Method for Identifying the Polygons with Maps

Peter Rottmann
Institute of Geodesy and Geoinformation, University of Bonn, Germany

Anne Driemel
Hausdorff Center for Mathematics, University of Bonn, Germany

Herman Haverkort
Institute of Computer Science, University of Bonn, Germany

Heiko Röglin
Institute of Computer Science, University of Bonn, Germany

Jan-Henrik Haunert
Institute of Geodesy and Geoinformation, University of Bonn, Germany

ABSTRACT: *We present a new method for the task of detecting groups of polygons in a given geographic data set and computing a representative polygon for each group. This task is relevant in map generalization where the aim is to derive a less detailed map from a given map. Following a classical approach, we define the output polygons by merging the input polygons with a set of triangles that we select from a constrained Delaunay triangulation of the input polygons' exterior. The innovation of our method is to compute the selection of triangles by solving a bicriteria optimization problem. While on the one hand we aim at minimizing the total area of the outputs polygons, we aim on the other hand at minimizing their total perimeter. We combine these two objectives in a weighted sum and study two computational problems that naturally arise. In the first problem, the parameter that balances the two objectives is fixed and the aim is to compute a single optimal solution. In the second problem, the aim is to compute a set containing an optimal solution for every possible value of the parameter. We present efficient algorithms for these problems based on computing a minimum cut in an appropriately defined graph. Moreover, we show how the result set of the second problem can be approximated with few solutions. In an experimental evaluation, we finally show that the method is able to derive settlement areas from building footprints that are similar to reference solutions.*

Keywords: Polygons, Optimization, Maps Study

Received: 19 July 2021, Revised 2 September 2021, Accepted 13 September 2021

DOI: 10.6025/jmpt/2021/12/4/95-109

1. Introduction

Map generalization is the process of deriving a less detailed map from a given map. It comprises multiple sub-tasks such as the selection, simplification, aggregation, and displacement of objects. In this paper we address the task of detecting polygon groups and aggregating each group to a single polygon, which we simply refer to as *aggregation of polygons*. This task is relevant, e.g., to derive settlement areas from mutually disjoint building footprints.

A popular method for the aggregation of polygons is the *adopt merge amalgamation operator* proposed by Jones et al. [19], which is based on a constrained Delaunay triangulation of the space not covered by the input polygons; see Figure 1. The approach is to select a set $T' \subseteq T$ from the set T of triangles of the triangulation to glue together groups of input polygons. More precisely, the connected regions in the union of the triangles in T' and the input polygons constitute the output polygons. We propose a new method that follows this approach. Our focus is to find an optimal selection of triangles, whereas Jones et al. [19] left it largely open how the selection T' of triangles is computed. They rather generally T' recommended to

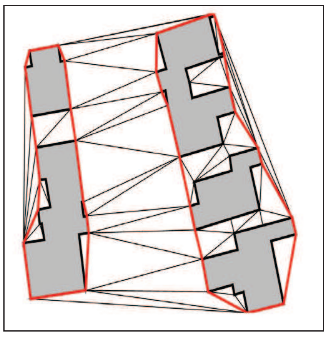


Figure 1. Input polygons (filled gray) aggregated to larger ones (red lines)

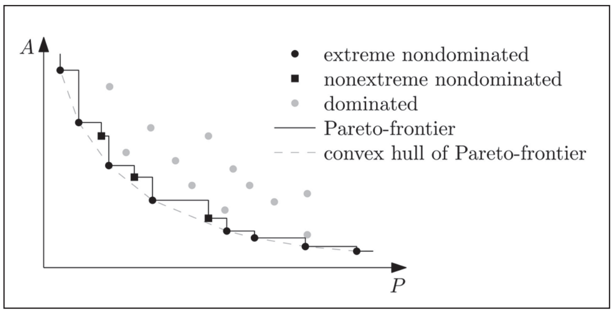


Figure 2. Different types of solutions and Pareto-frontier of a bicriteria optimization problem

use rules with thresholds. With our work we thus aim to overcome the issue raised by Li et al. [21], who noticed that the aggregation of polygons has often been discussed on a general conceptual level and that difficulties arise when engineering a method in detail. In particular, computing the set T of triangles based on rules with thresholds requires finding the right threshold values, which is a cumbersome task.

Since setting the parameters of an algorithm is a challenge of general relevance, we do not only aim to provide a new method for polygon aggregation but also set off to develop a generic approach for the systematic exploration of different parameter values. Generally, map generalization aims at finding a good balance between (i) the preservation of the information given with the input map and (ii) the legibility of the output map [7]. We will give particular consideration to setting a parameter that balances between these two objectives.

In our method for polygon aggregation, each of the two general objectives of map generalization is implemented with one basic quantitative measure. We consider the preservation of information by minimizing the total area of the output polygons, meaning that only little area should be added to the input polygons when merging the selected triangles with them. Legibility is considered by minimizing the total boundary length (or *perimeter*) of the output polygons, which can be considered as an implementation of Tufte's minimum-ink principle [38]. A parameter α, which we call *balance factor*, is used to combine these objectives with a weighted sum. To formalize this, we refer with $A(S)$ to the area and with $P(S)$ to the perimeter of the union of all polygons in a set S, where the union can be a polygon or a multipolygon. For a single polygon p we simply refer with $A(p)$ and $P(p)$ to its area and perimeter, respectively. With this we state the problem we aim to solve as follows.

Problem 1. *Given a subdivision of the plane as a set S of n simple polygons, a set $B \subseteq S$, and a balance factor* $\alpha \in [0, 1]$*, select a subset S' with $B \subseteq S' \subseteq S$ minimizing* $f_\alpha(S') = \alpha \cdot A(S') + (1-\alpha) \cdot P(S')$.

In our application, B is the set of input polygons, e.g., building footprints. The set S contains all polygons of a planar partition, including the polygons in B and the triangles of a triangulation partitioning the space not covered by the input polygons. Note that the restriction to a triangulation is not necessary, i.e., one may use any other partition of the plane instead. The requirement $B \subseteq S' \subseteq S$ means that the input polygons have to be in the selection S'. The balance factor α combines the two objectives with a weighted sum, yielding the overall objective function f_α. To state that a solution to Problem 1 is optimal for a certain balance factor α, we refer to it as an α-*optimal solution*.

Finally, with the following problem we address the challenge of setting the parameter α.

Problem 2. *Given a subdivision of the plane as a set S of n simple polygons and a set $B \subseteq S$, find a set containing for every* $\alpha \in [0, 1]$ *an optimal solution to Problem* 1.

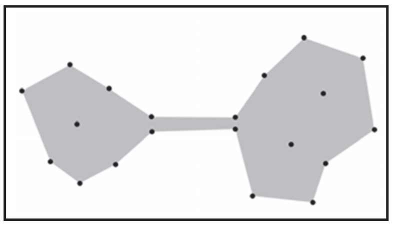


Figure 3. An α-shape that generates a narrow bridge between two point sets

By not requiring a pre-set value for α, our method becomes parameter free. However, it now returns a set of solutions instead of a single one. This can be useful to let an expert choose from a set of alternative solutions. Moreover, the different solutions could be evaluated automatically by comparing them with a reference solution, to infer a suitable value for α.

According to the definition of Problem 2, if for a fixed α there are multiple optimal solutions, only one of them has to be included in the result set. This corresponds to finding what is called the set of all *extreme nondominated points*, which together with the *nonextreme nondominated points* constitute the *Pareto-frontier* [3]; see Figure 2.

We now summarize our contribution and give an outline of the paper.

Our Contribution

1. We show how to solve Problem 1 efficiently by computing a minimum cut in an appropriately defined graph. This approach is inspired from image analysis, where graph cuts are commonly used for image segmentation and related problems.

