Manuscript for 28th ICC, 2017

Using the A* Algorithm to Find Optimal Sequences for Area Aggregation

Dongliang Peng[1], Alexander Wolff[1], and Jan-Henrik Haunert[2]

Abstract Given two land-cover maps of different scales, we wish to find a sequence of small incremental changes that gradually transforms one map into the other. We assume that the two input maps consist of polygons that constitute a planar subdivision and belong to different land-cover classes. Every polygon in the small-scale map is the union of a set of polygons in the large-scale map. In each step of the sequence that we compute, the smallest area in the current map is merged with one of its neighbors. We do not select that neighbor according to a prescribed rule but define the whole sequence of pairwise merges at once, based on global optimization. An important requirement for such a method is a formalization of the problem in terms of optimization objectives and constraints, which we present together with a solution that is based on the so-called A* algorithm. This algorithm allows us to limit the exploration of the search space such that we can compute solutions of high quality in reasonable time. We tested the method with a dataset of the official German topographic database ATKIS and discuss our results.

Key words: Continuous map generalization, Land cover, Type change, Compactness, Graph, Optimization

[1]Dongliang Peng
Chair of Computer Science I, University of Würzburg, Germany
e-mail: dongliang.peng@uni-wuerzburg.de

[1]Alexander Wolff
Chair of Computer Science I, University of Würzburg, Germany
Orchid ID: orcid.org/0000-0001-5872-718X

[2]Jan-Henrik Haunert
Institute of Geodesy and Geoinformation, University of Bonn, Germany
e-mail: haunert@igg.uni-bonn.de

1 Introduction

A *land-cover map* is a planar subdivision in which each area belongs to a land-cover class or *type*. Suppose that there are two land-cover maps of different scales that cover the same spatial region. We consider the problem of finding a sequence of small incremental changes that gradually transforms the large-scale map (the *start map*) into the small-scale map (the *goal map*). We use this sequence to generate and show land-cover maps of intermediate scales (see Fig. 1). In this way, we can avoid large and sudden changes that distracts a user when zooming in or out.

With the same motivation, a strategy of hierarchical schemes has been proposed. This strategy generalizes a more-detailed representation to obtain a less-detailed representation based on small incremental changes, e.g., the Generalized Area Partitioning tree (for short, GAP-tree), which can be constructed if only the large-scale map is given (van Oosterom and Schenkelaars 1995) or if both the large-scale map and the small-scale map are given (Haunert et al. 2009).

Typically, the next change in such a sequence is determined locally, in a greedy fashion. If we insist on finding a sequence that is, according to some global measure, optimal, the problem becomes complicated.

We assume that there exist many-to-one relationships between the areas of the start map and the areas of the goal map. This assumption is based on the fact that there are many algorithms (e.g., Haunert and Wolff 2010; van Smaalen 2003; Cheng and Li 2006) that aggregate some land-cover areas into one single land-cover area. The input and the generalized results of these methods can be used as our input. We term the areas of the goal map *regions*. That is, every region is the union of a set of areas on the start map.

The type of a region may differ from the types of its components. For example, a small water area together with multiple adjacent forest areas may constitute a large forest region in the smaller scale. We assume, however, that every region contains at least one area of the same type. Our assumptions hold if the goal map has been produced with an automatic method for area aggregation, for example, with the method of Haunert and Wolff (2010). That method produces a land-cover map at a single output scale, given a land-cover map at a larger scale. It is based on global optimization and minimizes type changes as well as a cost for non-compact shapes while satisfying constraints on the sizes of the output regions. Although the method

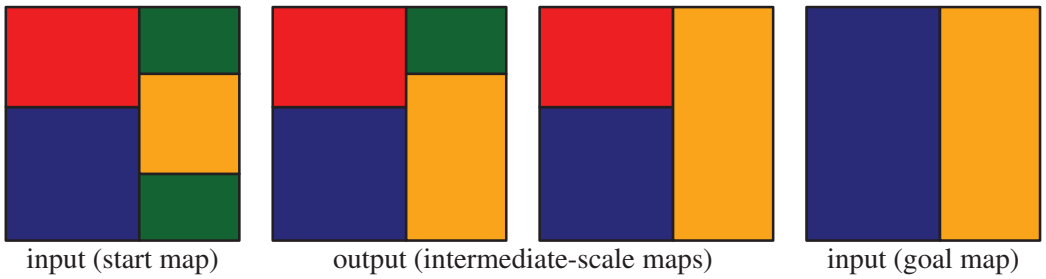


Fig. 1 The input and a possible output for an instance of our problem

of Haunert and Wolff leads to results of high quality, it does not yield land-cover maps at intermediate scales.

As stated by Chimani et al. (2014), "a sequence of well-generalized maps is not necessarily the same as a good sequence of maps." To generate a good sequence of maps, we must consider the sequence of maps as a whole instead of considering each intermediate-scale map separately. On the other hand, to simplify the problem, we first consider each region of the goal map (with its components on the start map) independent of the other regions. Once we have found a sequence of maps for each region, we integrate all the sequences into an overall sequence that transforms the start map into the goal map. Our aggregation sequence may be stored in data structures such as the GAP-face tree (van Oosterom 2005) or the map cube model (Timpf 1998) to support on-the-fly display.

Contribution. We give general ideas and basic concepts of our method, and analyze the size of our problem (see Sec. 2). We present a new global optimization approach to solve this problem. Our approach is based on the A* algorithm (see Sec. 3) which handles the combinatorial explosion that we will discuss in Sec. 2. Then, we do a case study using a dataset of the German topographic database ATKIS to assess the effectiveness and efficiency of our method (see Sec. 4). Finally, we conclude the paper (in Sec. 5).

We do not deal with the simplification of lines in this paper, since this can be handled separately from the aggregation of areas, for example, by using the method of Douglas and Peucker (1973) or Saalfeld (1999). Those methods can be used to set up the Binary Line Generalized tree (van Oosterom and Schenkelaars 1995) (for short, BLG-tree), which is a hierarchical data structure that defines a gradual line simplification process.

2 Preliminaries

To allow us to describe our method more easily, we assume that the goal map has only one region. On the start map, the same region consists of n land-cover areas (components). In other words, the union of the n land-cover areas is the only region on the goal map. We show how to compute an optimal aggregation sequence for a region. For a goal map with more than one region, we simply "interleaf" aggregation sequences of different regions (see Sec. 3.6).

