned edge* of the step, and one parallel to d_2 , called the *associated*

Fig. 10. Staircase examples for basic and evading edges. Gray vertices are significant. (a) Rectilinear basic edge. (b) Octilinear basic edge. (c) Two rectilinear evading edges. (d) Two octilinear evading edges.

edge. A step can start either with the former (an *assigned* step) or with the latter (an *associated* step). Moreover, every step should span exactly a fraction of $1/s_e$ of the length of e . The length of the assigned and associated edge is the same in all steps. Assuming that d_1 and d_2 are normalized vectors, these can be found by solving $(w-v)/s_e = l_1 \cdot d_1 + l_2 \cdot d_2$ for l_1 and l_2 . Every step adds area to one incident face of e and removes it from the other. Since an assigned and an associated step counterbalance the area change, we combine $s_e/2$ assigned steps with $s_e/2$ associated steps.³ By choosing these steps, the procedure guarantees an area-equivalent result. Starting at v , we alternate the steps starting with an assigned step to adhere to the assigned direction. The number of edges used in the staircase is $s_e + 1$. Figure 10(a–b) shows examples for the staircase of a basic edge.

- For an evading edge e , we know that there is another evading edge in the same sector of its shared significant vertex. To avoid intersections, we apply so-called *evasive behavior*. We build a staircase similar to the staircase for an unaligned basic edge. However, instead of alternating the steps, we now first place all assigned steps (starting at v) followed by all associated steps. This results in a staircase of which the first half lies completely on the far side of the evading edge with respect to the other evading edge. This guarantees that there are no intersections near the significant vertex. The number of edges in the staircase is $2s_e - 1$. Figure 10(c–d) shows examples of this evasive behavior.
- An aligned deviating edge is not converted with steps and uses a fixed number of edges. Instead of a number of steps s_e , we derive a value δ_e for such an edge; in Section 4.3 we show how to obtain a sufficiently small value to prevent intersections from occurring. In addition, it uses a small constant ε to ensure that the edge $e = (u, v)$ adheres to its direction at the insignificant vertex w . We use $\varepsilon = 0.1$. Let d_1 denote the assigned direction and d_2 the direction of the edge (i.e., $d_2 = w - v$). We start at vertex v . The first edge is directed along d_1 and has length δ_e . The second edge has length $\|e\|(1 - \varepsilon)/2$, where $\|e\|$ denotes the length of e , and is directed along d_2 . The third edge has length $2\delta_e$ and is directed in the opposite direction of the first edge. The fourth edge is analogous to the second. The fifth edge is analogous to the first. The sixth edge is directed along d_2 and has length $\varepsilon\|e\|$. The number of edges used in the staircase is six, independent of δ_e . Figure 11(a–b) shows examples.
- For an unaligned deviating edge, we make a staircase that resembles that of an evading edge, but adapt it to use the correct assigned direction. We first create the staircase as if the edge was an evading edge. For the purpose of the evasive behavior, we use as an ‘assigned direction’ the associated direction closest to its (actual) assigned direction. However, instead of using s_e steps, we use only $s_e - 1$ steps: $(s_e/2) - 1$ assigned steps followed by $s_e/2$ associated steps. This violates both the area-preservation constraint as well as the assigned direction. To correct for this, we append a region—the area of a single step—to the first edge by ‘dragging’ the

³We ensure that s_e is even for all edges in Section 4.3.

Fig. 11. Examples of rectilinear and octilinear staircases. Gray vertices are significant. (a–b) For an aligned deviating edge. (c–d) For an unaligned deviating edge. Appended region is shaded gray.

first edge of the intermediate staircase in the assigned direction. Note that s_e must be at least four. The number of edges used in the staircase is $2s_e - 1$. Figure 11(c–d) shows examples.

4.3. Computing the number of steps

Here we describe how to compute the number of steps, s_e , for an edge e such that the orientation-restricted result is free of intersections (Algorithm 2, Line 3–6). We ensure that the staircase of an edge is less than a distance $d_e/2$ away from the edge itself, where d_e is the minimal distance from e to another point in subdivision S . However, we need not and should not take all other edges into account. Neighboring edges of e have points that are arbitrarily close to e . This would cause an infinite number of steps. Moreover, we may exclude edges that cannot cause intersections regardless of the number of steps.

Interference. We first determine which edges may interfere with edge $e = (v, w)$, that is, which edges may have staircases that could intersect the staircase of e . Let e' denote some other edge of S . If e and e' do not share a vertex, we first make a rough estimate of whether the staircases could intersect. To this end, we define the *staircase region* of an edge. The staircase region is a bounding region of an edge that contains the staircase regardless of the number of steps.

- For an aligned basic edge, the staircase region is simply the edge itself.
- For unaligned basic and evading edges, the staircase region is the area bounded by lines oriented according to the associated directions (both at v and w); this is illustrated in Figure 12(a).
- For unaligned deviating edges, we use a similar staircase region as an evading edge, but it must accommodate for the appended area that is used to make the staircase adhere to the assigned direction (see Figure 12(b–c)). The appended area is largest for a minimal number of steps (4 steps); let point p denote the vertex of this maximal appended area that is furthest from v . The staircase region is the region containing e that is enclosed by the following lines: lines through w adhering to the associated directions of e ; line through v adhering to associated direction with largest angle to

Fig. 12. Examples of staircase regions (shaded gray). (a) Unaligned basic and evading edges. (b–c) Unaligned deviating edge. (d) Aligned deviating edge.

assigned direction; line through p adhering to associated direction with smallest angle to assigned direction. In the above, we assumed that p lies in the defined region (see Figure 12(b)). However, if this is not the case, we can extend the staircase region by including the vertices of the appended region; this is illustrated in Figure 12(c).

- Aligned deviating edges do not use steps, but a value δ_e instead. We use a value Δ_e as an upper bound for δ_e . To define this value, we use a constant fraction of the edge length: $\Delta_e = 0.1\|e\|$. We compute the staircase using $\delta_e = \Delta_e$ and use its convex hull as staircase region (see Figure 12(d)).

Any staircase with more steps (or $\delta_e < \Delta_e$) is contained in the staircase region. Hence, there is interference only if the staircase regions of e and e' intersect, assuming that these edges do not share a vertex. If e and e' share a vertex v , they interfere only if the edges reside in the same sector with respect to v . To this end, aligned deviating edges are considered to reside in both sectors. For these pairs, we determine interference purely based on the classification. Aligned basic edges cannot cause interference. Unaligned basic edges interfere with unaligned and aligned deviating edges: by definition an unaligned basic edge cannot be in the same sector as an evading edge. Evading edges interfere with other evading edges, unaligned and aligned deviating edges. Deviating edges, both aligned and unaligned, cannot interfere with one another: the assigned direction is not one of the associated directions and hence another edge must lie in between.

Edge distance. We define the distance d_e for an edge e as the minimum over all distances $d_{e,e'}$ for all edges e' that interfere with e . Distance $d_{e,e'}$ is computed as follows. If e and e' do not share a vertex, then $d_{e,e'}$ is simply the minimal distance between the edges. However, if e and e' do share a vertex, then we must again look at the classification. Depending on this classification, we ignore parts of the edges (measured from the shared vertex) in the distance computation. In these cases, the staircase construction guarantees that no intersections are introduced.

- If e is an unaligned basic edge, then e' is either an aligned or unaligned deviating edge. If e' is aligned, then we ignore a fraction of $(1 - \varepsilon)/2$ of e' . This fraction corresponds to the part of the staircase of e' that resides on the ‘far side’ of e' . If e' is unaligned, then we ignore a fraction of e' equal to the length of the first step. That is, we ignore a fraction of $1/(s_{e'} - 1)$. This is illustrated in Figure 13(a).
- If e is an evading edge, then e' is either an evading or deviating edge. If e' is an evading edge, we ignore the first half of e (but not of e'). Due to the evasive behavior in the construction, we know that there are no intersections here. This is illustrated in Figure 13(b). If e' is a deviating edge, we treat it as if e was an unaligned basic edge.
- If e is an aligned deviating edge, then e' is either an unaligned basic edge or an evading edge. Regardless of e' , we ignore a fraction of $(1 - \varepsilon)/2$ of e (see Figure 13(c)).