2. We show that it suffices to include $O(n)$ solutions in the result set of Problem 2 and that these solutions are geometrically nested. The linear size of the result set implies that Problem 2 can be solved with $O(n)$ min-cut computations via a generic recursive algorithm. We also show how the recursive algorithm can be used to compute a small set of solutions approximating the result set of Problem 2.

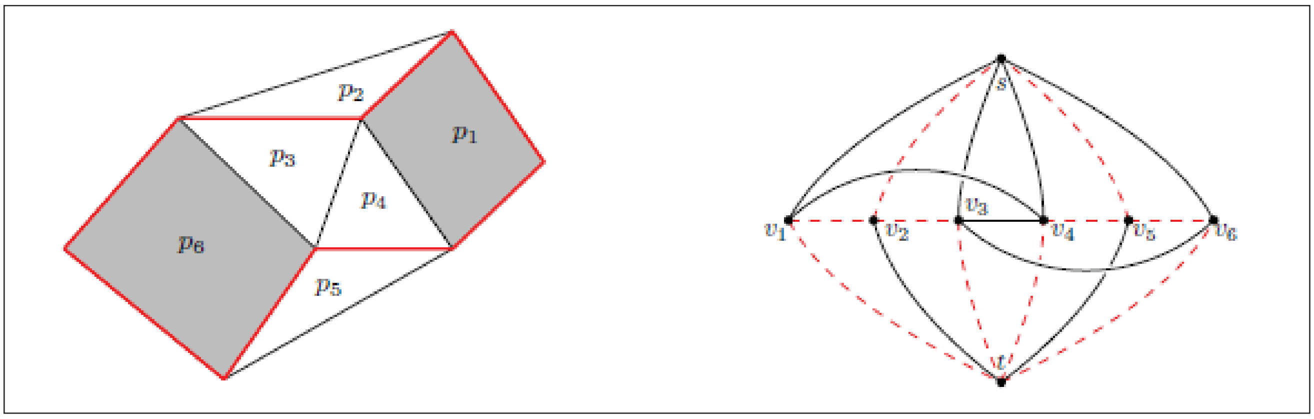
3. We evaluate our method on real data showing its applicability in practice. In the following, we review related work (Sect. 2), present the three above-mentioned contributions (Sects. 3–5), and provide a conclusion and outlook on future work (Sect. 6).

2. Related Work

Automatic map generalization remains a big challenge despite decades of research. The challenge lies on the one hand in the complex interplay between different processes of map generalization – a way to deal with this is to use multi-agent systems for the orchestration of multiple map generalization operators [15, 23]. On the other hand, the challenge lies in the acquisition of cartographic knowledge in a form that can be used by a computer. Machine learning approaches have been proposed to solve this task, with a recent shift towards deep learning [14, 37]. While we focus on a single map generalization operator, the aggregation of polygons, our method for exploring different parameter values can be used for automatic parameter tuning and, thus, considered as a machine-learning contribution. Polygon aggregation is relevant when generalizing categorical coverage maps [16, 18] or choropleth maps [27], where the polygons form a mosaic. These tasks are similar to districting tasks where the aim is to group small areal units to form larger regions such as electoral districts or police districts [10, 20]. In this paper, however, we deal with the aggregation of potentially non-adjacent polygons, e.g., buildings.

The aggregation of polygons is closely related to the aggregation of points. The common goal is to find a single polygon or multiple polygons enclosing the input data. A naïve method is to compute the convex hull of all input points. However, since this may enclose large empty regions, generalizations of the convex hull such as α-shapes [13] have been developed. As an unwanted side effect, α- shapes tend to introduce narrow bridges between two nearby point sets; see Figure 3. Such a bridge can consist of a single edge, in which case it can be easily removed [4], but handling bridges of multiple parallel edges is not straight forward. Similar issues can arise with the concave hull introduced Moreira and Santos [26], which is based on k-nearest neighbours clustering. In contrast, our method would never connect two polygons with a narrow bridge as the one in Figure 3 since *not* selecting the bridge would yield a solution with both a smaller perimeter and less area. Note that the parameter α of our method is different from that of α-shapes. Duckham et al. [12] defined the χ(chi)-hull as a further generalization of the convex hull. First, all points are triangulated using a Delaunay triangulation. Then, all boundary edges that are longer than a threshold are removed. This procedure always returns a single polygon without holes. This may include large empty regions and does not separate groups of points from each other. To address the latter issue, Duckham et al. [12] suggest to identify clusters in a pre-processing step. However, this does not prevent the method from covering large empty regions within a cluster.

With respect to the aggregation of polygons, Jones et al. [19] proposed a method for merging polygons by selecting triangles of a constrained Delaunay triangulation, which they call *adopt merge amalgamation*. They do not specify the criteria for the selection of the triangles but generally recommend to use rules based on thresholds on the triangles' edge lengths. Using such rules the adopt merge amalgamation operator has been implemented and experimentally evaluated by Li and Ai [22]. They showed that the method tends to generate narrow bridges that can consist of a single triangle touching an input polygon with only one of the triangle's vertices. The method of Li et al. [21] overcomes this deficit by selecting sequences of triangles instead of single ones. However, the rules used to govern the selection are set up to aggregate polygons with parallel boundaries separated by long and narrow corridors. With this the method can be used to derive built-up areas from city blocks but not, e.g., settlement areas from footprints of detached houses. Sayidov et al. [33] compute groups of polygons using a triangulation-based method and suggest to compute a representative polygon for each group in a separate processing step. To accomplish this step, automatic typification method can be used [1, 6, 24]. Similarly, Steinger et al. [36] suggest a method that first detects groups of islands and then generates a representative polygon for each group. They define



(a) an instance of Problem 1, where $B = \{p_1, p_6\}$; red (bold) edges delineate a solution with selection $S = \{p_1, p_3, p_4, p_6\}$.

(b) graph G for the instance in (a); red (dashed) edges show an s-t-cut modelling the solution S

Figure 4. Algorithmic solution of Problem 1 via a graph cut

the groups with a subgraph of a minimum spanning tree and generate for each group the convex hull. Damen et al. [9] present an approach for building generalization based on morphological operators. They combine multiple closing and opening operations to simplify but also aggregate the input polygons. To retain the original rectangular geometry that is typical for buildings they implement the closing operator with a Minkowski sum of the input polygon and a square aligned with it.

Our method for polygon aggregation computes a binary partition of the set of polygons of a planar subdivision. For computing the binary partition we adopt a technique from computer vision based on graph cuts [5]. Graph cuts are very commonly used for image segmentation [35, 39] or stereo matching [2]. However, most applications are concerned with raster data. An exception is the work by Sadlacek and Zara. [34] dealing with the generation of polygonal objects from three-dimensional point clouds. With respect to the automatic tuning of a parameter that balances two objectives, the work of Peng and Veksler [30] is most related to ours. While they focus on the development of quality metrics for the evaluation of image segmentation solutions, they simply use a constant step width to sample different values for the parameter of a weighted-sum model. In contrast, we focus on a more systematic exploration of different parameter values.

3. Polygon Aggregation via Minimum Cuts

For solving Problem 1 with graph cuts, we set up an undirected weighted graph $G = (V, E)$ modeling all feasible solutions as well as our minimization goal. This approach is illustrated in Figure 4 and described in detail in the following.