To find a sequence of small changes that transforms the start map into the goal map, we require that every change involves only two areas of the current map. More precisely, in each step the smallest area a is merged with one of its neighbors (adjacent areas) b such that a and b are replaced by their union. The union must take over the type of either a or b. If the union uses the type of a, we say that area b is *aggregated into* area a, and vice versa. How to aggregate exactly is decided by optimizing a global cost function. This requirement ensures that the size of the smallest area in the map increases in each step and thus the sequence reflects a gradual reduction

of the map's scale. From another perspective, we consider the smallest area as the least important, instead of involving more rules for importance, to avoid making our problem too complicated. Even though the requirement reduces the number of possible solutions, there is still enough room for optimization since we leave open with which of its neighbors the smallest area is aggregated. We term a sequence of changes that adheres to our requirement an *aggregation sequence*.

Model. We consider a directed graph G, which we call the *subdivision graph* (see Fig. 2). The node set V of G contains a node for every possible map (or *subdivision*, including the start map, all possible intermediate-scale maps, and the goal map). The arc set E of G contains an arc uv between any two maps $u, v \in V$ such that v can be reached from u with a single aggregation operation involving the smallest area. Then, any directed path in G from the start map to the goal map defines a possible aggregation sequence. Our idea is to compute an optimal aggregation sequence by computing a minimum-weight path from start to goal. This idea obviously requires that the arc weights are set such that a minimum-weight start–goal path does actually correspond to an aggregation sequence of maximum cartographic quality. Moreover, putting the idea to practice is far from trivial since the graph G can be huge as we will argue below. Thus, exploring all of its nodes must be avoided. To limit the search space when computing an optimal path, we use the A* algorithm (Hart et al. 1968).

Notation. Each land-cover area is represented by a polygon with a type. A *patch* is a set of polygons whose union is connected. The patch also has the property type.

Recall that there are n land-cover areas on the start map. Hence, the desired aggregation sequence consists of $n-1$ steps. There are n subdivisions on a path from the start map to the goal map. We use $t \in \mathcal{T} = \{1, 2, \ldots, n\}$ to denote the *time* of the path. When $t = 1$, the subdivision consists of n patches, and there is only 1 patch when $t = n$. The subdivision graph consists of layers $\mathcal{L}_1, \ldots, \mathcal{L}_n$, where layer $\mathcal{L}_t = \{\mathcal{P}_{t,1}, \ldots, \mathcal{P}_{t,n_t}\}$ contains every possible subdivision $\mathcal{P}_{t,i}$ with $n-t+1$ patches (see Fig. 2).

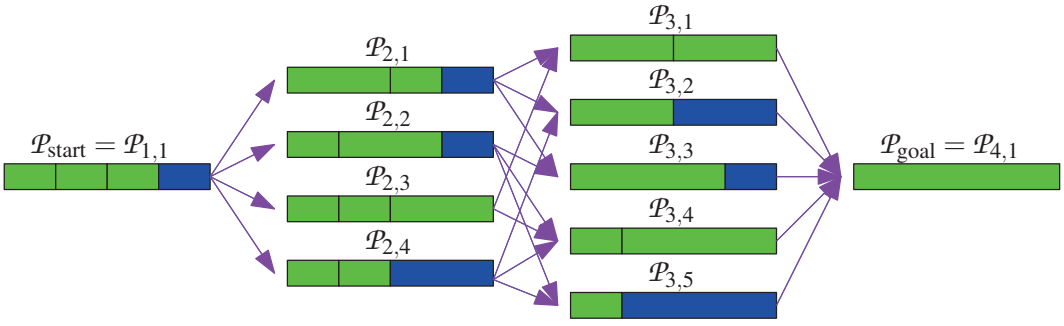


Fig. 2 The subdivision graph G. The nodes of the graph are the subdivisions. There is an arc from node $\mathcal{P}_{t,i}$ to node $\mathcal{P}_{t+1,j}$ if $\mathcal{P}_{t+1,j}$ is the result of aggregating the smallest area p of $\mathcal{P}_{t,i}$ with a neighbor of p.

Exponential lower bound. We now analyze the size of G. Consider an instance that consists of a start map $\mathcal{P}_{1,1}$ with $n = 2k$ unit squares in a row and a goal map $\mathcal{P}_{n,1}$ that is simply the union of the n squares, that is, an $(n \times 1)$-rectangle. One of the intermediate subdivisions in layer $\mathcal{L}_{k+1}$—let's call it $\mathcal{P}_{k+1,1}$—consists of a row of k (2×1)-rectangles. A lower bound for the number of nodes of G is the number of ways the (2×1)-rectangles in $\mathcal{P}_{k+1,1}$ can be subdivided into unit squares: each way corresponds to a unique node in a layer between the start map and $\mathcal{P}_{k+1,1}$. Clearly, there are exactly $2^k = 2^{n/2}$ ways.

3 Using the A* Algorithm

3.1 Formalizing Area Aggregation as a Pathfinding Problem

We model our aggregation problem as finding a shortest path in the subdivision graph G, which is weighted. We define the weight of an arc to be the cost of the corresponding aggregation step. We try to find a shortest path from subdivision $\mathcal{P}_{\text{start}}$, the start map, to $\mathcal{P}_{\text{goal}}$, the goal map. There are various algorithms that compute shortest paths in graphs, most notably Dijkstra's algorithm. As the graph G—our search space—can be of exponential size (see Sec. 2), we avoid computing the whole graph explicitly.

Once we have found a shortest path, we can generate an optimal aggregation sequence. Note that we only know subdivisions $\mathcal{P}_{\text{start}}$ and $\mathcal{P}_{\text{goal}}$ at the beginning. To save time and memory, we generate a subdivision $\mathcal{P}_{t,i}$ only when we want to visit it.

We decided to use the A* algorithm to find a shortest path from the start node $\mathcal{P}_{\text{start}}$ to the goal node $\mathcal{P}_{\text{goal}}$ because A* uses a clever best-first strategy. For a node $\mathcal{P}_{t,i}$, A* considers the exact cost of a shortest path from $\mathcal{P}_{\text{start}}$ to $\mathcal{P}_{t,i}$ and estimates the cost to get from $\mathcal{P}_{t,i}$ to $\mathcal{P}_{\text{goal}}$. The A* algorithm can be seen as a refinement of Dijkstra's algorithm where nodes are explored earlier if they are estimated to be close to the goal.