Fig. 13. Examples of parts ignored in the computation of $d_{e,e'}$, as indicated by dotted lines. (a) Basic edge e and unaligned deviating edge e' . The staircase of e' is already known. (b) Two evading edges. (c) Aligned deviating edge e . (d) Unaligned deviating edge e .

- If e is an unaligned deviating edge, then e' is either an unaligned basic edge or an evading edge. A fraction of $(s_e/2 - 1)/(s_e - 1)$ evades the other edge, where s is the computed number of steps for e . Since this fraction is increasing with s and there is a minimum of 4 steps, the minimal fraction is $1/3$. Regardless of the class of e' , we ignore a fraction of $1/3$ of e . This is illustrated in Figure 13(d).

The edge distance depends on $s_{e'}$ if e is an unaligned basic or evading edge and e' is an unaligned deviating edge. Since $s_{e'}$ can be computed for e' without dependencies, this poses no problem. We first compute the distances and step numbers for all deviating edges, followed by the computation for the remaining edges.

Step number. Consider an edge $e = (v, w)$ incident to a significant vertex v and insignificant vertex w . The number of steps s_e for $e = (v, w)$ depends on its classification. The following cases present the computation.

- For basic and evading edges, the maximal distance between the edge and its staircase is attained in the apex of a step. Let α_1 denote the absolute angle between vector $w - v$ and the assigned direction of e . Similarly, α_2 denotes the absolute angle between vector $w - v$ and the other associated direction of e . Basic computations show that a step must span less than $l_{\max} = ((\tan \alpha_1)^{-1} + (\tan \alpha_2)^{-1})d_e/2$ to avoid deviating more than a distance $d_e/2$ from e . Hence, the number of steps is computed as $\lceil \|e\|/l_{\max} \rceil_2$, where $\lceil x \rceil_2$ denotes the smallest even integer that is strictly greater than x . Note that the staircase may not be at exactly distance $d_e/2$ from e ; hence, if x is an even integer, $\lceil x \rceil_2$ equals $x + 2$.
- An aligned deviating edge does not use the step number. Instead it requires a distance δ_e . This distance is an upper bound on the maximal distance between the staircase of e and e itself. Hence, we use $\delta_e = \min \{d_e/2, \Delta_e\}$, where $\Delta_e = 0.1\|e\|$ as defined for the staircase regions.
- For an unaligned deviating edge, the maximal distance to e is attained in one of the corners of the appended region. The easiest way to compute s_e is by first computing the maximal distance to e when using a step of unit length and the corresponding appended region. Let d_1 denote this maximal distance for e . Scaling to a different step length scales the entire step and appended region uniformly. Hence, we find that $d_f = f \cdot d_1$ when scaling by a factor f . Since f corresponds directly to the step length, we find that the maximum step length is $d_e/(2d_1)$. From this, we can derive that $s_e = \max \{4, \lceil 2d_1\|e\|/d_e + 1 \rceil_2\}$.

4.4. Analysis

Let us now analyze the complexity of result R with respect to the complexity of the given subdivision S . Since every edge of the input is converted into a staircase, $|R|$ is at least as big as $|S|$. Often, consecutive edges may have oppositely assigned directions at their shared vertex. In such cases, the shared vertex becomes superfluous in the resulting subdivision. Removing these vertices allows $|R|$ to become smaller than $|S|$, though this is unlikely to occur in practice.

Depending on the vertex-edge distances and the angles between edges with a shared vertex, the staircases use fewer or more steps. The smaller the vertex-edge distances or angles become, the more steps a staircase needs such that no intersections are introduced. Most importantly, the number of steps for a single edge does not depend on $|S|$. Hence, $|R|$ is worst-case linear in $|S|$. A theoretic upper bound can be computed based on minimal angles, lengths, and distances. However, the increase predicted by such an upper bound is far greater than the increase observed in practice. Deviating edges especially may cause a large increase (locally) in the number of edges in worst-case conditions. These are caused by vertices of degree 3 or 4 such that all edges lie

Table I. The increase in complexity caused by Algorithm 2 for the various inputs used in Section 5. The increase is given as a percentage of the output complexity, $|R|$, with respect to the input complexity, $|S|$. Last two columns indicate computation time (wall-clock time, on a Microsoft Surface Pro 3 i5 8GB) for both algorithms in computing the result of the indicated figure.

Figure	Input S	$ S $	$\mathcal{C}$	$ R $	Increase	Alg. 2	Alg. 1
Figure 14	Languedoc-Roussillon	847	$\mathcal{C}_4$	2 046	242%	2.3s	50.4s
Figure 15	Antartica	225	$\mathcal{C}_3$	686	305%	0.2s	6.1s
			$\mathcal{C}_3(\frac{\pi}{2})$	735	327%	0.2s	6.8s
			$\mathcal{C}_6$	618	275%	0.2s	5.5s
Figure 16	Australia	223	$\mathcal{C}_2$	682	306%	0.2s	5.1s
			$\mathcal{C}_3$	580	260%	0.1s	4.0s
			$\mathcal{C}_4$	597	268%	0.1s	4.6s
			$\mathcal{C}_5$	616	276%	0.2s	4.6s
Figure 17	Great Britain	549	$\mathcal{C}_3$	1 808	329%	1.0s	37.6s
			$\mathcal{C}_4$	1 369	249%	0.4s	24.9s
			$\{\frac{\pi}{12}, \frac{5\pi}{12}, \frac{7\pi}{12}\}$	2 428	442%	1.3s	64.2s
Figure 18	Italy	1 297	$\mathcal{C}_4$	3 020	233%	2.9s	24.6s
Figure 19	Japan	2 000	$\mathcal{C}_4$	5 286	264%	11.2s	203.3s
			$\mathcal{C}_6$	5 494	275%	11.3s	221.4s
Figure 20	Southeast Asia	250	$\mathcal{C}_4$	658	263%	0.3s	2.5s
			$\{\frac{\pi}{12}, \frac{\pi}{2}, \frac{11\pi}{12}\}$	719	288%	0.2s	2.4s
Figure 21	The Netherlands	1 506	$\mathcal{C}_4$	3 804	253%	5.2s	110.3s

in the same sector or same adjacent sectors. This situation rarely occurs for territorial outlines.

Table I lists the increase in complexity caused by Algorithm 2 for the territorial outlines that are used in Section 5. To measure the increase, the table contains both the complexity of the input subdivision as well as the complexity of the $\mathcal{C}$ -oriented subdivision as computed by the orientation-restriction algorithm. The latter indicates the complexity, after removing the beforementioned superfluous vertices. In these experiments, the increase was on average 286% and never exceeded 450%.

The orientation-restriction algorithm described in this section executes in $O(|S|^2 + |R|)$ time. This assumes a simple implementation in which the interfering edges are found by iterating over the other edges (Algorithm 2, Line 5). This may possibly be improved upon by using a more intelligent search. However, note that an improvement here does not yield an asymptotic improvement, when combining Algorithm 2 with Algorithm 1 which then runs in $O(|R|^2)$ time.

5. DISCUSSION

We have implemented the algorithms described in this paper and we used them to generate results for various territorial outlines. Here we present and briefly discuss results for single outlines (polygons) and multiple outlines (subdivisions). There are three different types of results, indicated in the figures with different colors. Simplification results are computed using only Algorithm 1. Schematization results are obtained by applying the simplification algorithm to the result of Algorithm 2.

Polygons. Figure 14 shows a sequence of octilinear schematizations for Languedoc-Roussillon (a department of France), starting from its orientation restriction to the final convex polygon. The first results (a–c) are barely distinguishable from the input. The results with medium complexity (d–f) are clearly recognizable as schematizations and have a shape that corresponds well to the original geometry. Even at low complexity (g), the result is very reasonable, given the constraints. The final result is a convex polygon with five edges (h). On this extremely low complexity, the result deteriorates; a better schematization is illustrated. Many shapes exhibit this deterioration at extremely low complexities. Hence, for the other examples, we shall showcase only results with a few more edges.