As starting point we use the adjacency graph $G' = (V', E')$ of the planar subdivision given with the set S of polygons. Assuming that the polygons are numbered in an arbitrary order as $p_1, \ldots, p_{n,}$ we refer to the corresponding nodes in V' as $v_1, \ldots, v_n$. The edge set E' contains an edge $\{v_i, v_j\}$ for every two polygons p_i and p_j whose boundaries share at least one line segment. We define the node set of G as $V = V \cup \{s, t\}$, where s is a node called *source* and t a node called *sink*. The edge set E of G contains all edges in E' as well as, for $i = 1, \ldots, n$, the two edges $\{s, v_i\}$ and $\{v_i, t\}$; see Figures 4a and b. An s-t-cut in G is a set of edges whose removal from G causes s and t to be in different connected components. We solve Problem 1 by defining an edge weighting w: $E \rightarrow \mathbb{R}_{\geq 0}$ and computing a *minimum* s-t-cut in G, i.e., an s-t-cut in G of minimum total edge weight.

Formally, for any s-t-cut $C \subseteq E$, we define its weight as $w(C) = \sum_{e \in C} w(e)$ and the graph $G_C = (V, E \setminus C)$. We call the connected component of G_C containing s the *source component* and the connected component of G_C containing t the *sink component* of C. Moreover, we refer to the set of polygons represented by nodes in the source component as the solution *modeled* by C; see the red edges in Figures 4a and b. It remains to ensure that any solution modeled by a *minimum* s-t-cut in G is feasible and optimal with respect to Problem 1. For this we define the edge weighting w as follows:

For every edge $e = \{v_i, v_j\}$, we set $w(e) = (1-\alpha) \cdot l(p_i, p_j)$, where (p_i, p_j) is the length of the common boundary of polygons p_i and p_j.

For every node v_i with $p_i \notin B$, we set $w(\{s, v_i\}) = 0$ and $w(\{v_i, t\}) = \alpha \cdot A(p_i) + (1-\alpha) \cdot l(p_i)$, where $l(p_i)$ is the length of the boundary between p_i and the outer face (0 if p_i and the outer face are not adjacent).

For every node v_i with $p_i \in B$, we set $w(\{s, v_i\}) = \infty$ and $w(\{v_i, t\}) = \alpha \cdot A(p_i) + (1 - \alpha) \cdot l(p_i)$. This avoids that $\{s, v_i\}$ is selected for the cut and thus ensures that v_i is in the source component. (In practice, we use a floating-point number format with a special value representing ∞.)

For computing a minimum s-t-cut and the corresponding optimal solution to Problem 1 we then use a standard graph algorithm.

Theorem 1. The solution modeled by any minimum s-t-cut in G is an optimal solution to Problem 1. This allows Problem 1 to be solved in $O(n^2/\log n)$ *time.*

Proof. We prove that (i) each selection S' with $B \subseteq S' \subseteq S$ is modeled by an s-t-cut in G whose total weight is $\alpha \cdot A(S') + (1-\alpha) \cdot P(S') = f_\alpha(S')$ and (ii) each s-t-cut in G of total weight $W \neq \infty$ models a solution whose objective value measured with f_α is at most W. This together implies that any solution modeled by a minimum s-t-cut in G is an optimal solution to Problem 1.

To show (i), let S' be an arbitrary solution with $B \subseteq S' \subseteq S$ and C the cut defined as follows. For each $p_i \in S'$, we add edge $\{v_i, t\}$ to C, which amounts to weight $\alpha \cdot \sum_{p_i \in S'} A(p_i) + (1-\alpha) \cdot \sum_{p_i \in S'} \ell(p_i)$. Moreover, we add each edge $\{v_i, v_j\} \in E'$ with $p_i \in S'$ and $p_j \notin S'$ to C, which amounts to weight $(1-\alpha) \cdot \sum_{\{v_i,v_j\} \in E''} \ell(p_i, p_j)$ where $E'' = \{\{v_i, v_j\} \in E' \mid p_i \in S' \wedge p_j \notin S'\}$. The set C is an s-t-cut in G modeling S' because the nodes for polygons in S' plus node s constitute the source component of C. The weight of C is

$$\begin{aligned} w(C) &= \alpha \cdot \textstyle\sum_{p_i \in S'} A(p_i) + (1-\alpha) \cdot \sum_{p_i \in S'} \ell(p_i) + (1-\alpha) \cdot \sum_{\{v_i,v_j\} \in E''} \ell(p_i, p_j) \\ &= \alpha \cdot A(S') + (1-\alpha) \cdot \left(\textstyle\sum_{p_i \in S'} \ell(p_i) + \sum_{\{v_i,v_j\} \in E''} \ell(p_i, p_j) \right) \\ &= \alpha \cdot A(S') + (1-\alpha) \cdot P(S') = f_\alpha(S'). \end{aligned}$$

To show (ii), we consider an arbitrary s-t-cut C' in G of total weight $w(C') = W \neq \infty$. Let S' be the solution modeled by C'. Because of $w(C') \neq \infty$, S' satisfies $B \subseteq S' \subseteq S$. Now, let C be the cut for S' as defined in the proof of (i). As argued before, C models S and its weight equals the objective value of S', i.e., $w(C) = \alpha \cdot A(S') + (1 - \alpha) \cdot P(S')$. Moreover, the weight of C is at most the weight W of C', since every edge in C is included in C as well, which can be seen as follows. Assume that there exists an edge $e = \{u, v\}$ with $e \in C$ and $e \notin C'$. Because of $e \in C$ and the way we constructed C, one of the nodes u and v lies in the source component of C and the other one in the sink component of C. Because of $e \notin C'$, u and v lie in the same connected component of $G_{C'}$. This contradicts the assumption that C and C model the same solution. Therefore, the assumption $e \in C$ and $e \notin C'$ must have been false. This allows us to conclude that $w(C') \geq w(C) = \alpha \cdot A(S') + (1 - \alpha) \cdot P(S')$.

The currently fastest algorithm for computing a minimum s-t-cut in a graph with $O(n)$ edges runs in $O(n^2/\log n)$ time [29]. This result applies to our case since G' is planar (which implies that it has $O(n)$ edges) and since G has only two more edges for each node of G'. The running time for computing the cut dominates the running times for computing G from the input and generating the output selection from the cut (e.g., via a depth-first search in the graph without the cut edges, starting from the source). Hence, the overall running time for solving Problem 1 is $O(n^2/\log n)$.

4. Computing Solutions for Multiple Parameter Values

In this section we deal with Problem 2, which asks for a set containing an α-optimal solution for every $\alpha \in [0, 1]$. First, in Sect. 4.1, we show that a set with the required property and $O(n)$ solutions exists. Then, in Sect. 4.2, we present an algorithm or computing such a set or, more generally, a set containing for every α a solution that is at most a factor $(1 + \varepsilon)$ worse than an α-optimal solution while avoiding redundancy. The latter is relevant if the aim is to approximate the output set of Problem 2 with few solutions.

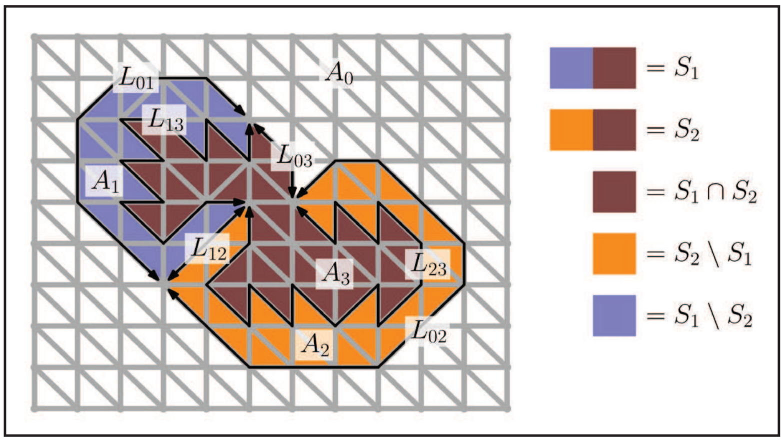


Figure 5. Schematic visualization of two α-optimal solutions S_1 and S_2. The labels A_i refer to the areas and the labels L_{ij} refer to the line lengths used in the proof of Lemma 2

4.1. Linear Number of Solutions

Our argument about a sufficient size for the set requested by Problem 2 is based on the following structural lemma.