More formally, we define $g(\mathcal{P}_{t,i})$ to be the exact cost of a shortest path from $\mathcal{P}_{\text{start}}$ to $\mathcal{P}_{t,i}$, and we define $h(\mathcal{P}_{t,i})$ to be the estimated cost to get from $\mathcal{P}_{t,i}$ to $\mathcal{P}_{\text{goal}}$. Then, the total cost at node $\mathcal{P}_{t,i}$ is

$$F(\mathcal{P}_{t,i}) = g(\mathcal{P}_{t,i}) + h(\mathcal{P}_{t,i}). \tag{1}$$

If $h(\mathcal{P}_{t,i})$ is always bounded from above by the exact cost of a shortest path from $\mathcal{P}_{t,i}$ to $\mathcal{P}_{\text{goal}}$, A* is guaranteed to find a shortest path from $\mathcal{P}_{\text{start}}$ to $\mathcal{P}_{\text{goal}}$, that is, an optimal aggregation sequence. Using the estimation, A* is able to reduce the search space. The better the estimation, the more A* can reduce the search space. In the following, we show how we define $g(\mathcal{P}_{t,i})$ and $h(\mathcal{P}_{t,i})$.

3.2 Cost Functions

We consider two aspects of cost for our aggregation problem. First, we want to minimize the total type change, over all aggregation steps. Second, for the patches of a subdivision, we consider the one with the minimum compactness. We want to maximize the sum of the minimum compactnesses, over all intermediate subdivisions.

We define the cost of type change as follows. Suppose that we are at the step of generalizing from subdivision $\mathcal{P}_{s-1,i}$ to subdivision $\mathcal{P}_{s,j}$. In this step, patch p is aggregated into patch q. We denote the types of the two patches by $T(p)$ and $T(q)$. We define the cost of type change of this step by

$$f_{\text{type}}(\mathcal{P}_{s-1,i}, \mathcal{P}_{s,j}) = \frac{A_p}{A_R} \cdot \frac{Cost(T(p), T(q))}{T_{\max}}, \tag{2}$$

where A_p is the area of patch p, and A_R is the area of region R. The constant $T_{\max}$, the maximum cost over all type changes, is used to unify the cost of type change and is known from the input. The input specifies, for each pair (T_1, T_2) of types, the cost $Cost(T_1, T_2)$ of changing type T_1 to type T_2.

For an example, refer to Fig. 3a, where the brown and the green patch play the roles of patches p and q in Eq. 2. The shaded area in Fig. 4a shows the contribution of type change to the cost.

For path $\Pi = (\mathcal{P}_{1,i_1}, \mathcal{P}_{2,i_2}, \ldots, \mathcal{P}_{t,i_t})$, we define the exact cost of type change over all steps by

$$g_{\text{type}}(\Pi) = \sum_{s=2}^{t} f_{\text{type}}(\mathcal{P}_{s-1,i_{s-1}}, \mathcal{P}_{s,i_s}) \tag{3}$$

Recall that we also want to maximize the sum of the minimum compactnesses over all intermediate subdivisions. However, the A* algorithm tries to find a solution that minimizes a cost. To adapt our requirement to the A* algorithm, we modify our request to minimize the sum of a *shape* cost

$$f_{\text{shape}}(\mathcal{P}_{s,k}) = \frac{1 - \min_{p \in \mathcal{P}_{s,k}} c_p}{n-2}, \tag{4}$$

where $c_p = 2\sqrt{\pi A_p}/L_p$ is the compactness of patch p (Frolov 1975), and L_p is the perimeter of p. We use $n-2$ as the denominator to balance between the costs

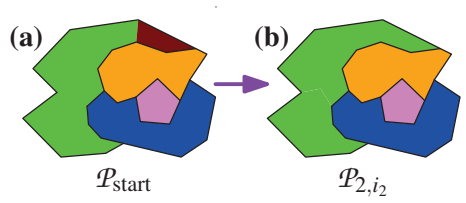


Fig. 3 An aggregation step, where the brown patch is aggregated into the green patch.

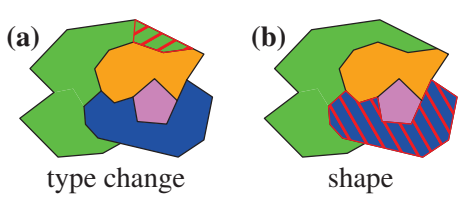


Fig. 4 The shaded parts show the contributions to the costs.

of type change and shape. Take Fig. 4b for example, the compactness of the shaded blue patch will be used to compute f_{shape} because the blue patch is the least compact one among the four patches.

For path $\Pi = (\mathcal{P}_{1,i_1}, \mathcal{P}_{2,i_2}, \ldots, \mathcal{P}_{t,i_t})$, we define the exact cost of shape over all intermediate subdivisions by

$$g_{\text{shape}}(\Pi) = \sum_{s=2}^{t-1} f_{\text{shape}}(\mathcal{P}_{s,i_s}). \tag{5}$$

Combining type change and shape, we define the total exact cost of path Π by

$$g(\Pi) = (1-\lambda) g_{\text{type}}(\Pi) + \lambda g_{\text{shape}}(\Pi), \tag{6}$$

where $\lambda \in [0,1]$ is a parameter to balance f_{type} and f_{shape}. In most cases, we use $\lambda = 0.5$ in our experiments. We want to find a path from $\mathcal{P}_{\text{start}}$ to $\mathcal{P}_{t,i}$ that minimizes, among all such paths, $g(\Pi)$. Slightly abusing notation, we denote the length (or cost) of this shortest path from $\mathcal{P}_{\text{start}}$ to $\mathcal{P}_{t,i}$ by $g(\mathcal{P}_{t,i})$.

To narrow down the search space, we set up estimation functions for type change (Sec. 3.3) and shape (Sec. 3.4). These functions are meant to direct A* towards the goal.