Fig. 14. Octilinear schematization results with decreasing complexity for Languedoc-Roussillon. (a) Orientation restriction, 2 046 edges (up from 847). (b) 499 edges. (c) 250 edges. (d) 100 edges. (e) 50 edges. (f) 24 edges. (g) 11 edges. (h) 5 edges. An alternative, superior solution is sketched by the dashed polygon.

Fig. 15. Results for Antarctica with 25 edges. (a) Input. (b) Simplification. (c) Hexilinear, C_3 . (d) Hexilinear, $C_3(\frac{\pi}{2})$. (e) Dodecilinear, C_6 .

Fig. 16. Results for Australia with (at most) 25 edges. (a) Input. (b) Simplification. (c) Rectilinear, C_2 . (d) Hexilinear, C_3 . (e) Octilinear, C_4 . (f) Decilinear, C_5 .

Figure 15 shows results for the continent of Antarctica. Two schematizations use hexilinear orientation sets with different initial angles (c–d). In general, the horizontal variant (c) better represents the upper and lower boundary; the vertical variant (d) better represents the left and right boundary. However, combining the two sets into a dodecilinear orientation set yields a result (e) that more resembles a simplification than a C -oriented schematization. We conclude that simply providing more orientations does not necessarily imply a better schematization.

Figure 16 shows results for Australia using a variety of regular orientation sets. The rectilinear result is possibly hard to recognize separately, but a good schematization considering the constraints. The other schematizations are closer to the original shape. Especially the hexilinear and decilinear schematizations work well. However, the octilinear variant seems slightly worse. In particular, the Gulf of Carpentaria in the north is represented worse.

Figure 17 shows various results for Great Britain. We consider the hexilinear schematization to be the best result. The octilinear schematization requires multiple edges to represent the rather straight eastern coastline. The result with the irregular set works quite well considering its restrictions. However, a small edge remains on the southern coast. These edges could probably have been put to better use by giving some more detail to Wales instead. Also, the Thames estuary is represented only slightly.

Fig. 17. Results for Great Britain with 50 edges. (a) Input. (b) Simplification. (c) Hexilinear, C_3 . (d) Octilinear, C_4 . (e) Irregular, $\{\frac{\pi}{12}, \frac{5\pi}{12}, \frac{7\pi}{12}\}$.

Subdivisions. Figure 18 shows a simplification and octilinear schematization of the regions of Italy. The first results (b–c) use a comparatively high number of edges for schematization purposes. However, this is unavoidable to maintain topology and some resemblance, since there are many borders to be represented. Our algorithm is able to further reduce the complexity (d–e), but resemblance suffers. Especially for the simplified result (d), deformation in the southern regions of Puglia and Calabria greatly reduces the characteristic ‘boot shape’ of Italy. In the schematized result (e), this is less severe though the northwestern region of Liguria becomes too narrow.

Figure 19 shows a simplification and schematizations of the four major islands of Japan. As with Antarctica, the dodecilinear result (d) more resembles a simplified outline. Despite its orientation restrictions—or perhaps because of it—the dodecilinear result approximates some parts better than the simplification. For example, the southernmost island (Kyushu) retains more of its shape and the southern shore of the largest island (Honshu) has a better representation.

Figure 20 (also shown in the introduction) shows results for a group of countries in southeast Asia (Cambodia, Laos, part of Malaysia, Myanmar, Thailand, and Vietnam). Both schematizations (and to a lesser extent the simplification) have made the narrow

Fig. 18. Results for the regions of Italy with 244 edges. (a) Input. (b) Simplification with 244 edges. (c) Octilinear, C_4 , with 244 edges. (d) Simplification with 206 edges. (e) Octilinear, C_4 , with 153 edges.

Fig. 19. Results for four islands of Japan with 120 edges. (a) Input. (b) Simplification. (c) Octilinear, C_4 . (d) Dodecilinear, C_6 .

Fig. 20. Results for a group of countries in southeast Asia with 130 edges. (a) Input. (b) Simplification. (c) Octilinear, C_4 . (d) Irregular set, $\{\frac{\pi}{12}, \frac{\pi}{2}, \frac{11\pi}{12}\}$.

part of Thailand even more narrow, even so far as to suggest that these are actually two separate regions. Though it is geometrically accurate, legibility would be improved by slightly exaggerating this strip. In addition, the local ‘continuation’ changed for some of the degree-3 vertices, for example, at the southernmost vertex of the Myanmar-Thailand border in the octilinear schematization. We do not consider this to be very problematic. If so desired, it can likely be counteracted by restricting edge-moves near degree-3 vertices.

In Figure 21 we show our schematization result for the Netherlands, at equal complexity as the manual schematization given in Figure 1(a). For comparison, we also show only the outline of this manual schematization. Note that, for example, the exaggeration of southern Limburg is not possible with our algorithm.

Fig. 21. Result for the Netherlands. (a) Input. (b) Octilinear, C_4 , with 240 edges. (c) Outline extracted from Figure 1(a); this outline has 240 edges.

6. CONCLUSIONS

We studied the problem of simplifying and schematizing territorial outlines under area- and topology-preservation constraints. To this end, we introduced an operation called an edge-move that can be used for simplification. In our non-vertex-restricted simplification algorithm, we perform these in pairs such that the complexity of the outline is reduced and the area and topology are indeed maintained. We proved that any nonconvex polygon allows such a pair of edge-moves.

Since edge-moves do not alter orientations, the simplification algorithm is also suitable to compute C -oriented schematizations. If we desire an output that is C -oriented, then we need to ensure only that the input for the simplification algorithm is C -oriented as well. Hence, we introduced an area-preserving algorithm to convert any subdivision into a topologically equivalent, C -oriented subdivision. Combining these two algorithms, we obtain an algorithm to schematize any simple polygon or subdivision. We observe that in some cases it is actually desirable to introduce an orientation, as is illustrated in Figure 22.

Fig. 22. (a) A rectilinear ‘triangle’. (b) Any octilinear result of our algorithm uses only rectilinear orientations. (c) A proper octilinear schematization, unattainable by our current algorithm.

We illustrated that the results of these algorithms preserve the important structures of the input. However, we assumed that the desired orientations are given. The quality of schematization depends quite strongly on the chosen orientation set $\mathcal{C}$. Hence, it is desirable to design automated ways to decide what orientation set is suitable for a given territorial outline. Similarly, the desired complexity k is treated as a given. Can we automatically determine when to stop the simplification process? Note that due to the iterative nature of our algorithm, we may store intermediate results by keeping track of the local changes without asymptotic overhead (similar to Van Dijk et al. [2014]). This allows for efficiently retrieving schematizations with varying levels of complexity after a single execution of our algorithm.

Our algorithm chooses operations that change the area as little as possible. That is, we greedily choose the contraction that causes the least symmetric difference with respect to the current result. While this strategy allows for an efficient algorithm, it may not always be the ‘best’ choice for the complete simplification. That is, choices that are ‘worse’ according to the symmetric difference may lead to a better solution in the end. Moreover, it might lead to an asymmetric result for a symmetric input. Hence, other strategies or other criteria might be more appropriate. For example, Haunert [2012] presents an algorithm for detecting symmetries in buildings and building groups. Is such a method effective also for territorial outlines? If we can augment territorial outlines with symmetry information, how can we incorporate this into our schematization algorithm?

To improve upon the greedy nature of the iterative algorithm, one may consider a strategy to optimize a common distance measure. However, as argued in Appendix A, such optimization may result in unsatisfying simplifications or schematizations as well. Nonetheless, we observe that not all counterexamples are reachable by edge-moves, while the better result is. For example, this is the case for the symmetric difference. The question rises whether it is desirable to find a result that minimizes the symmetric difference over all results that can be obtained using edge-moves. If so, is there an efficient algorithm to compute it? Our greedy approach does not minimize the symmetric difference. Moreover, an optimal result for a given complexity k does not necessarily lead to an optimal result for lower complexity.