Lemma 2. *For an instance of Problem* 1*, let* S_1 *be an* α_1*-optimal solution and* S_2 *be an* α_2*-optimal solution, where* $\alpha_1 > \alpha_2$*. Then* $S_1 \subseteq S_2$.

Proof. We distinguish four classes of polygons: those that are neither in S_1 nor in S_2; those that are only in S_1; those that are only in S_2; and those that are in $S_1 \cap S_2$. We denote these classes by $T_0 = S \setminus (S_1 \cup S_2)$; $T_1 = S_1 \setminus S_2$; $T_2 = S_2 \setminus S_1$; and $T_3 = S_1 \cap S_2$. By A_i we denote the area of T_i, and by L_{ij} we denote the total length of the edges that are shared by T_i and T_j; see Figure 5. Let α'_i be $1 - \alpha_i$.

The α_1-optimal solution $S_1 = T_1 \cup T_3$ is at least as good as $S_1 \cap S_2 = T_3$ for $\alpha = \alpha_1$:

$$\alpha_1(A_1 + A_3) + \alpha'_1(L_{01} + L_{03} + L_{12} + L_{23}) \le \alpha_1 A_3 + \alpha'_1(L_{03} + L_{13} + L_{23}) \Leftrightarrow \alpha_1 A_1 + \alpha'_1 L_{12} \le \alpha'_1(L_{13} - L_{01}). \quad (1)$$

The α_2-optimal solution $S_2 = T_2 \cup T_3$ is not worse than $S_1 \cup S_2 = T_1 \cup T_2 \cup T_3$ for $\alpha = \alpha_2$:

$$\alpha_2(A_2 + A_3) + \alpha'_2(L_{02} + L_{03} + L_{12} + L_{13}) \le \alpha_2(A_1 + A_2 + A_3) + \alpha'_2(L_{01} + L_{02} + L_{03}) \Leftrightarrow \alpha'_2(L_{13} - L_{01}) \le \alpha_2 A_1 \le \alpha'_2 L_{12}. \quad (2)$$

Now suppose $S_1 \not\subseteq S_2$, that is, $A_1 > 0$. With $\alpha_1 > \alpha_2 \ge 0$, Equation (1) then implies $L_{13} - L_{01} > 0$. Since $\alpha_2 < \alpha_1$, we also have $\alpha'_1 < \alpha'_2$. Combining this with (1) and (2) gives:

$$\alpha_1 A_1 \le \alpha_1 A_1 + \alpha'_1 L_{12} \le \alpha'_1(L_{13} \le L_{01}) < \alpha'_2(L_{13} \le L_{01}) \le \alpha_2 A_1 \le \alpha'_2 L_{12} \le \alpha_2 A_1.$$

Thus, $\alpha_1 A_1 < \alpha_2 A_1$. However, this contradicts the assumption $\alpha_2 < \alpha_1$. Therefore, the assumption $S_1 \not\subseteq S_2$ must have been false, and we conclude $S_1 \subseteq S_2$.

Suppose that we continuously decrease α from 1 to 0 while maintaining an α-optimal solution in a lazy fashion, meaning that we replace the current solution only if it ceases being α-optimal. At the beginning of this process we have the solution containing only the polygons in B, since for $\alpha = 1$ area is all that matters. Due to Lemma 2, whenever we have to replace the current solution, we can define the new solution by including all the polygons selected in the current solution plus at least one additional polygon from the set S. Since there can be at most $n - 1$ such events, we can conclude the following lemma.

***Lemma 3*:** There exists a set of cardinality $O(n)$ that contains an α-optimal solution for every $\alpha \in [0, 1]$.

4.2. The Search Algorithm

According to Lemma 3, a linear number of solutions suffices to have an α-optimal solution for every $\alpha \in [0, 1]$ as demanded by Problem 2. Our goal is now to find such a linear-size set efficiently within the set of all possible solutions, which has exponential size. We do this with a classical method for multi-objective optimization [8], which in the literature is sometimes referred to as *dichotomic scheme* [32] or *chord algorithm* [11]. In the following we describe the algorithm and how it is applied in our setting to approximate the result set of Problem 2. For this we assume a user-specified error tolerance $\varepsilon \ge 0$ as input. We ask for an output set containing for every $\alpha \in [0, 1]$ a solution S' satisfying $f_\alpha(S') \le (1 + \varepsilon) \cdot opt$, where opt is the value of f_α for an α-optimal solution. Approximating for $\varepsilon = 0$ will solve Problem 2.

Before describing the approximation algorithm we choose a geometric representation for solutions that simplifies the discussion. Consider visualizing a solution S' in a diagram with α on the horizontal axis and the solution value $\alpha A(S') + (1 - \alpha) P(S')$ on the vertical axis (see Figure 6 for an illustration). Thus, the function graph of S' is a line through the points $(0, P(S'))$ and $(1, A(S'))$. Now suppose we put the function graphs of all possible solutions in a single diagram. A solution S is α-optimal if the line for S' is the first line that is hit when we shoot a vertical ray from $(\alpha, 0)$ upwards. Thus, to compute an α-optimal solution for every α, we need to compute the lower envelope (the piecewise minimum) of the arrangement of all possible solutions' lines, restricted to the α-range $[0, 1]$; see Figure 6a. Moreover, approximating the set of all α-optimal solutions roughly means to find a set of solutions whose lower envelope is not too far above the lower envelope of all solutions. To solve the approximation problem, we use the recursive procedure Approx presented in Algorithm 1, which only considers a range $[\alpha_L, \alpha_U]$ of values for α. To consider all values for α in the range [0, 1], we proceed as follows:

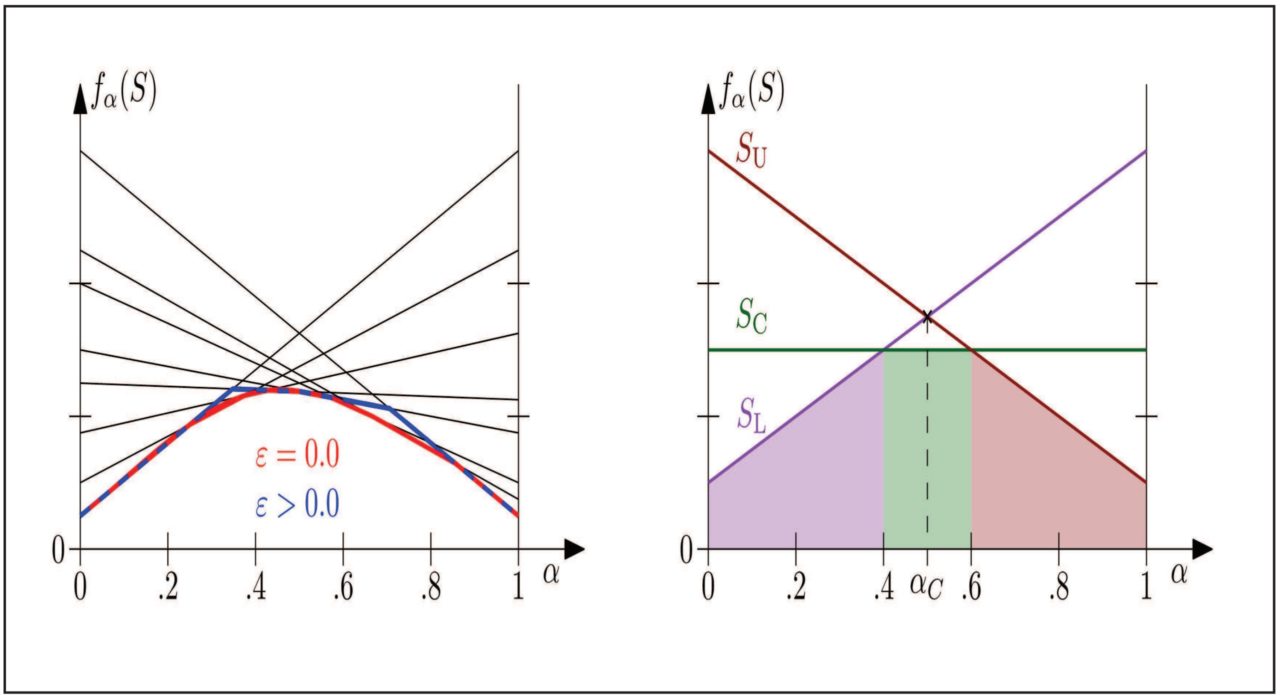
1. Compute α-optimal solutions S_L for $\alpha = 0$ and S_U for $\alpha = 1$ using the min-cut algorithm.