3.3 Estimating the Cost of Type-Change

As illustrated in Sec. 3.1, if the estimated cost is always a lower bound of the exact cost of a shortest path from $\mathcal{P}_{t,i}$ to $\mathcal{P}_{\text{goal}}$, A* is guaranteed to find a shortest path. To find a lower bound of the cost of type change, we simply assume that every patch will be aggregated into a patch with type T_{goal}, where T_{goal} is the type of the only patch on the goal map. As long as the cost of type change is a metric, this aggregation strategy indeed returns a lower bound. For subdivision $\mathcal{P}_{t,i}$, let $(\mathcal{P}_{t,i_t}, \mathcal{P}_{t+1,i_{t+1}}, \ldots, \mathcal{P}_{n,i_n})$, $\mathcal{P}_{t,i_t} = \mathcal{P}_{t,i}$ and $\mathcal{P}_{n,i_n} = \mathcal{P}_{\text{goal}}$, be the path that always changes the type of the smallest patch to T_{goal}. Then, the estimated cost of type change is

$$h_{\text{type}}(\mathcal{P}_{t,i}) = \sum_{s=t}^{n-1} f_{\text{type}}(\mathcal{P}_{s,i_s}, \mathcal{P}_{s+1,i_{s+1}}). \tag{7}$$

For an example, after the aggregation step of Fig. 3, we compute h_{type} according to the aggregation sequence of Fig. 5. Note that the step from $\mathcal{P}_{2,i_2}$ to $\mathcal{P}_{3,i_3}$ in Fig. 5 is impossible in reality because the violet patch cannot be assigned color green if none of its neighbors is green. However, this assignment is allowed for estimation because we may find a shortest path as long as the estimated cost is no more than the exact cost of a shortest path.

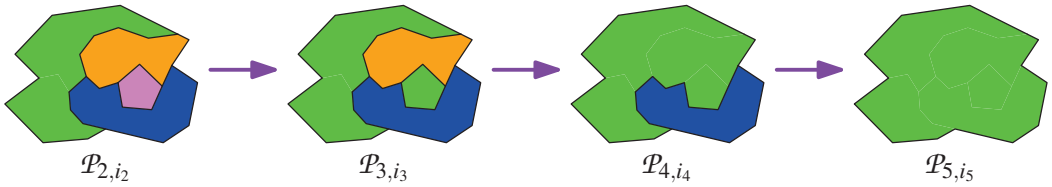


Fig. 5 An "aggregation sequence" for computing h_{type}, based on the aggregation result of Fig. 3b

3.4 Estimating the Cost of Shape

We find a lower bound of $f_{\text{shape}}(\mathcal{P}_{s,k})$, in Eq. 4, based on the edge number of the boundaries. We use n_{ext} to denote the edge number of the region's exterior boundaries. As the exterior boundaries will not be changed by aggregation, n_{ext} is a constant. We use $B(\mathcal{P}_{s,k})$ to denote the set of interior boundaries of subdivision $\mathcal{P}_{s,k}$. Note that a boundary between two patches is not necessarily connected; for an example, see the orange boundary in Fig. 7a. The number of interior edges is

$$e_{\text{int}}(\mathcal{P}_{s,k}) = \sum_{b \in B(\mathcal{P}_{s,k})} \|b\|, \tag{8}$$

where $\|b\|$ is the edge number of boundary b. Considering that each interior edge is used by two patches, the total number of edges over all patches is

$$e_{\text{sum}}(\mathcal{P}_{s,k}) = e_{\text{ext}} + 2e_{\text{int}}(\mathcal{P}_{s,k}). \tag{9}$$

To compute a lower bound of $f_{\text{shape}}(\mathcal{P}_{s,k})$ is to find an upper bound of $\min_{p \in \mathcal{P}_{s,k}} c_p$. As the regular polygon is the most compact one among all polygons with the same number of edges, we construct regular polygons using the edges we have, ignoring their lengths. A regular polygon with more edges is more compact than one with fewer edges. We now suppose that we know an upper bound of edge number $e_{\text{sum}}(\mathcal{P}_{s,k})$. We denote this upper bound by $E(\mathcal{P}_{s,k})$. We will show how to compute $E(\mathcal{P}_{s,k})$ later. Now, we define the upper bound of $\min_{p \in \mathcal{P}_{s,k}} c_p$ based on $E(\mathcal{P}_{s,k})$.

At time s, there are $n-s+1$ patches. As each patch has 3 edges at least, $E(\mathcal{P}_{s,k}) \geq 3(n-s+1)$. If we are allowed to freely construct the patches, we should evenly distribute the edges and construct $n-s+1$ regular patches in order to maximize the minimum compactness. In this case, each patch has

$$\rho_{s,i} = \lfloor E(\mathcal{P}_{s,k})/(n-s+1) \rfloor$$

edges at least. The smallest compactness of such a set of regular patches is

$$C_{\min}(\mathcal{P}_{t,i}) = \sqrt{\frac{\pi}{\rho_{s,i}} \Big/ \tan\frac{\pi}{\rho_{s,i}}}.$$

This value describes the largest minimum-compact situation, and thus is an upper bound for $\min_{p\in\mathcal{P}_{s,k}} c_p$. The lower bound of $f_{\text{shape}}(\mathcal{P}_{s,k})$ based on the edge number can be computed, according to Eq. 4, by

$$l(\mathcal{P}_{s,k}) = \frac{1 - C_{\min}(\mathcal{P}_{s,k})}{n-2}.$$

In our case, the patches can never be circles, thus $C_{\min}(\mathcal{P}_{s,k}) < 1$ and $l(\mathcal{P}_{s,k}) > 0$. This property is important when we overestimate (see Sec. 3.5).

One remaining task is to compute upper bound $E(\mathcal{P}_{s,k})$. Removing a boundary may result in joining several boundaries to one (see Fig. 6 for an example). At each step of the aggregation, we remove one interior boundary. To make sure that we have an upper bound, we always remove the original interior boundary, from the set of interior boundaries $B(\mathcal{P}_{t,i})$, with the fewest edges (see Fig. 7 for an example). Then at each step we compute $E(\mathcal{P}_{s,k})$ according to Eqs. 8 and 9. Note that we do this only for estimation. (When computing the real compactness, we consider the boundary length and the area of a patch; see Sec. 3.2.)

Let $(\mathcal{P}_{t,j_t}, \mathcal{P}_{t+1,j_{t+1}}, \ldots, \mathcal{P}_{n,j_n})$, $\mathcal{P}_{t,j_t} = \mathcal{P}_{t,i}$ and $\mathcal{P}_{n,j_n} = \mathcal{P}_{\text{goal}}$, be the path from $\mathcal{P}_{t,i}$ to $\mathcal{P}_{\text{goal}}$ where we repeatedly remove the interior boundary with the fewest edges. We define the estimated cost of shape by

$$h_{\text{shape}}(\mathcal{P}_{t,i}) = \sum_{s=t}^{n-1} l(\mathcal{P}_{s,j_s}), \tag{10}$$

where we define $l(\mathcal{P}_{1,i_1}) = 0$.