Finally, future work may include establishing benchmark data sets for simplification and performing a comparative study between the various simplification methods in literature for their merits and performance in terms of quality and execution time.

ACKNOWLEDGMENTS

The authors would like to thank Marc van Kreveld for helpful discussions on the topic of this article. K. Buchin is partially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.001.207. W. Meulemans and B. Speckmann were supported by NWO under project number 639.022.707. W. Meulemans is currently supported by the EU Horizon 2020 programme, under a Marie Skłodowska-Curie Individual Fellowship, project number 656741. B. Speckmann is currently supported by NWO under project number 639.023.208. A. van Renssen was supported in part by NSERC and Carleton University’s President’s 2010 Doctoral Fellowship and is currently supported by JST, ERATO, Kawarabayashi Large Graph Project.

REFERENCES

- M. A. Abam, S. Daneshpajouh, L. Deleuran, S. Ehsani, and M. Ghodsi. 2014. Computing homotopic line simplification. *Computational Geometry: Theory and Applications* 47, 7 (2014), 728–739.
- E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. 1991. An Efficiently Computable Metric for Comparing Polygonal Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13, 3 (1991), 209–216.

- T. Barkowsky, L. J. Latecki, and K.-F. Richter. 2000. Schematizing Maps: Simplification of Geographic Shape by Discrete Curve Evolution. In *Spatial Cognition II (LNCS 1849)*. 41–53.
- M. de Berg, M. van Kreveld, and S. Schirra. 1998. Topologically Correct Subdivision Simplification Using the Bandwidth Criterion. *Cartography and Geographic Information Science* 25, 4 (1998), 243–257.
- P. Bose, S. Cabello, O. Cheong, J. Gudmundsson, M. van Kreveld, and B. Speckmann. 2006. Area-Preserving Approximations of Polygonal Paths. *Journal of Discrete Algorithms* 4, 4 (2006), 554–566.
- K. Buchin, W. Meulemans, and B. Speckmann. 2011a. Area-Preserving C-Oriented Schematization. In *Abstracts of the 27th European Workshop on Computational Geometry*. 163–166.
- K. Buchin, W. Meulemans, and B. Speckmann. 2011b. A New Method for Subdivision Simplification with Applications to Urban-Area Generalization. In *Proceedings of the 19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*. 261–270.
- M. E. Buchin. 2007. *On the Computability of the Fréchet Distance Between Triangulated Surfaces*. Ph.D. Dissertation. Freie Universität Berlin.
- S. Cabello, M. de Berg, and M. van Kreveld. 2005. Schematization of Networks. *Computational Geometry: Theory and Applications* 30, 3 (2005), 223–238.
- T. Christ, D. Pálvölgyi, and M. Stojaković. 2010. Consistent Digital Line Segments. In *Proceedings of the 26th Symposium on Computational Geometry*. 11–18.
- S. Cicerone and M. Cermignani. 2012. Fast and Simple Approach for Polygon Schematization. In *Proceedings of the 12th International Conference on Computational Science and Its Applications (LNCS 7333)*. 267–279.
- S. Daneshpajouh, M. Ghodsi, and A. Zarei. 2012. Computing polygonal path simplification under area measures. *Graphical Models* 74, 5 (2012), 283–289.
- D. Delling, A. Gemsa, M. Nöllenburg, and T. Pajor. 2010. Path Schematization for Route Sketches. In *Proceedings of the 12th Scandinavian Workshop on Algorithm Theory (LNCS 6139)*. 285–296.
- T. C. van Dijk, A. van Goethem, J.-H. Haunert, W. Meulemans, and B. Speckmann. 2013. Accentuating Focus Map via Partial Schematization. In *Proceedings of the 21st ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*. 438–441.
- T. C. van Dijk, A. van Goethem, J.-H. Haunert, W. Meulemans, and B. Speckmann. 2014. Map Schematization with Circular Arcs. In *Proceedings of the 8th International Conference on Geographic Information Science (LNCS 8728)*.
- D. H. Douglas and T. K. Peucker. 1973. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica* 10, 2 (1973), 112–122.
- R. Estkowski and J. S. B. Mitchell. 2001. Simplifying a Polygonal Subdivision While Keeping it Simple. In *Proceedings of the 17th Symposium on Computational Geometry*. 40–49.
- M. Fink, H. Haverkort, M. Nöllenburg, M. J. Roberts, J. Schuhmann, and A. Wolff. 2013. Drawing Metro Maps Using Bézier Curves. In *Proceedings of the 20th International Symposium Graph Drawing (LNCS 7704)*. 463–474.
- A. Gemsa, M. Nöllenburg, T. Pajor, and I. Rutter. 2011. On d-Regular Schematization of Embedded Paths. In *SOFSEM 2011: Theory and Practice of Computer Science (LNCS 6543)*. 260–271.
- A. van Goethem, W. Meulemans, A. Reimer, H. Haverkort, and B. Speckmann. 2013. Topologically Safe Curved Schematization. *The Cartographic Journal* 50, 3 (2013), 276–285.
- A. van Goethem, W. Meulemans, B. Speckmann, and J. Wood. 2014. Exploring curved schematization. In *Proceedings of the 7th IEEE Pacific Visualization Symposium*. 1–8.
- L. J. Guibas, J. E. Hersherberger, J. S. B. Mitchell, and J. S. Snoeyink. 1993. Approximating Polygons and Subdivisions with Minimum-Link Paths. *International Journal of Computational Geometry and Applications* 3 (1993), 383–415.
- J.-H. Haunert. 2012. A symmetry detector for map generalization and urban-space analysis. *ISPRS Journal of Photogrammetry and Remote Sensing* 74 (2012), 66–77.
- H. Imai and M. Iri. 1988. Polygonal Approximations of a Curve—Formulations and Algorithms. *Computational Morphology* (1988), 71–86.
- B. van de Kraats, M. van Kreveld, and M. Overmars. 1995. Printed Circuit Board Simplification: Simplifying Subdivisions in Practice. In *Proceedings of the 11th Symposium on Computational Geometry*. 430–431.
- A. Marzal and V. Palazón. 2005. Dynamic Time Warping of Cyclic Strings for Shape Matching. In *Pattern Recognition and Image Analysis (LNCS 3687)*. 644–652.
- D. Merrick and J. Gudmundsson. 2006. Path Simplification for Metro Map Layout. In *Proceedings of the 14th International Symposium on Graph Drawing (LNCS 4372)*. 258–269.

- W. Meulemans. 2014. *Similarity Measures and Algorithms for Cartographic Schematization*. Ph.D. Dissertation. Technische Universiteit Eindhoven.
- W. Meulemans, A. van Renssen, and B. Speckmann. 2010. Area-Preserving Subdivision Schematization. In *Proceedings of the 6th International Conference on Geographic Information Science (LNCS 6292)*. 160–74.
- M. E. Munich and P. Perona. 1999. Continuous Dynamic Time Warping for Translation-Invariant Curve Alignment with Applications to Signature Verification. In *Proceedings of the 7th IEEE International Conference on Computer Vision*. 108–115.
- N. Mustafa, E. Koutsofios, S. Krishnan, and S. Venkatasubramanian. 2001. Hardware-Assisted View-Dependent Map Simplification. In *Proceedings of the 17th Symposium on Computational Geometry*. 50–59.
- G. Neyer. 1999. Line Simplification with Restricted Orientations. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures (LNCS 1663)*. 13–24.
- M. Nöllenburg and A. Wolff. 2011. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Transactions on Visualization and Computer Graphics* 17, 5 (2011), 626–641.
- P. M. van der Poorten and C. B. Jones. 2002. Characterisation and Generalisation of Cartographic Lines using Delaunay Triangulation. *International Journal of Geographical Information Science* 16, 8 (2002), 773–794.
- N. Regnauld, A. Edwardes, and M. Barrault. 1999. Strategies in Building Generalisation: Modelling the Sequence, Constraining the Choice. In *Proceedings of the ICA '99 Workshop on Progress in Automated Map Generalization*.
- A. Reimer. 2010. Understanding Chorematic Diagrams: Towards a Taxonomy. *The Cartographic Journal* 47, 4 (2010), 330–350.
- A. Reimer and W. Meulemans. 2011. Parallelity in Chorematic Territorial Outlines. In *Proceedings of the 14th ICA Workshop on Generalisation and Multiple Representation*.
- M. J. Roberts. 2012. *Underground maps unravelled - Explorations in information design*. Self-published.
- J. Swan, S. Anand, M. Ware, and M. Jackson. 2007. Automated Schematization for Web Service Applications. In *Web and Wireless GISystems (LNCS 4857)*. 216–226.
- D. Tutić and M. Lapaine. 2009. Area Preserving Cartographic Line Generalization. *Cartography and Geoinformation* 8, 11 (2009), 84–100.
- B. Tversky. 1993. Cognitive maps, cognitive collages, and spatial mental models. In *Spatial Information Theory: A Theoretical Basis for GIS (LNCS 716)*. 14–24.
- M. Visvalingam and J. D. Whyatt. 1993. Line generalisation by repeated elimination of points. *The Cartographic Journal* 30, 1 (1993), 46–51.