2. Invoke Approx with $\alpha_L = 0$ and $\alpha_U = 1$: *Approx*$(S, B, 0, 1, S_L, S_U, \varepsilon)$

The solutions S_L and S_U are passed over to the procedure only to allow it to access their associated areas $A(S_L)$ and $A(S_U)$ as well as perimeters $P(S_L)$ and $P(S_U)$. To keep the presentation simple, we refer to these as A_L and A_U as well as P_L and P_U, respectively.

The recursive procedure operates as follows. If $A_L = A_U$ and $P_L = P_U$ (this may happen in special cases), then the recursive procedure returns without output (ending in a leaf of the recursion tree). Otherwise, we compute the α-value $\alpha_c = (P_U - P_L)/(A_L - A_U + P_U - P_L)$ for which S_L and S_U are equally good. (In effect, we compute the α-coordinate of the intersection point C of the lines representing S_L and S_U). Then, we call the min-cut algorithm to obtain an α_C-optimal solution S_C. We compare the objective function value for this new solution with the objective function for S_L (comparing to S_U would yield the same result). If the new solution is substantially better (depending on the approximation factor ε), then the corresponding line passes far below the intersection point C in the diagram between α_L and α_U (refer to Figure 6b for an illustration). We then call the procedure recursively twice: once with range (α_L, α_C) and the solutions S_L and S_C, and once with range (α_C, α_U) and the solutions S_C and S_U. In between these two calls, we output the new solution S_C. The described process is outlined by Algorithm 1.

For computing a complete set of solutions $S_1, \ldots, S_k$ representing the k linear pieces that form the lower envelope we call the algorithm with $\varepsilon = 0$. Correctness of the procedure can be shown by induction on the recursion tree. As for the running time, observe that, by Lemma 3, the number of pieces of the lower envelope k is bounded by $O(n)$. Analysing the recursive algorithm



(a) lower envelope (red) with approximation (blue)   (b) recursive step of the algorithm

Figure 6. Geometric representation of solutions as used in the discussion of the search algorithm

Algorithm 1 Approximation of set of extreme nondominated points.

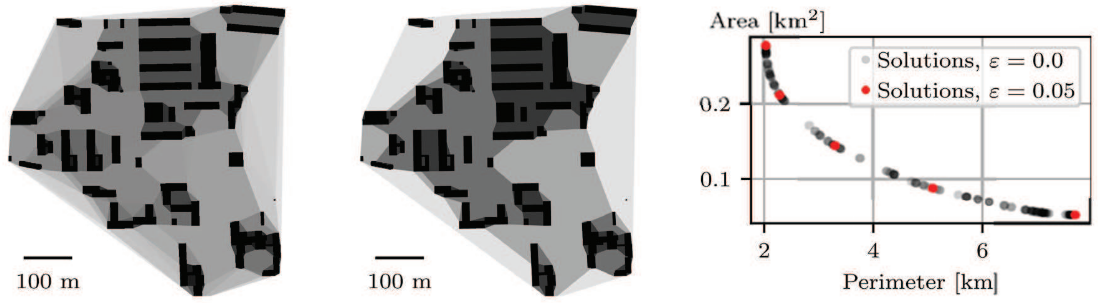
```
Input: Polygon sets $S, B$, lower $\alpha$ ($\alpha_L$), upper $\alpha$ ($\alpha_U$), $\alpha$-optimal solutions $S_L, S_U$ for $\alpha_L, \alpha_U$, maximal error $\varepsilon$
Output: Report $\alpha$-optimal solutions with $\alpha_L < \alpha < \alpha_U$
1: procedure Approx($S, B, \alpha_L, \alpha_U, S_L, S_U, \varepsilon$)
2: if $A(S_L) = A(S_U)$ and $P(S_L) = P(S_U)$ then return
3: $\alpha_C \leftarrow$ crossing($S_L$, $S_U$)                Compute $\alpha_C$ with $f_{\alpha C}(S_L) = f_{\alpha C}(S_U)$
4: $S_C \leftarrow$ mincut($S, B$, $\alpha_C$)                   Solve Problem 1 via graph cut
5: if $f_{\alpha C}(S_L) > (1 + \varepsilon) \cdot f_{\alpha C}(S_C)$ then     Maximal error exceeded
6: Approx($S, B, \alpha_L, \alpha_C, S_L, S_C, \varepsilon$)
7: Report $S_C$
8: Approx($S,B, \alpha_C, \alpha_U, S_C, S_U, \varepsilon$)
```

we see that overall at most $O(n)$ invocations of the min-cut algorithm are performed in the worst case. This also holds for the approximation algorithm (using $\varepsilon > 0$).

5. Experimental Evaluation

To evaluate our algorithms, we implemented them in Java and conducted tests with building footprints from OpenStreetMap [28]. We transformed the data to UTM coordinates to accurately calculate areas and perimeters of polygons with a metric of unit of measurement.

To provide an overview, we first discuss a test with a smaller instance of 44 buildings; see Figure 7. The triangulation for this instance contains 546 triangles. With this test we rather generally tried to get an idea of the kind of polygon groups and aggregated polygons that our algorithms produce. Using the recursive algorithm to compute an α-optimal solution for every α (i.e., $\varepsilon = 0.0$) we obtained 230 different solutions; see Figure 7a. Some of these solutions differ by only single triangles and are hardly distinguishable in the visualization. When using the algorithm to approximate the result set with $\varepsilon = 0.05$, we obtained only 5 different solutions that are substantially different from each other; see Figure 7b. One can clearly perceive the geometrically nested structure of the different solutions stated in Lemma 2. Plotting the solutions by their associated perimeters and areas yields a diagram with the extreme nondominated points; see Figure 7c. We observe that these points occur at irregular



(a) 230 solutions for $\varepsilon = 0.0$ (b) 5 solutions for $\varepsilon = 0.05$ (c) areas and perimeters for solutions

Figure 7. Solutions obtained with Algorithm 1 for the same instance without (a) and with aapproximation (b) and their values for the two objectives (c). Darker polygons correspond to solutions for higher α values, which consist of multiple groups

distances, meaning that clusters but also larger gaps exist. Nevertheless, our approximation approach yields a small and representative set of solutions. Due to this result, we decided to focus on approximated result sets in the further experiments, which we present in the following.