We define the total estimated cost to be

$$h(\mathcal{P}_{t,i}) = (1-\lambda)h_{\text{type}}(\mathcal{P}_{t,i}) + \lambda h_{\text{shape}}(\mathcal{P}_{t,i}),$$

where λ is the same as in Eq. 6. Recall that we compute the total cost at node $\mathcal{P}_{t,i}$ by Eq. 1.

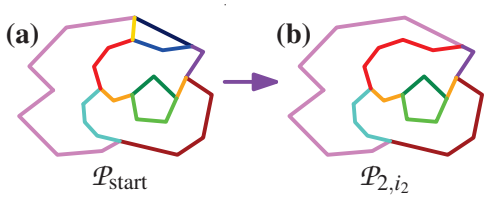


Fig. 6 An example to show the change of the remained boundaries after removing a boundary. **a** Original boundaries. **b** The boundaries after removal. This example corresponds to the aggregation step in Fig. 3. Note that the pink boundary and the navy boundary in **a** join together after we remove the yellow boundary. So do the red boundary and the blue boundary.

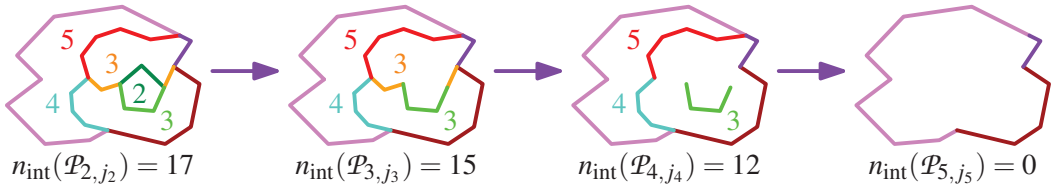


Fig. 7 An "aggregation" sequence for computing h_{shape} (see Eq. 10) based on the number of edges. At each step we remove the boundary with the fewest edges. At the last step, i.e., from $\mathcal{P}_{4,j_4}$ to $\mathcal{P}_{5,j_5}$, we simply remove all the interior boundaries because we know the exact shape from the input. The numbers represent the edge numbers of the interior boundaries. This example corresponds to the aggregation step in Fig. 3.

3.5 Overestimation

Since the number of subdivisions can be exponential, it can happen that we run out of main memory before we find an optimal solution. To solve this problem, we overestimate the cost. Overestimation is a popular technique in the A* algorithm. For example, Pohl (1973) used a strategy of dynamic weighting to overestimate. We propose another strategy in order to find a solution faster. We first try finding a solution by the A* algorithm. If we cannot find one after we have visited a predefined number M of nodes of G, then we overestimate the cost by a factor of 2 and try again. We repeat this until we have a feasible solution. We revise Eq. 1 that defines the total cost at a node $\mathcal{P}_{t,i}$ to

$$F(\mathcal{P}_{t,i}) = g(\mathcal{P}_{t,i}) + K \cdot h(\mathcal{P}_{t,i}), \tag{11}$$

where $K = 2^k$ is the overestimation factor, and $k \geq 0$ is the number of repetitions. Whenever overestimating ($k \geq 1$), we cannot guarantee optimality anymore. We need to make sure that $h(\mathcal{P}_{t,i}) > 0$, otherwise the overestimation is of no effect.

3.6 Integrating Aggregation Sequences of Different Regions

We compute an aggregation sequence for each region. Then, we integrate the sequences of different regions with respect to the order of the smallest patches (see Fig. 8 for an example). This integration is similar to the merge step in the sorting algorithm Mergesort (see Cormen et al. 2009, pp. 29–37).

4 Case Study

We have implemented our methods based on C# (Microsoft Visual Studio 2012) and ArcObjects SDK 10.2. We ran our case study under 64-bit Windows 7 on a 3.3 GHz dual core CPU with 8 GB RAM. We measured processing time by the

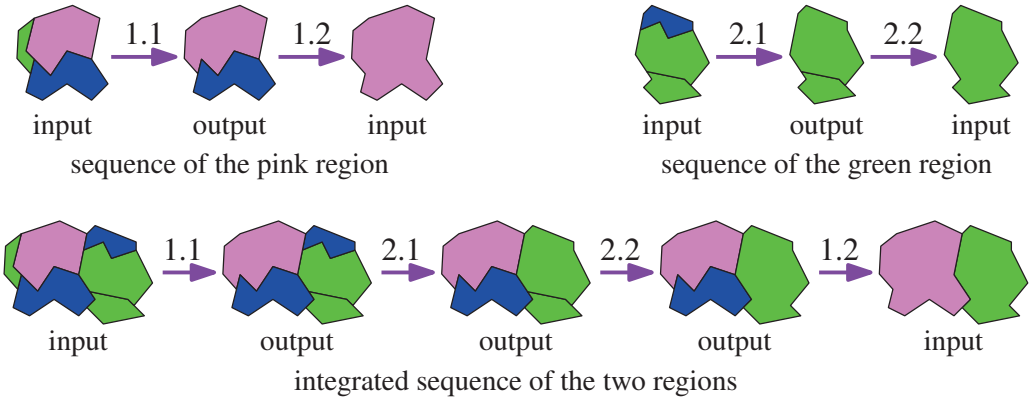


Fig. 8 Integrating two aggregation sequences of different regions: the resulting sequence contains the given sequences as subsequences and always takes the subdivision with smallest patch next

built-in C# class *Stopwatch*. We tested our method on a dataset from the German topographic database ATKIS DLM 50. The dataset represents the place "Buchholz in der Nordheide" at scale 1 : 50,000. Our start map is the map after the collapse of areas by Haunert (2009, pp. 61–66), which has 5,537 polygons. Our goal map was generalized from the start map by Haunert and Wolff (2010) setting the scale at 1 : 250,000 (see Fig. 10). The goal map has 734 polygons, which means that there are $N = 734$ regions. We used a tree-based method introduced by Schwering (2008) to define the cost of type change; the cost is the distance between two types in the type hierarchy (see Fig. 9). For example, the distance between type fence and type street is 4. In this tree, the maximum distance is 6. Therefore, $T_{\max} = 6$ in our case study for Eq. 2.