APPENDIX

A. OPTIMIZING DISTANCE MEASURES

In this appendix, we consider the quality of area-equivalent schematizations of simple rectilinear polygons with respect to several standard distance metrics. We show examples for each of these distance measures where among two schematizations with equal complexity the one with smaller distance has a worse shape compared to the input polygon. Hence, while it is in principle desirable that a schematization has a small distance to the input, it is not true that the schematization that minimizes this distance ensures the best shape. Our examples assume that a rectilinear output is desired. Since this is a special case of $\mathcal{C}$ -oriented schematization, this conclusion extends to $\mathcal{C}$ -oriented schematization as well. The first three distance measures we discuss are also discussed by Meulemans et al. [2010].

We observe that the examples given are of a constructed nature and do not directly resemble ‘realistic input’. However, the examples are ‘robust’: small perturbations and resampling do not greatly affect the results.

Symmetric difference. The *symmetric difference*, $d_{SD}(P, Q)$, between two polygons P and Q is defined as the total area that is covered by one polygon but not by the other: it is exactly the area in which they differ from each other. Interpreting P and Q as the set of points that are in the polygon, we can express it as

$$d_{SD}(P, Q) = |P \cup Q| - |P \cap Q|.$$

Fig. 23. A polygon (a) and two area-equivalent rectilinear schematizations: with minimal symmetric difference (b), and with better shape (c).

Consider the example in Figure 23. The input is a 12-sided rectilinear polygon which we would like to schematize in an area-preserving manner with an 8-sided rectilinear polygon. The solution with minimal symmetric difference loses the vertical axis of symmetry and converts the polygon from a U-shape to a C-shape.

Hausdorff distance. The *Hausdorff distance*, $d_H(P, Q)$, measures the distance between two sets P and Q of points in the plane. It finds for each point in P the closest point in Q , and vice versa, and then takes the maximum of these distances. This distance is expressed in the following formula:

$$d_H(P, Q) = \max \left\{ \sup_{p \in P} \inf_{q \in Q} d(p, q), \sup_{q \in Q} \inf_{p \in P} d(p, q) \right\}.$$

The example in Figure 24 shows that an area-equivalent schematization with the smallest Hausdorff distance can have a worse shape than a schematization with a slightly larger distance. Furthermore, since the Hausdorff distance is determined by a maximum value, it is sensitive to outliers. The entire boundary of a polygon is considered for the Hausdorff distance. It is also possible to use only the vertices, resulting in a discrete Hausdorff distance. However, the same objections hold.

Fig. 24. A polygon (a) and two area-equivalent schematizations: with minimal Hausdorff distance (b), and with better shape (c).

Fréchet distance. The *Fréchet distance* measures the similarity of two curves by measuring the minimal maximal difference when ‘walking’ along the two curves without moving backwards. Although the Fréchet distance is defined on continuous curves it can also be applied to polygons. More formally, let C and C' be two curves. The Fréchet distance between C and C' is given as

$$d_F(C, C') = \inf_{t, t'} \max_{x \in [0, 1]} d(C_t(x), C'_{t'}(x)),$$

where C_t and C'_t are monotonous parameterizations of the curves C and C' defined by t and t' respectively. The Fréchet distance is sensitive to outliers. Consider the example in Figure 25 where an 8-sided polygon is approximated by an area-equivalent rectangle. The solution with the smallest Fréchet distance tries to approximate the thin part, while ‘ignoring’ most of the polygon. In the example, the thin part has a width of 1, but the width can be arbitrarily small without significantly changing the result that is optimal with respect to the Fréchet distance.

Fig. 25. A polygon (a) and two area-equivalent schematizations: with minimal Fréchet distance (b), and with better shape (c).

To alleviate the sensitivity to outliers, we could look for an average or summed Fréchet distance. However, this approach also has its disadvantages, as was shown by Buchin [2007]. Buchin describes and compares multiple intuitive definitions of the average Fréchet distance. All variants have disadvantages: most importantly, all considered definitions do not fulfill the triangle inequality. Even worse, they also do not fulfill the relaxed triangle inequality, that is $d(X, Z) < c \cdot (d(X, Y) + d(Y, Z))$ for some constant $c > 0$, where $d(P, Q)$ denotes the average Fréchet distance between polygons P and Q . Furthermore, it is an open problem how to compute the continuous average Fréchet distance.

Turning angle distance. The *turning angle distance* is defined by Arkin et al. [1991]. Roughly, it is defined as follows. For a curve C , its turning function $\Theta_C(x)$ is defined for $x \in [0, 1]$, expressing the angle of the direction a fraction of x (with respect to the curve length) along the curve with respect to some reference angle. The turning angle distance between two curves C and C' is expressed as

$$\delta_{TA}(C, C') = \left(\int_0^1 |\Theta_C(x) - \Theta_{C'}(x)|^p dx \right)^{\frac{1}{p}},$$

for some p . The use of $p = 2$ is suggested by Arkin et al. [1991]. For polygons, the distance is the minimum over all possible ways to ‘cut’ these into chains and all rotations of C' . This distance is designed to be independent of scale, translation and rotation. This already indicates that the distance measure is likely not suitable in our setting, though scaling is less of a problem due to area preservation. However, the authors also indicate that the measure is sensitive to non-uniform noise. While in their setting, (near-)uniform noise may be a reasonable assumption, we do not wish to put such restriction on the input. Figure 26 shows an example of how non-uniform ‘noise’ can influence the optimal solution with respect to the turning angle distance. The more ‘battlements’ we add to the shape, the worse this effect becomes, tending towards a square for a shape with minimal turning angle distance.

Cyclic dynamic time warp distance. The *cyclic dynamic time warp distance* [Marzal and Palazón 2005] is the dynamic time warp distance adapted for polygons. In essence,

Fig. 26. A polygon (a) and two area-equivalent schematizations: with minimal turning angle distance (b), and with better shape (c). The turning functions are given.

this is the discrete summed Fréchet distance, but instead of using the Euclidean distance between points, any function can be used to evaluate the distance. Marzal and Palazón [2005] suggest to use curvature. This causes the method to be invariant to rotation, scale and translation. Like with the turning angle distance, this is undesirable for our purpose. However, using for example the Euclidean distance also results in undesirable behavior, as is shown in Figure 27. The problem is caused by the discrete steps from vertex to vertex and the uneven sampling between polygon and its approximation. Continuous versions of the dynamic time warping distance have also been considered [Munich and Perona 1999]. However, the version presented uses a different distance measure between points on the curve, such that the distance between curves is invariant under translation. Again, we consider this to be undesirable for simplification. Using the Euclidean distance instead results in a summed Fréchet distance.

Fig. 27. A polygon (a) and two area-equivalent approximations: with minimal cyclic dynamic time warp distance (b), and with better shape (c).