We conducted further experiments with the aim to aggregate building polygons to settlement areas as defined in the digital landscape model DLM250 of the German "Authoritative Topographic-Cartographic Information System" (ATKIS). This corresponds to a map scale 1 : 250 000. We evaluate our approach on two instances of different sizes from the building data set, each of which roughly corresponds to one settlement polygon in the DLM250 but also includes buildings outside of the settlement; see Figure 8. The larger instance consists of 16881 buildings and the smaller one of 853. They correspond to the town Euskirchen and the village Ahrem, respectively. Figure 9 depicts for Ahrem the influence of ε on the number of output solutions. Allowing a maximum error of $\varepsilon = 0.01 = 1\%$ already decreases the number of solutions by several orders of magnitude in comparison to an exact solution ($\varepsilon = 0.0$). For visualizing the aggregation results of the two instances we set $\varepsilon = 0.1$, resulting in six solutions for Euskirchen (see Figures. 8a and 10a) and five for Ahrem (see Figures. 8b and 10b).

To evaluate our aggregation results, we computed a rather accurate approximation for Euskirchen ($\varepsilon = 10^{-6}$) and an exact solution to Problem 2 for Ahrem. We then compared every solution in the result set with the corresponding settlement polygon in the DLM250.

For comparing a solution S_1 with the reference S_2 we used the following four different metrics:

The *Jaccard index*, which is also known as *Intersection over Union* (*IoU*), for polygons:

$$IoU = \frac{A(S_1 \cap S_2)}{A(S_1 \cup S_2)}$$

The *Area Similarity* and *Perimeter Similarity*, as suggested by Podolskaya et al. [31] for quality assessment of polygon generalization:

$$V_{\mathrm{A}} = 1 - \frac{|A(S_1) - A(S_2)|}{\max\{A(S_1), A(S_2)\}} \qquad V_{\mathrm{P}} = 1 - \frac{|P(S_1) - P(S_2)|}{\max\{(P(S_1), P(S_2)\}}$$

The *discrete Hausdorff distance* d_H of the polygons' boundaries, whose vertex sets we denote as $V(S_1)$ and $V(S_2)$. It adds to the other three measure that it indicates the maximum difference of the solutions with respect to the Euclidean distance *d*.

Dataset	α	***IoU*** $\uparrow$	$V_A \uparrow$	$V_P \uparrow$	d_H **[m]** $\downarrow$
Euskirchen	0.0048	0.8951	**0.9997**	0.8033	298.476
	0.0064	**0.8999**	0.9809	0.8437	273.568
	0.0073	0.8943	0.9650	0.8495	**264.562**
	0.0109	0.8301	0.8795	**0.9989**	926.226
Ahrem	0.0021	0.5882	0.6973	**0.9998**	336.437
	0.0041	**0.7896**	**0.9685**	0.9106	**114.570**

Table 1. Statistics for every solution that is the best for at least one evaluation metric. For the Hausdorff distance lower is better (↓), else higher is better (↑)

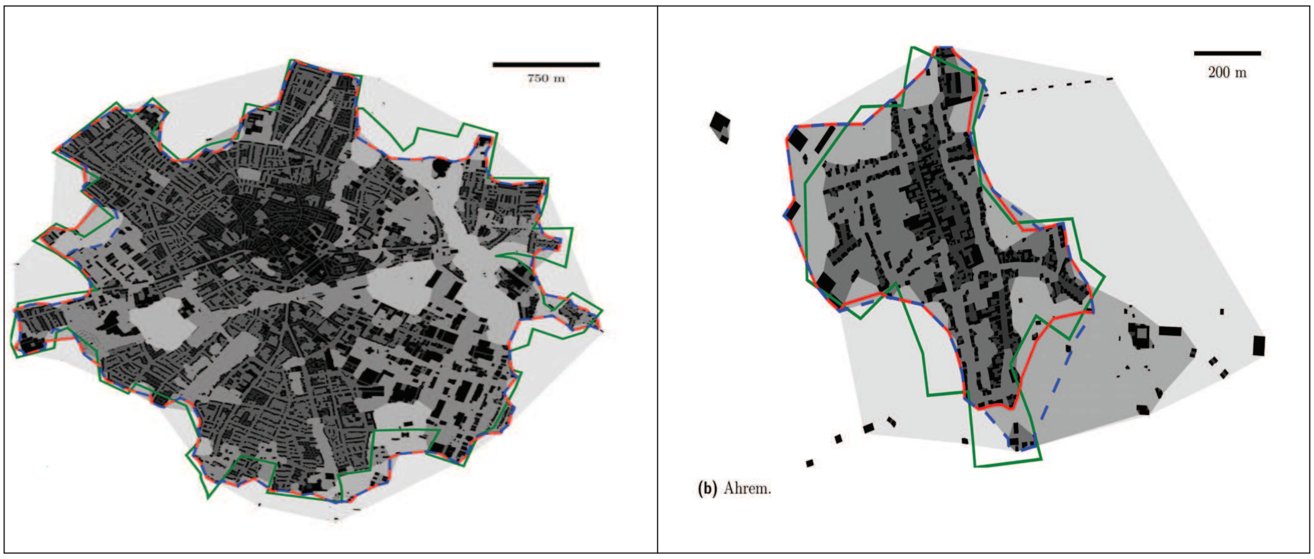
$$d_{\mathrm{H}} = \max_{v_1 \in V(S_1)} \{ \min_{v_2 \in V(S_2)} \{ d(v_1, v_2) \} \}$$

The similarity measures *IoU*, V_A, and V_B are in the range of [0, 1], where 0 represents minimum similarity and 1 represents maximum similarity. For cases where one polygon is contained by a second polygon, V_A equals *IoU*.

An important selection criterion of the DLM250 is that all settlement areas must be larger than 40 hectares. Therefore, we removed all polygons smaller than the threshold from a solution before computing the four metrics. The resulting metrics for Euskirchen are shown in Fig. 11. For all $\alpha > 0.105$ the solution contained no contiguous polygon reaching the 40 hectares threshold, meaning that no similarity can be found. The graph reveals a correlation of *IoU* and Area Similarity V_A; in fact they are the same for $\alpha > 0.05$. The Perimeter Similarity V_P, on the other hand, turned out to be quite meaningless as its graph shows a rather erratic behaviour. This is because solutions for large α often contain polygons smaller than the threshold, leading to a short perimeter after their removal.

For each metric we identified the value for α that yields the solution of maximum similarity or minimum Hausdorff distance; see Table 1. For the cases where *IoU*, V_A, and Hausdorff distance have their respective best values, the other metrics are similar to their best values. At the maximum of V_P, the other metrics are significantly away from their best values. The solutions of minimum Hausdorff distance and maximum *IoU* are also depicted in Figure 8 as blue and red lines, respectively. We observe that the shapes are indeed similar, but that the boundary of the reference solutions contain more angles close to 90^o. Using a schematization algorithm in a post-processing step might increase the similarity in this respect.

In our Java implementation, we used the Push Relabel algorithm implemented in the library JGraphT [25] to compute the graph cuts. This algorithm can be implemented to run in the worst case in $O(nmlog(n^2/m))$ time on a graph with n nodes and m edges [17], but the JGraphT documentation reports a worst-case running time of $O(n^3)$. This implies that our implementation of the algorithm for Problem 1 runs in $O(n^3)$ time, where n is the number of polygons in the set S. This is substantially higher than the sub-quadratic running time that is achievable according to Theorem 1, but sufficiently low to solve problem instances as the ones we discussed above. Our implementation of the recursive algorithm for Problem 2 uses parallelization



(a) Euskirchen (b) Ahrem

Figure 8. Result sets of two evaluation data sets with ε = 0.1. The green outline corresponds to the ground truth polygon. The red and blue outlines, which are mostly the same, represent the best solutions in terms of *IoU* and d_H

for the two recursive calls, which turned out to improve the running time substantially. In particular, the running time improves from 79 seconds to 12 seconds for $\varepsilon = 10^{-6}$ (528 solutions) on the instance of Ahrem by using parallelization.