We overestimated whenever we could not find a solution after having visited M nodes (see Sec. 3.5). We tried $M = 200{,}000$ and $M = 400{,}000$. The results are shown in Table 1. When $M = 200{,}000$, we have found optimal aggregation sequences for 692 of the 734 regions (94.3%); see column #over in Table 1, where we overestimated for the other 42 regions.

Table 1 A comparison of the A* algorithm for two different settings that depend on the maximum number of nodes, M, that we allowed the A* algorithm to explore. #over is the number and percentage of regions (out of 734) that needed overestimation. Variable k_{sum} is the total repetitions. Symbols #nodes and #arcs are the total numbers of nodes and arcs that the A* algorithm visited. (For instances where we needed overestimation, only the final attempt is counted.) We used g_{t}, g_{s}, and g to denote the sums of $g_{\text{type}}(\mathcal{P}_{\text{goal}})$, $g_{\text{shape}}(\mathcal{P}_{\text{goal}})$, and $g(\mathcal{P}_{\text{goal}})$, respectively, over all instances (see Eqs. 3, 5, and 6). The percentage in the time column is the fraction of the total runtime spent on solving the instances that needed overestimation

M	#over	k_{sum}	#nodes	#arcs	g_{t}	g_{s}	g	Time (h)
200,000	42 (5.7%)	113	$2.4 \cdot 10^6$	$4.9 \cdot 10^6$	51.5	259.1	155.3	0.6 (88.7%)
400,000	39 (5.3%)	96	$4.3 \cdot 10^6$	$8.9 \cdot 10^6$	51.4	259.0	155.2	1.1 (92.2%)

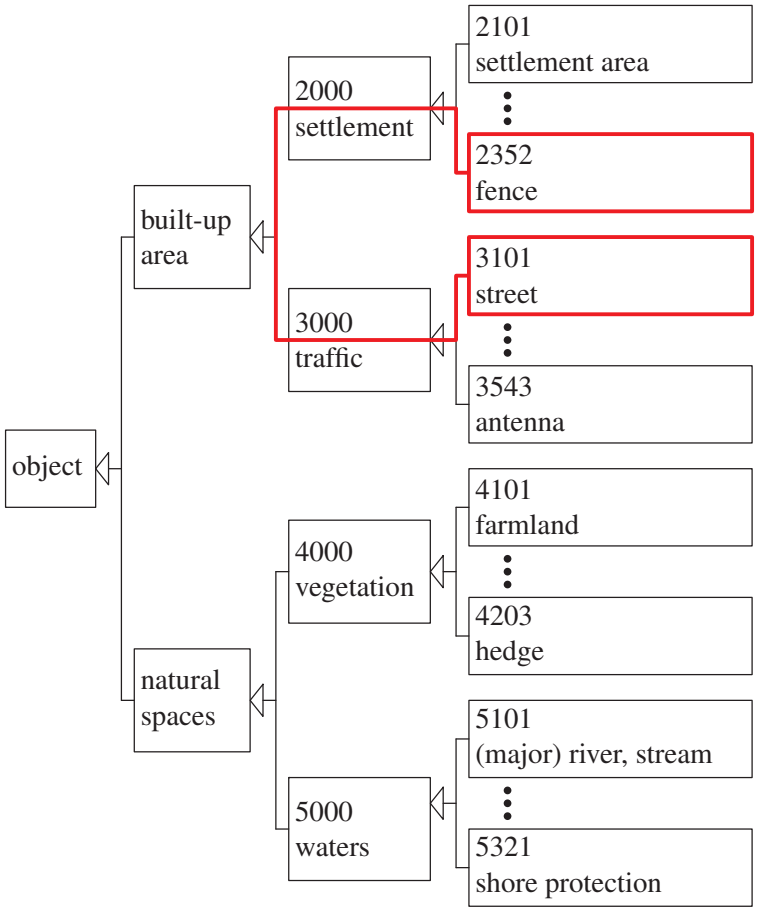
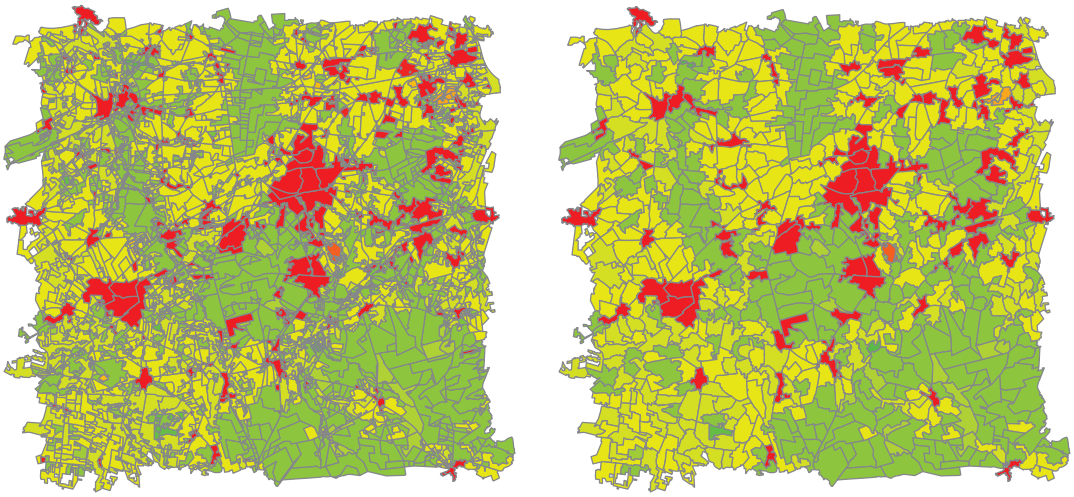


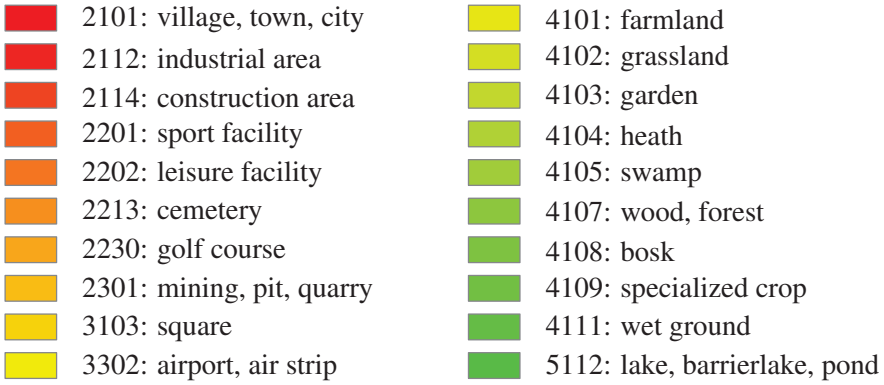
Fig. 9 The type distance defined in our case study. For example, the distance between type fence and type street is 4.