B. PROOF: EXISTENCE OF FEASIBLE CONFIGURATIONS IN POLYGONS

In Section 3.3, we claimed the theorem below. In this appendix, we provide the full proof of this claim.

THEOREM 3.2. *Every nonconvex simple polygon has a nonconflicting pair of complementary configurations.*

Definitions and notation. In order to prove the theorem, we first introduce some additional terminology and notation. We define a *chain* $\mathcal{Y}$ as a sequence of at least three consecutive edges of simple polygon P . Its edges are denoted by $\langle s_1, \dots, s_m \rangle$ where m is the number of edges in $\mathcal{Y}$. Its vertices are denoted by $u_0, \dots, u_m$ and edge s_i is directed from u_{i-1} to u_i . The edges s_1 and s_m are the *outer edges* of $\mathcal{Y}$; the other edges are its *inner edges*. Likewise, u_0 and u_m are outer vertices and the other vertices are inner vertices. By $\alpha(\mathcal{Y})$, we denote the sum of the exterior angles of the inner vertices of $\mathcal{Y}$.

A *lid* is an open line segment between a point on s_1 (strictly before u_1) and a point on s_m (strictly after u_{m-1}) and is fully contained in the interior of P . If $\mathcal{Y}$ has any lid, it is a *closable chain*. If the open line segment (u_0, u_m) is a lid, $\mathcal{Y}$ is a *proper chain*. For a closable chain $\mathcal{Y}$ and a lid l , we denote by $R_l(\mathcal{Y})$ the region enclosed by $\mathcal{Y}$ and l . For a

Fig. 28. (a) Unclosable chain. (b) Closable chain with a lid l (dashed) and $R_l(\mathcal{Y})$ (shaded gray). (c) Proper chain with its unique lid (dashed) and $R(\mathcal{Y})$ (shaded gray).

proper chain, $R(\mathcal{Y})$ denotes this region using the lid (u_0, u_m) implicitly. Due to the lid, we know that for every closable chain, any point on ∂P that is inside $R_l(\mathcal{Y})$ must be part of $\mathcal{Y}$. Some examples are given in Figure 28. A configuration is a (not necessarily closable) chain of length three.

Proving the theorem. We prove Theorem 3.2 via a sequence of six lemmas. The first two lemmas state basic properties of closable chains.

LEMMA B.1. *For any closable chain $\mathcal{Y}$, $\alpha(\mathcal{Y}) > 0$ holds.*

PROOF. Since $\mathcal{Y}$ is a closable chain, it must have some lid l . Let u and v denote the endpoints of l . Endpoints u and v and the inner vertices of $\mathcal{Y}$ define a simple polygon P' that corresponds to $R_l(\mathcal{Y})$. We know that the sum of the exterior angles for any simple polygon is exactly 2π . Hence, we find that $\alpha(P') = 2\pi = \alpha(\mathcal{Y}) + \alpha(u) + \alpha(v)$. Moreover, we know that the exterior angle of any vertex is strictly less than π . Hence, we may conclude that $\alpha(\mathcal{Y}) = 2\pi - \alpha(u) - \alpha(v) > 0$. $\square$

LEMMA B.2. *Let $\mathcal{Y}$ be a closable chain without a convex inner edge such that the first inner vertex is reflex. Then $\mathcal{Y}$ contains a configuration X such that the first inner vertex is reflex and $\alpha(X) > 0$.*

PROOF. Since $\mathcal{Y}$ is closable, we know that $\alpha(\mathcal{Y}) > 0$ (Lemma B.1). By assumption we know that $\mathcal{Y}$ contains no two consecutive convex inner vertices and that its first inner vertex is reflex. Hence, we may characterize the inner vertices of $\mathcal{Y}$ by k sequences of one or more reflex vertices followed by a single convex vertex. Let ς_i denote the i^{th} sequence. Slightly abusing notation, let $\alpha(\varsigma_i)$ denote the sum of exterior angles of *all* the vertices of sequence ς_i . Now we know that $\sum_{i=1}^k \alpha(\varsigma_i) = \alpha(\mathcal{Y}) > 0$. In particular, this implies that $\alpha(\varsigma_i) > 0$ holds for at least one sequence. Since reflex vertices have a negative exterior angle, this implies that the exterior angle of the single convex vertex is bigger than the reflex vertex preceding it. Hence, at the end of a sequence ς_i with $\alpha(\varsigma_i) > 0$, we find configuration X such that the first inner vertex is reflex and $\alpha(X) > 0$. $\square$

We now prove the existence of a feasible negative configuration, that is, a configuration X for which the negative contraction region $R^-(X)$ is empty. We provide two lemmas assuming different properties of a chain (Lemma B.3 and Lemma B.4).

LEMMA B.3. *Let $\mathcal{Y} = \langle s_1, \dots, s_m \rangle$ be a closable chain without a convex inner edge such that the first inner vertex is reflex. Then $\mathcal{Y}$ has a feasible negative configuration X such that the first inner vertex is reflex, $\alpha(X) > 0$ and $R^-(X) \subseteq R_l(\mathcal{Y})$ for any lid l of $\mathcal{Y}$.*

PROOF. We prove this lemma by induction. For the base case, assume that $m = 3$. Configuration $X' = \langle s_1, s_2, s_3 \rangle$ has a first inner vertex that is reflex. Any lid l of $\mathcal{Y}$ must be on one side of the line through s_1 , whereas this track enforces a triangular contraction region on the other side. Hence, $R^-(X') \subseteq R_l(\mathcal{Y})$ holds and X' is a feasible negative configuration. Figure 29 shows closable and, for comparison, unclosable configurations.

Fig. 29. Configurations with $\alpha > 0$; the first inner vertex is reflex. (a) Closable chains. (b) Unclosable chains.

For the inductive step, we use as induction hypothesis that this lemma holds for any closable chain with less than m edges. Let $X' = \langle s_{i-1}, s_i, s_{i+1} \rangle$ be the first configuration such that the first inner vertex is reflex and $\alpha(X') > 0$. This implies that the second inner vertex is convex. From Lemma B.2, we conclude that $\mathcal{Y}$ must contain such a configuration. If X' is feasible, we are done. If X' is not feasible, let p denote the corresponding blocking point. We prove that p is part of $\mathcal{Y}$ and comes after s_{i+1} .

Refer to Figure 30(a–b). Chain $\mathcal{Y}$ is closable and hence has some lid l that does not intersect any part of $\mathcal{Y}$ (dotted in the figure). Suppose l intersects $R^-(X')$, the contraction region of X' . Since the contraction region is bounded by s_i and s_{i+1} of $\mathcal{Y}$ on two sides, we know that the lid must intersect the track of s_{i-1} in some point q . Note that this is actually possible, as illustrated in Figure 30(a). However, this implies that some other part of $\mathcal{Y}$ also intersects this track and it does so before q (indicated with a dashed line in the figure). We prove this by contradiction. Assume that there is no intersection between $\mathcal{Y}$ and the track of s_{i-1} that lies between s_{i-1} and q . This implies that the endpoint of l on s_1 , the chain $\langle s_2, \dots, s_{i-2} \rangle$ and q define a simple polygon (indicated with a hachured-shaded region in Figure 30(b)). Since the sum of exterior angles of a simple polygon is 2π and u and q have an exterior angle strictly smaller than π , the sum of exterior angles in the inner vertices $\langle s_1, \dots, s_{i-1} \rangle$ is positive. However, this contradicts that X' is the first configuration such that the first inner vertex is reflex and $\alpha(X') > 0$. Therefore, there must be an intersection of $\mathcal{Y}$ that is closer to the inner edge of X' than intersection q . In particular, this also means that anything outside of $R_l(\mathcal{Y})$ that lies in $R^-(X')$ cannot be the blocking point; this area is shaded dark gray in Figure 30(b).