6. Conclusion and Outlook

We have presented efficient algorithms for polygon aggregation optimizing a balance between a small total area and a short total perimeter of the output polygons. We combined the two criteria in a weighted sum, which we parameterized with a single parameter $\alpha \in [0, 1]$. The first problem we studied asked for a single optimal solution for a fixed α. It turned out that this problem can be solved by computing a minimum cut in a graph. A second problem asked for an output set containing an optimal solution for every possible value for α. We showed that a linear-size set with the requested property can be efficiently computed with a recursive algorithm that uses the graph-cut algorithm as a subroutine. Moreover, we showed how to approximate such a set using the same recursive algorithm.

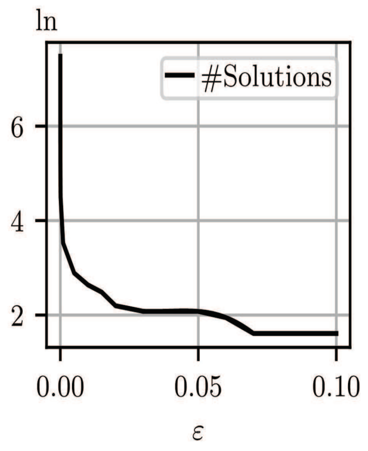


Figure 9. Number of solutions by α for Ahrem

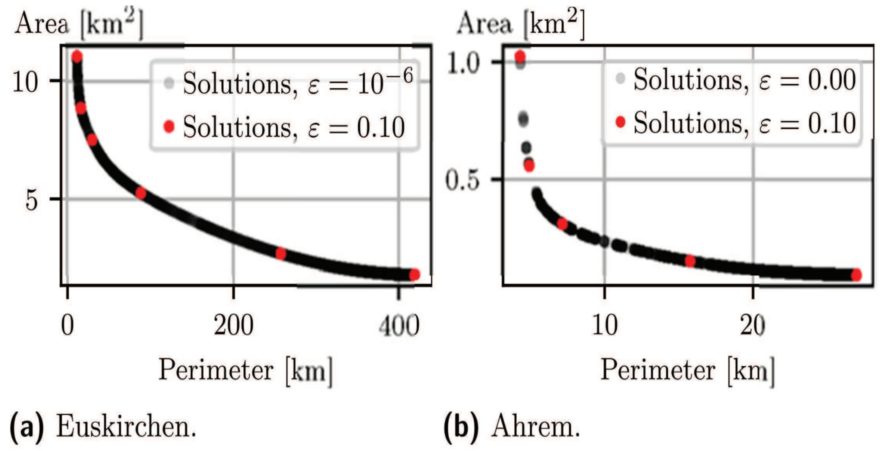


Figure 10. All α-optimal solutions and the approximated solutions which are displayed in Figure 8

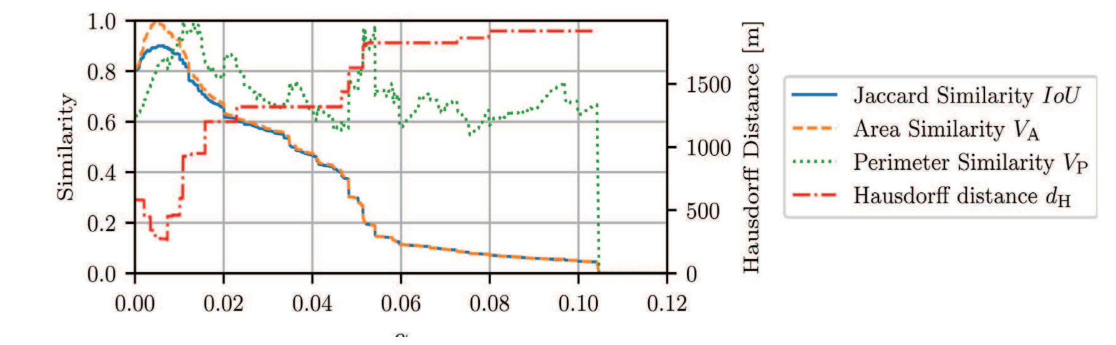


Figure 11. Evaluation metrics for Euskirchen shown in Figure 8a. For $\alpha > 0.105$ all polygon of the result set are below the specified minimum area

Our experiments showed that the presented algorithms are fast enough to process realistic problem instances, although we did not use the fastest known (i.e., sub-quadratic) graph-cut algorithm in our implementation. We consider it astonishing how few solutions were needed to approximate a set containing an optimal solution for every α: For our largest instance with

16881 building footprints, a set of six solutions sufficed to include for every α a solution that is at most 10% worse than optimal. Since the number of graph cuts computed by the recursive algorithm is in the order of the size of its output set, the approximation for the above-mentioned instance was achieved relatively fast, in 20.3 seconds. Finally, our experiments support the claim that our algorithm can aggregate building footprints to polygons that are quite similar to settlement areas as given in an official topographic database of scale 1 : 250 000. However, experiments on a larger data basis are needed to substantiate our finding. It would also be interesting to investigate further whether different instances can be solved with the same α to obtain a solution similar to a reference solution.

As an idea for future research one could replace the triangulation used in our method with other partitions of the plane. For example, one could try to preserve regularities of the input polygons by subdividing the plane with linear extensions of the polygons' edges. Moreover, it would be interesting to consider relaxed or more constrained versions of the aggregation problem. For example, one could relax the requirement to include every input polygon in the output and/or introduce a hard size constraint for the output polygons. Most importantly, however, we see our work as a step towards multi-criteria optimization in cartography. As next steps one could consider more than two criteria or look at Pareto-optimal solutions rather than just at the extreme nondominated (i.e., α-optimal) solutions.

References

[1] Anders, Karl-Heinrich., Sester, Monika. (2000). Parameter-free cluster detection in spatial databases and its application to typification. *In*: Proceedings 19th ISPRS Congress, volume 33 of International Archives of Photogrammetry and Remote Sensing, pages 75–83.

[2] Bleyer, Michael., Gelautz, Margrit. (2007). Graph-cut-based stereo matching using image segmentation with symmetrical treatment of occlusions. Signal Processing: *Image Communication*, 22 (2) 127–143. doi:10.1016/j.image.2006.11.012.

[3] Bökler, Fritz., Mutzel, Petra. (2015). Output-sensitive algorithms for enumerating the extreme nondominated points of multiobjective combinatorial optimization problems. *In*: Proceedings 23rd Annual European Symposium on Algorithms (ESA '15), pages 288–299. doi:10.1007/ 978-3-662-48350-3_25.

[4] Bonerath, Annika., Niedermann, Benjamin., Haunert, Jan-Henrik. (2019). Retrieving a-shapes and schematic polygonal approximations for sets of points within queried temporal ranges. In: Proceedings 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS '19), pages 249–258, 2019. doi:10.1145/3347146.3359087.

[5] Boykov, Yuri., Veksler, Olga. (2006). Graph cuts in vision and graphics: Theories and applications. In Handbook of Mathematical Models in Computer Vision, chapter 5, pages 79–96. Springer, Boston, MA, USA. doi:10.1007/0-387-28831-7_5.

[6] Burghardt, Dirk., Cecconi, Alessandro. (2007). Mesh simplification for building typification. *International Journal of Geographical Information Science*, 21(3), 283–298. doi: 10.1080/13658810600912323.