Recall Eq. 11, for region with ID i we define k_i as the least repetitions that we do to find a feasible solution, and we define the least feasible overestimation factor $K_i = 2^{k_i}$. We define the total repetitions as $k_{\text{sum}} = \sum_{i=1}^{N} k_i$, where N is the number of regions. After we increased M to 400,000, we found optimal aggregation sequences for only three more regions, but the total repetitions k_{sum} decreased quite a bit, from 113 to 96. The numbers of regions that needed a certain overestimation factor are shown in Fig. 11. Besides, we visited more arcs and nodes, used more time, but got (slightly) less cost when $M = 400{,}000$. Although the number of regions that needed overestimation is relatively small, A* spent most of the running time on these few regions: 5.7% and 5.3% of the regions cost 88.7% and 92.2% of the total running time, respectively (see Table 1).

The details of some regions are presented in Table 2. According to the entries with overestimation factor $K_i = 1$, we often have $R_{\text{type}} = 1$ and a large R_{shape}. Recall that when $K = 1$, $R_{\text{type}} \geq 1$ and $R_{\text{shape}} \geq 1$; see Sec. 3.1. $R_{\text{type}} = 1$ means that we have the best estimation for the cost of type change. A large R_{shape} means that our estimation for the cost of shape is poor.



Start map, at scale 1 : 50,000 Goal map, at scale 1 : 250,000



The 20 land-cover types appearing in our data

Fig. 10 The data of our case study

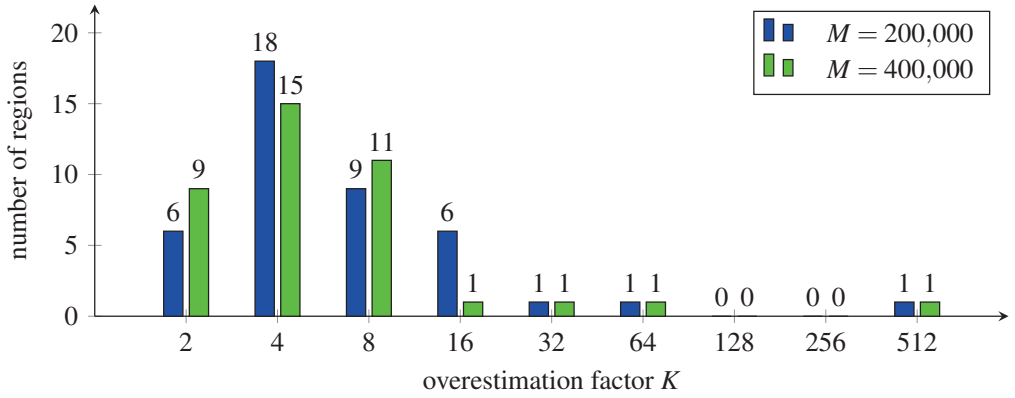


Fig. 11 The numbers of regions where our approach was forced to use the given overestimation factors in order to find a solution without exploring more than $M \in \{200{,}000,\ 400{,}000\}$ nodes of the subdivision graph.

Table 2 The costs in detail of some regions, where $M = 200{,}000$. The parameters n and m are the numbers of patches and their adjacencies on the start map, respectively. Parameter K_i is the overestimation factor defined in this section. We evaluate the quality of our estimations for type change and shape by listing the numbers $R_{\text{type}} = g_{\text{type}}(\mathcal{P}_{\text{goal}})/h_{\text{type}}(\mathcal{P}_{\text{start}})$ and $R_{\text{shape}} = g_{\text{shape}}(\mathcal{P}_{\text{goal}})/h_{\text{shape}}(\mathcal{P}_{\text{start}})$. Note that if $h_{\text{type}}(\mathcal{P}_{\text{start}}) = 0$, we define $R_{\text{type}} = 1$.

ID	n	m	K_i	R_{type}	R_{shape}	Time (s)
543	20	40	512	1.000	0.516	175.7
94	32	74	64	0.016	4.577	163.6
190	22	45	32	0.034	21.535	84.9
301	30	61	16	0.066	21.895	78.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮
569	15	32	2	0.514	69.011	16.7
358	21	32	1	1.000	1020.133	1.3
97	20	35	1	1.000	179.132	0.9
257	20	33	1	1.000	463.178	5.9
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Interestingly enough, region 543 has neither the most patches nor the most adjacencies, but it needs the largest overestimation factor, which is 512 (see the first entry in Table 2). The reason is that all the patches of this region are with the same type (see region 543 with 20 patches in Fig. 12). The type change has no contribution to the estimation, while the estimation of shape is poor to help find an aggregation sequence.

We show the intermediate aggregation results of some regions in Fig. 12. Although we had to overestimate for regions 543 and 436, the results seem fine (see the first row and the second row of Fig. 12). In Fig. 12, the third row shows an optimal aggregation sequence for region 77. However, we noticed some inappropriate aggregations (see Fig. 13 for an example). This inappropriate aggregation could have been caused by the overestimation or by our definition of the shape cost.

It turned out that the objective function g seems rather robust against changes of λ. For region 77, we varied λ in the range $\{0.1, 0.5, 0.9\}$, but the resulting aggregation sequences stayed almost the same. When we used even larger values for λ (e.g., 0.99), the (good) influence of type change was mitigated and, as a result, we needed to overestimate (with factor $K = 8$).