Now that we have concluded that blocking point p of X' is part of $\mathcal{Y}$, we must argue that it comes after s_{i+1} . Refer to Figure 30(c). If blocking point p occurs along an edge prior to X' , then we may again construct a simple polygon using p and the edges after p up to s_{i-1} (indicated with a hachured-shaded region in the figure). Since p and the first vertex along $\mathcal{Y}$ after p have an exterior angle smaller than π , we again conclude that the sum of exterior angles leading up to s_{i-1} is positive, contradicting the definition of X' . We conclude that blocking point p must come after s_{i+1} on $\mathcal{Y}$.

Fig. 30. Negative configuration X' with $R^-(X')$ shaded gray. The lid of the proper chain is dotted. (a) Intersection between the lid and $R^-(X')$ is possible, but $R^-(X')$ must then be crossed by some part of the chain (dashed) after X' . (b) If the hachured-shaded polygon is simple, we derive a contradiction on the definition of X' . If q exists, the darker part of $R^-(X')$ cannot contain the blocking point. (c) Blocking point p must come after X' . Otherwise, the hachured-shaded polygon contradicts the definition of X' . (d) Chain $\mathcal{Y}'$ (thick line) defined by p .

Since p is a point on $\mathcal{Y}$ after s_{i+1} , we consider the closable chain $\mathcal{Y}' = \langle s_{i+1}, \dots, p \rangle$, that is, until the edge containing p or incident to p if p is a vertex. Chain $\mathcal{Y}'$ is illustrated in Figure 30(d). Since $\mathcal{Y}$ does not contain convex edges, we know that $\mathcal{Y}'$ does not have convex edges and that its first inner vertex is reflex. Since $\mathcal{Y}'$ also has less edges than $\mathcal{Y}$, we know from the induction hypothesis that $\mathcal{Y}'$ has a feasible negative configuration. $\square$

LEMMA B.4. *Let $\mathcal{Y} = \langle s_1, \dots, s_m \rangle$ denote a proper chain with a convex inner edge. Then $\mathcal{Y}$ has a feasible negative configuration $X = \langle s_{i-1}, s_i, s_{i+1} \rangle$ with $R^-(X) \subseteq R(\mathcal{Y})$ and $\alpha(X) > 0$. In addition, at least one of the following conditions hold for X :*

- (1) x_i is a convex edge; or
- (2) the first inner vertex of X is reflex; or
- (3) $s_i \neq s_{m-1}$, the inner edge of X is not the last inner edge of $\mathcal{Y}$.

PROOF. We prove this lemma by induction on m , the number of edges in $\mathcal{Y}$. For the base case, assume $m = 3$. Since $\mathcal{Y}$ is a proper chain and s_2 is a convex edge, $\langle s_1, s_2, s_3 \rangle$ is a configuration satisfying the conditions for X : since s_2 is convex, $\alpha(\langle s_1, s_2, s_3 \rangle) > 0$ holds and condition 1 is met.

For the inductive step, we use as induction hypothesis that this lemma holds for any suitable chain with less than m edges. Let s_j be a convex inner edge. Hence, $X' = \langle s_{j-1}, s_j, s_{j+1} \rangle$ is a negative configuration. If X' is feasible, we are done since a convex edge implies $\alpha(X') > 0$ and condition 1 is met. If X' is not feasible, then there is a blocking point that is in fact a vertex of $\mathcal{Y}$. This follows from the fact that the inner edge of X' is convex: this implies $R^-(X') \subseteq R(\mathcal{Y})$ and that $R^-(X')$ is bounded on three sides by the edges of X' . Let u_k denote the blocking vertex and assume that $k > j + 1$. Consider the proper chain $\mathcal{Y}' = \langle s_j, \dots, s_{k-1} \rangle$ (see Figure 31(a)). If $\mathcal{Y}'$ has a convex inner edge, it must have a feasible negative configuration satisfying the conditions for X by induction. However, if it does not, $\mathcal{Y}'' = \langle s_{j+1}, \dots, s_{k-1} \rangle$ is a closable chain such that its first inner vertex is reflex and it has no convex inner edge. Note that the direction of $\mathcal{Y}''$ should be reversed when $k < j - 1$. By Lemma B.3, $\mathcal{Y}''$ has a feasible negative configuration X'' . Moreover, the first inner vertex of X'' is reflex in the direction of $\mathcal{Y}''$. Hence, condition 2 is met.

Fig. 31. (a) Chain $\mathcal{Y}'$ defined by blocking vertex u_k and the last outer edge of X' . (b) Reversed chain $\mathcal{Y}''$ defined by blocking vertex u_k and the first outer edge of X' .

If $k < j - 1$, we use a reversed chain $\mathcal{Y}'' = \langle s_{j-1}, \dots, s_{k+1} \rangle$ (see Figure 31(b)). We may follow the same line of argument. The first inner vertex in the direction of $\mathcal{Y}''$ of the found configuration may be reflex: condition 2 will not hold in the direction of $\mathcal{Y}$. However, we know that s_m is not part of $\mathcal{Y}''$ and thus s_{m-1} cannot be the inner edge of X'' . Hence, X'' meets condition 3. $\square$

Now that we know that feasible negative configurations exist, we need a feasible positive configuration to make a proper configuration pair. The existence of a feasible positive configuration is proven in Lemma B.5. However, not any positive configuration suffices: it must not conflict with the negative one. We prove in Lemma B.6 that such a proper configuration pair indeed exists.

LEMMA B.5. *Every simple nonconvex polygon P has a feasible positive configuration X with $\alpha(X) < 0$ or all positive configurations are feasible.*

PROOF. If P has a reflex edge, then let X be a configuration with a reflex inner edge. If X is feasible, we are done as $\alpha(X) < 0$. If it is not, we can define a chain $\bar{\mathcal{Y}}$ that is inverted: the interior of $\bar{\mathcal{Y}}$ is in fact the exterior of the polygon. This is done as follows.

We cut polygon P into two chains $\mathcal{Y}_1$ and $\mathcal{Y}_2$ using the blocking point and one inner vertex of X . For both, the line segment l between the blocking point and the inner vertex can be considered a proper lid as it does not cross any part P . Hence, the enclosed regions $R(\mathcal{Y}_1)$ and $R(\mathcal{Y}_2)$ are well defined. One of these enclosed regions is fully outside polygon P (i.e., their interiors are disjoint); the other encloses both P and the region enclosed by the other chain. The former is the inverted chain that we use. Figure 32(a–b) indicates the inverted chain for both inner vertices with a thick line; the other chain is indicated with a thin line.

Fig. 32. Feasible positive configurations are found using inverted chains. The dotted line is the lid used to define an inverted proper chain. (a–b) The inverted chains $\bar{\mathcal{Y}}_a$ and $\bar{\mathcal{Y}}_b$ when P has a reflex edge. (c) The inverted chain $\bar{\mathcal{Y}}_c$ when P has no reflex edge.

We use both inner vertices of X to obtain two inverted chains. Both of these are in fact proper chains and one is a single edge longer than the other. Let $\bar{\mathcal{Y}}_a$ denote the longer and $\bar{\mathcal{Y}}_b$ the shorter of the chains. If $\bar{\mathcal{Y}}_a$ has a convex inner edge (i.e., a reflex edge in the polygon), then Lemma B.4 states that a feasible negative configuration X^- with $\alpha(X^-) > 0$ exists in $\bar{\mathcal{Y}}_a$. Since $\bar{\mathcal{Y}}_a$ is inverted, this corresponds to a feasible positive configuration X^+ in P with $\alpha(X^+) < 0$. If $\bar{\mathcal{Y}}_a$ does not have a convex inner edge, then $\bar{\mathcal{Y}}_b$ must have a first inner vertex that is reflex (with respect to $R(\bar{\mathcal{Y}}_b)$). Hence, we can now apply Lemma B.3 to $\bar{\mathcal{Y}}_b$ in a similar way to conclude that there is a feasible positive configuration X^+ in P with $\alpha(X^+) < 0$.