[7] Burghardt, Dirk., Schmid, Stefan., Stoter, Jantien. (2007). Investigations on cartographic constraint formalisation. *In*: Proceedings 11th ICA Workshop on Generalisation and Multiple Representation, URL: https://kartographie.geo.tu-dresden.de/downloads/ica-gen/ workshop2007/Burghardt-ICAWorkshop.pdf.

[8] Cohon, Jared L. (1978). Multiobjective Programming and Planning. Academic Press, New York, NY, USA, 1978.

[9] Damen, Jonathan., Kreveld, Marc van., Spaan, Bert. (2008). High quality building generalization by extending the morphological operators. *In*: Proceedings 11th ICA Workshop on Generalisation and Multiple Representation, pages 1–12, 2008. URL: https://kartographie.geo.tu-dresden. de/downloads/ica-gen/workshop2008/04_Damen_et_al.pdf.

[10] Amico, Steven J. D'., Wang, Shoou-Jiun., Batta, Rajan., Rump, Christopher M. (2002). A simulated annealing approach to police district design. *Computers & Operations Research*, 29 (6) 667–684. doi:10.1016/S0305-0548(01)00056-9.

[11] Daskalakis, Constantinos., Diakonikolas, Ilias., Yannakakis, Mihalis. (2016). How good is the chord algorithm? SIAM Journal on Computing, 45 (3) 811–858. doi:10.1137/13093875X.

[12] Duckham, Matt., Kulik, Lars., Worboys, Mike., Galton, Antony. (2008). Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41 (10) 3224–3236. doi:10.1016/

j.patcog.2008.03.023.

[13] Edelsbrunner, Herbert., Kirkpatrick, David., Seidel, Raimund. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29 (4) 551–559. doi:10.1109/TIT.1983.1056714.

[14] Feng, Yu., Thiemann, Frank., Sester, Monika. (2019). Learning cartographic building generalization with deep convolutional neural networks. *ISPRS International Journal of Geo-Information*, 8 (6). doi:10.3390/ijgi8060258.

[15] Galanda, Martin. (2003). Automated polygon generalization in a multi agent system. PhD thesis, University of Zurich, Zürich. doi:10.5167/uzh-163108.

[16] Gedicke, Sven., Oehrlein, Johannes., Haunert, Jan-Henrik. (2021). Aggregating land-use polygons considering line features as separating map elements. *Cartography and Geographic Information Science*, 48 (2) 124–139. doi:10.1080/15230406.2020.1851613.

[17] Goldberg, Andrew V., Tarjan, Robert E. (1988). A new approach to the maximum-flow problem. Journal of the ACM, 35 (4) 921–940. doi:10.1145/48014.61051.

[18] Haunert, Jan-Henrik., Wolff, Alexander. (2010). Area aggregation in map generalisation by mixedinteger programming. *International Journal of Geographical Information Science*, 24 (12) 1871– 1897. doi:10.1080/13658810903401008.

[19] Jones, Christopher B., Bundy, Geraint Ll., Ware, Mark J. (1995). Map generalization with a triangulated data structure. *Cartography and Geographic Information Systems*, 22 (4) 317–331. doi:10.1559/152304095782540221.

[20] Kamyoung Kim, Denis J. Dean, Hyun Kim, and Yongwan Chun. Spatial optimization for regionalization problems with spatial interaction: a heuristic approach. International *Journal of Geographical Information Science*, 30 (3) 451–473. doi:10.1080/13658816. 2015.1031671.

[21] Chengming Li, Yong Yin, Xiaoli Liu, and Pengda Wu. An automated processing method for agglomeration areas. ISPRS *International Journal of Geo-Information*, 7 (6) 204. doi:10.3390/ijgi7060204.

[22] Jingzhong, Li., Tinghua, Ai. (2010). A triangulated spatial model for detection of spatial characteristics of GIS data. In Proceedings 2010 IEEE International Conference on Progress in Informatics and Computing (PIC '10), volume 1, pages 155–159. doi:10.1109/PIC.2010.5687417.

[23] Maudet, Adrien., Touya, Guillaume., Duchêne, Cécile., Picault, Sébastien. (2014). Multi-agent multilevel cartographic generalisation in CartAGen. *In*: Proceedings 12th International Conference on Advances in Practical Applications of Heterogeneous Multi-Agent Systems (PAAMS '14), pages 355–358. doi:10.1007/978-3-319-07551-8_37.

[24] McMaster, Robert B., Stuart Shea, K. (1992). Generalization in digital cartography. Association of American Geographers, Washington, DC, USA.

[25] Michail, Dimitrios., Kinable, Joris., Naveh, Barak., Sichi, John V. (2020). JGraphT—a Java library for graph data structures and algorithms. *ACM Transactions on Mathematical Software*, 46 (2). doi:10.1145/3381449.

[26] Moreira, Adriano., Santos, Maribel Y. (2007). Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. *In*: Proceedings 2nd International Conference on Computer Graphics Theory and Applications (GRAPP '07), pages 61–68, 2007. URL: http://hdl.handle.net/1822/6429.

[27] Oehrlein, Johannes., Haunert, Jan-Henrik. (2017). A cutting-plane method for contiguity-constrained spatial aggregation. *Journal of Spatial Information Science*, 15 (1) 89–120. doi:10.5311/ JOSIS.2017.15.379.

[28] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2020.

[29] Orlin, James B. (2013). Max flows in $O(nm)$ time, or better. *In: Proceedings 45th Annual ACM Symposium on Theory of Computing (STOC '13)*, page 765–774. doi:10.1145/2488608.2488705.

[30] Peng, Bo., Veksler, Olga. (2008). Parameter selection for graph cut based image segmentation. In: Proceedingd British Machine Vision Conference (BMVC '08), pages 16.1–16.10. doi: 10.5244/C.22.16.

[31] Podolskaya, Ekaterina S., Anders, Karl-Heinrich., Haunert, Jan-Henrik., Sester, Monika. (2007). Quality assessment for polygon generalization. In Quality Aspects in Spatial Data Mining, chapter 16, pages 211–220. CRC Press, Boca Raton, FL, USA. doi:10.1201/9781420069273.ch16.

[32] Przybylski, Anthony., Gandibleux, Xavier., Ehrgott, Matthias. (2010). A recursive algorithm for finding all nondominated

extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22 (3) 371–386. doi:10.1287/ijoc. 1090.0342.

[33] Sayidov, Azimjon., Weibel, Robert., Leyk, Stefan. (2020). Recognition of group patterns in geological maps by building similarity networks. *Geocarto International*, pages, 1–20. doi:10.1080/ 10106049.2020.1730449.

[34] Sedlacek, David., Zara, Jiri. (2009). Graph cut based point-cloud segmentation for polygonal reconstruction. In: Proceedingd 5th *International Symposium on Advances in Visual Computing* (ISVC '09), pages 218–227. doi:10.1007/978-3-642-10520-3_20.

[35] Shi, Jianbo., Malik, Jitendra. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8) 888–905. doi:10.1109/34.868688.

[36] Steiniger, Stefan., Burghardt, Dirk., Weibel, Robert. (2006). Recognition of island structures for map generalization. In Proceedings 14th Annual ACM International Symposium on Advances in Geographic Information Systems (ACM GIS '06), pages 67–74. doi:10.1145/1183471.1183484.

[37] Touya, Guillaume., Zhang, Xiang., Lokhat, Imran. (2019). Is deep learning the new agent for map generalization? *International Journal of Cartography*, 5 (2–3) 142–157. doi:10.1080/23729333.2019.1613071.

[38] Tufte, Edward R. (1992). The visual display of quantitative information. Graphics Press, Cheshire, CT, USA. doi:10.1119/ 1.14057.

[39] Wu, Zhenyu., Leahy, Richard. (1993). An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 (11) 1101–1113. doi:10.1109/34.244673.