5 Conclusions

In this paper, we proposed a method for finding an optimal aggregation sequence between two maps of land-cover areas based on the A* algorithm. This method reduces the search space of attaining an optimal result. We have a good estimation for the cost of type change, which helps a lot to reduce the search space. In contrast, our estimation for the cost of shape is poor. If we investigate the shapes of the patches

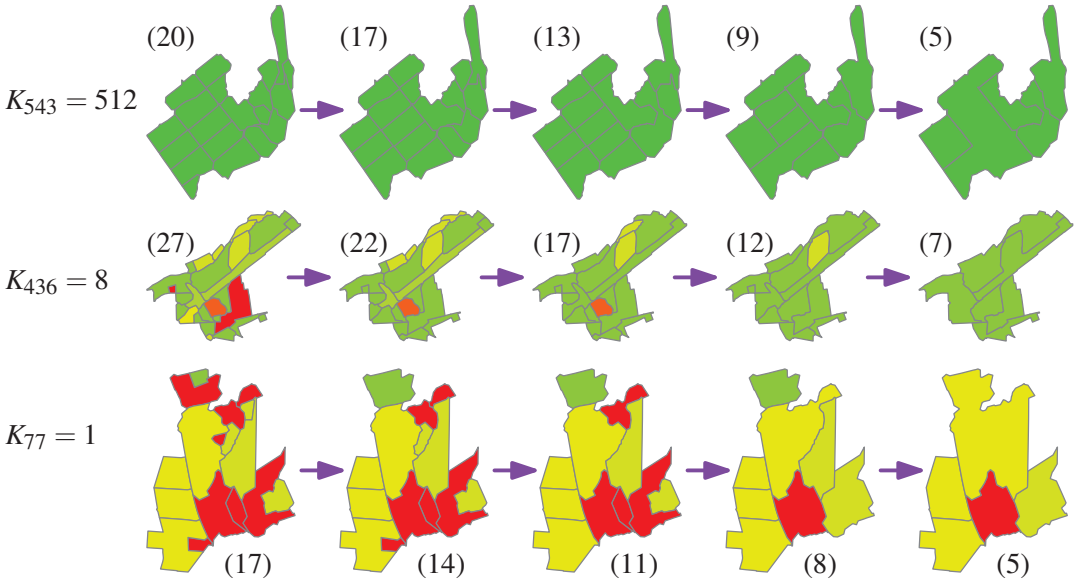


Fig. 12 Some intermediate aggregation results of Regions 543, 436, and 77. The numbers in parentheses are the numbers of remaining patches.

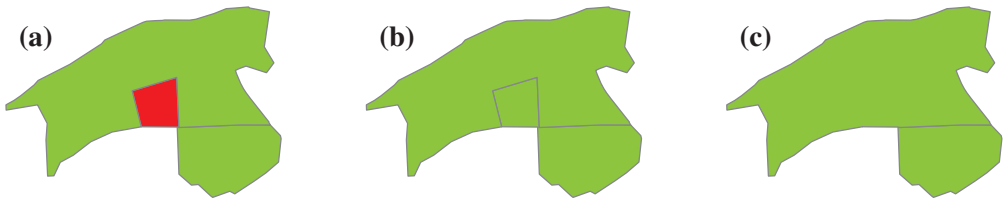


Fig. 13 An inappropriate result and the result we prefer. Figures **a** and **b** are extracted from region 436 when there are 27 and 22 patches (see Fig. 12). Our method aggregates the red patch of figure **a** into the lower-right patch (see figure **b**). We prefer aggregating the red patch into the upper green patch (see figure **c**).

more carefully, we may improve our estimation for the cost of shape. Our current method did not manage to find optimal aggregation sequences for some of the regions in our datasets. For these regions, we overestimate in order to find feasible aggregation sequences. Note that a larger overestimation factor does not necessarily mean a poorer aggregation sequence. We compared our A*-based method with a method based on integer linear programming, and found that, within a fixed time frame, the A*-based method managed to solve more instances to optimality than the integer linear program.

To keep our problem concise, we did not incorporate many cartographic rules. An example is that a settlement is important and should be remained longer if the settlement is surrounded by farmlands. We did not consider the case like aggregating farmland with a hedge results in a land-cover area with type vegetation. These issues may be considered in our future work. Our method provides a way for continuous generalization of land-cover areas. We proposed our cost functions (i.e., type change

and shape) based on cartographic requirements. However, to what extent our output remains faithful to real-world geography still needs to be evaluated.

Acknowledgments

We thank Thomas C. van Dijk, Joachim Spoerhase, and Sabine Storandt for their valuable suggestions.

References

Cheng T and Li Z (2006) Toward quantitative measures for the semantic quality of polygon generalization. In: *Cartographica* 41.2, pp. 487–499

Chimani M, van Dijk TC, and Haunert J-H (2014) How to eat a graph: computing selection sequences for the continuous generalization of road networks. In: *Proc. 22nd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACMGIS'14)*, pp. 243–252

Cormen TH, Leiserson CE, Rivest RL, and Stein C (2009) *Introduction to algorithms, third edition*. The MIT Press

Douglas DH and Peucker TK (1973) Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In: *Cartographica* 10.2, pp. 112–122

Frolov YS (1975) Measuring the shape of geographical phenomena: a history of the issue. In: *Soviet Geography* 16.10, pp. 676–687

Hart PE, Nilsson NJ, and Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. In: *IEEE Transactions on Systems, Science, and Cybernetics* 4.2, pp. 100–107

Haunert J-H (2009) Aggregation in map generalization by combinatorial optimization. Dissertation. Leibniz Universität Hannover, Germany

Haunert J-H and Wolff A (2010) Area aggregation in map generalisation by mixed-integer programming. In: *International Journal of Geographical Information Science* 24.12, pp. 1871–1897

Haunert J-H, Dilo A, and Oosterom P van (2009) Constrained set-up of the tGAP structure for progressive vector data transfer. In: *Computers and Geosciences* 35.11, pp. 2191–2203

Pohl I (1973) The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In: *Proceedings of the 3rd international joint conference on artificial intelligence*. IJCAI'73. Stanford, USA: Morgan Kaufmann Publishers Inc., pp. 12–17

Saalfeld A (1999) Topologically consistent line simplification with the Douglas–Peucker algorithm. In: *Cartography and Geographic Information Science* 26.1, pp. 7–18

Schwering A (2008) Approaches to semantic similarity measurement for geo-spatial data: a survey. In: *Transactions in GIS* 12.1, pp. 5–29

Timpf S (1998) Hierarchical structures in map series. Dissertation. Technical University Vienna, Austria

van Oosterom P (2005) Variable-scale topological datastructures suitable for progressive data transfer: the GAP-face tree and GAP-edge forest. In: *Cartography and Geographic Information Science* 32.4, pp. 331–346

van Oosterom P and Schenkelaars V (1995) The development of an interactive multi-scale GIS. In: *International Journal of Geographical Information Systems* 9.5, pp. 489–507

van Smaalen J (2003) Automated aggregation of geographic objects. Dissertation. Wageningen University, The Netherlands.