If P has no reflex edge, then let X be an infeasible positive configuration. If none exists, all positive configurations are feasible. Otherwise we can define an inverted chain using X . Let $\bar{\mathcal{Y}}_c$ be the inverted proper chain obtained by using the lid defined by the blocking point of X and its reflex vertex (see Figure 32(c)). Note that, $\bar{\mathcal{Y}}_c$ must start with a reflex vertex (a convex vertex in P), otherwise, P would have a reflex edge. Thus, by Lemma B.3, P has a feasible positive configuration X^+ with $\alpha(X^+) < 0$. $\square$

LEMMA B.6. *Every simple nonconvex polygon P has a proper configuration pair.*

PROOF. Recall that a proper configuration pair is a pair of complementary nonconflicting configurations; two configurations conflict when they share an edge, unless they share only outer edges and one of these has a convex and a reflex vertex (see Section 3.1).

From Lemma B.5, we conclude that polygon P has a feasible positive configuration $X^+ = \langle e_{i-1}, e_i, e_{i+1} \rangle$. Assume without loss of generality that the second inner vertex of X^+ , p_{i+1} , is reflex. Let p_j denote the first convex vertex after p_{i+1} . Configuration $X^- = \langle e_{j-1}, e_j, e_{j+1} \rangle$, of which p_j is the first inner vertex, is negative. We distinguish two cases.

Case 1: Assume that X^- is feasible. We have a proper configuration pair (X^-, X^+) if they do not conflict. To investigate this, we distinguish the following three cases; refer to Figure 33 for illustrations.

- (a) If no edge is shared, then the configurations do not conflict: (X^-, X^+) is a proper configuration pair.
- (b) If outer edge $e_{i+1} = e_{j-1}$ is shared, then this shared edge has a convex and a reflex vertex. Hence, (X^-, X^+) is a proper configuration pair.
- (c) If outer edge $e_{i-1} = e_{j+1}$ is shared but the other outer edge is not, then $p_j, p_{j+1} = p_{i-1}$, and p_i are the only convex vertices in P and there is at least one edge in between e_i and e_{j-1} . This edge is the inner edge of a feasible positive configuration, one that does not conflict with X^- . As a result, this feasible configuration constitutes a proper configuration pair with X^- .

Fig. 33. Three subcases if X^- is feasible (case 1). (a) No edge is shared. (b) Edge $e_{i+1} = e_{j-1}$ is shared. (c) Edge $e_{i-1} = e_{j+1}$ is shared.

Case 2: Now assume that X^- is not feasible. By construction, the blocking point cannot be in between v_i and v_{j+1} . If X^- is blocked by a vertex v_h , then, depending on the convexity of v_{j+1} and v_{j+2} , either Lemma B.3 or Lemma B.4 shows that there is a (nonconflicting) feasible negative configuration. If X^- is blocked by an edge e_h , p_{j+1} must be reflex. We distinguish two subcases on the closable chain $\mathcal{Y} = \langle e_j, \dots, e_h \rangle$.

Case 2.1: If $\mathcal{Y}$ does not have a convex inner edge, then we refer to Lemma B.3 to find a feasible negative configuration X' . Again, we may distinguish two subcases, depending on whether X' conflicts with X^+ .

- (a) If X' does not conflict with X^+ , we have a proper configuration pair (see Figure 34(a)).
- (b) If X' conflicts with X^+ , we know that $e_h = e_{i-1}$ holds and that e_{h-1} is the inner edge of X' (see Figure 34(b)). Moreover, we argue as follows that $\alpha(X^+) > 0$ holds in this case. As e_h blocks configuration X^- with inner edge e_j , we can make a pseudotriangle consisting of three convex vertices— p_i, p_j and the intersection p^* between the lines spanned by e_h and e_j —and a sequence of one or more reflex vertices $p_{i+1}, \dots, p_{j-i}$. This pseudotriangle is illustrated in Figure 34(c). The sum of exterior angles of this pseudotriangle must be equal to 2π , i.e., $p^* + \sum_{k=i}^j \alpha(p_k) = 2\pi$. Since p_j and p^* are positive and strictly smaller than π , we derive that $\sum_{k=i}^{j-1} \alpha(p_k) > 0$. As the vertices $p_{i+2}, \dots, p_{j-1}$ are reflex, their exterior angle is negative. Hence, this also implies that $\alpha(p_i) + \alpha(p_{i+1}) > 0$. These are exactly the inner vertices of X^+ and

Fig. 34. Two cases if X^- is not feasible and chain $\mathcal{Y} = \langle e_j, \dots, e_h \rangle$ has no convex inner edge (case 2.1). (a) X' does not conflict with X^+ . (b) X' conflicts with X^+ . (c) This pseudotriangle shows that $\alpha(X^+) > 0$.

thus $\alpha(X^+) > 0$. Thus, according to Lemma B.5, we need to consider this case only when all positive configurations are feasible. In particular, the positive configuration $\langle e_i, e_{i+1}, e_{i+2} \rangle$ is feasible and it does not conflict with X' : in other words, they form a proper configuration pair.

Case 2.2: If $\mathcal{Y}$ has a convex inner edge e_t , then let $X' = \langle e_{t-1}, e_t, e_{t+1} \rangle$ denote the corresponding negative configuration. We may distinguish three subcases, depending on whether X' is feasible and, if it is feasible, whether it conflicts with X^+ .

- (a) If X' is feasible and not conflicting with X^+ , then (X', X^+) constitutes a proper configuration pair.
- (b) If X' is feasible but conflicting with X^+ , we can argue as in case 2.1(b): $\langle e_i, e_{i+1}, e_{i+2} \rangle$ is a feasible positive configuration and it does not conflict with X' . Again, we find a proper configuration pair.
- (c) If X' is not feasible, it must be blocked by some vertex p_b . Assume that the proper chain $\mathcal{Y}' = \langle e_t, e_{t+1}, \dots, e_{b-2}, e_{b-1} \rangle$ does not contain edge e_i . If it does, we may argue analogously for the reversed chain $\langle e_t, e_{t-1}, \dots, e_{b+1}, e_b \rangle$. Depending on the convexity of vertex p_{t+2} , Lemma B.3 or Lemma B.4 shows that there is a feasible negative configuration X'' in $\mathcal{Y}'$. These lemmas imply that at least one of the following three properties hold for X'' :
 - (1) the first inner vertex of X'' along $\mathcal{Y}'$ is reflex;
 - (2) the inner edge of X'' is convex;
 - (3) or the inner edge of X'' is not the before-last edge of $\mathcal{Y}'$.

To derive a contradiction, we assume that X'' conflicts with X^+ and showing that none of the above three properties can hold under this assumption. As X'' is part of $\mathcal{Y}'$, an edge can be shared only if $\mathcal{Y}'$ and X^+ have some edge in common. We constructed $\mathcal{Y}'$ to not contain e_i , the inner edge of X^+ . Hence, we conclude that a conflict can occur only if the blocking vertex p_b is in fact equal to p_i (or $p_b = p_{i+1}$ in the analogous reversed case). As X'' is assumed to conflict with X^+ , X'' is the last configuration in $\mathcal{Y}'$, i.e., $X'' = \langle e_{i-3}, e_{i-2}, e_{i-1} \rangle$. Since p_b is the blocking point of X' , $p_b = p_i$ is a reflex vertex. Therefore, p_{i-1} must be reflex as well: otherwise the shared edge would have a convex and a reflex vertex and thus not cause a conflict. Since X'' is a negative configuration, p_{i-2} is therefore convex. Let us now consider the three possible properties of X'' again.

- (1) The first inner vertex of X'' , p_{i-2} , is convex: property (1) cannot hold.
- (2) The inner edge of X'' ends in p_{i-2} and p_{i-1} which are a convex and reflex vertex respectively: property (2) cannot hold.
- (3) Configuration X'' is the last in $\mathcal{Y}'$. Hence, its inner edge is the before-last edge in the chain: property (3) cannot hold.

None of the three properties for X'' can hold, thus contradicting our assumption that X'' conflicts with X^+ . We conclude that also in this case, we find a proper configuration pair.

In all cases, we find a proper configuration pair, thus proving the lemma. $\square$

A proper configuration pair is a nonconflicting complementary pair of configurations that admit a contraction. Thus, Theorem 3.2 follows directly from Lemma B.6.