Abstract. The paper presents a scalable approach for generalization of large land-use datasets using a partitioning in a spatial database and fast generalization algorithms. In the partitioning step the dataset is split into rectangular overlapping tiles. These are processed independently and then composed into one result. For each tile semantic and geometric generalization operations are performed to remove too small features from the dataset. The generalization approach is composed of several steps consisting of topologic cleaning, aggregation, feature partitioning, identification of mixed feature classes to form heterogeneous classes and simplification of feature outlines. The workflow will be presented with examples for generating CORINE Land Cover (CLC) features from the high resolution German authoritative land-use dataset of the whole area of Germany (DLM-DE). The results will be discussed in detail, including runtimes as well as dependency of the result on the parameter setting.
1 Introduction

1.1 Project Background

The European Environment Agency (EEA) collects the Coordinated Information on the European Environment (CORINE) Land Cover (CLC) dataset to monitor the land-use changes in the European Union. The member nations have to deliver this data every few years. Traditionally this dataset was derived from remote sensing data. However, the classification of land-use from satellite images in shorter time intervals becomes more cost intensive.

Therefore in Germany the federal mapping agency (BKG) investigates an approach of deriving the land cover data from topographic information. The BKG collects the digital topographic landscape models (ATKIS Base DLM) from all federal states. The topographic base data contains up-to-date land-use information. This data will be transformed to a high resolution land-use dataset called DLM-DE. After this transformation there are still some differences between DLM-DE and CLC. Table 1 summarizes the main characteristics of the two datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CORINE LC</th>
<th>DLM-DE LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>1:100 000</td>
<td>&gt; 1:10 000</td>
</tr>
<tr>
<td>source</td>
<td>satellite images</td>
<td>aerial images, cadastre</td>
</tr>
<tr>
<td>min. area size</td>
<td>25 ha</td>
<td>&lt; 1 ha</td>
</tr>
<tr>
<td>min. width</td>
<td>100 m</td>
<td>&lt; 10 m</td>
</tr>
<tr>
<td>Classes of heterogeneous</td>
<td>4 / 2 relevant</td>
<td>marginal, mostly separated in its homogeneous components</td>
</tr>
<tr>
<td>agricultural use</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of ATKIS and CLC

1.2 CORINE Land Cover (CLC)

CORINE Land Cover is a polygon dataset in the form of a planar partitioning (or tessellation): polygons do not overlap and cover the whole area without gaps. The scale is 1:100 000. Each polygon has a minimum area of 25 hectare and a minimum width of 100 meter. There are no adjacent polygons with the same land-use class as these have to be merged.
Land cover is classified hierarchically into 46 classes in three levels, for which a three digit numerical code is used. The first and second level groups are:

- 1xx artificial (urban, industrial, mine)
- 2xx agricultural (arable, permanent, pasture, heterogeneous)
- 3xx forest and semi-natural (forest, shrub, open)
- 4xx wetland (inland, coastal)
- 5xx water (inland, marine)

In CLC there are four aggregated classes for heterogeneous agricultural land-use. Such areas are composed of small areas of different agricultural land-use. In Germany only two of these four classes occur. Class 242 is composed of alternating agricultural uses (classes 2xx). Class 243 is a mixture of agricultural and (semi-) natural areas.

### 1.3 DLM-DE LC

The land cover (LC) layer of the Digital Landscape Model (DLM) of Germany (DE) is a new product of Germany’s national mapping agency (BKG). DLM-DE LC is derived by a semantic generalization from the Authoritative Topographic Cartographic Information System (ATKIS) which is Germany’s large scale topographic landscape model. After selecting all relevant features from ATKIS the topological problems like overlaps and gaps are solved automatically using appropriate algorithms. The reclassification to the CLC nomenclature is done using a translation table which takes the ATKIS classes and their attributes into account. In the cases where a unique translation is not possible, a semi-automatic classification from remote sensing data is used. The scale of DLM-DE is approx. 1:10 000. The minimum area for polygons is less than one hectare.

### 1.4 Automatic derivation of CLC from DLM-DE

The aim of the project is the automated derivation of CLC data from ATKIS. This derivation can be considered as a generalization process, as it requires both thematic selection and reclassification, and geometric operations due to the reduction in scale. Therefore, the whole workflow consists of two main parts. The first part is a model transformation and consists of the extraction, reclassification and topological correction of the data. The derived model is called DLM-DE LC. The second part, the generalization part, which will be described in more detail in this paper, is the aggrega-
tion, classification and simplification for the smaller scale. For that purpose a sequence of generalization operations is used. The operators are dissolve, aggregate, split, simplify and a heterogeneous class filter. The program computing the generalization is called CLC-generator.

The classification of agricultural heterogeneous areas to 24x-classes in the case that a special mixture of land-uses occurs is one of the main challenges. The difficulty is to separate these areas from homogeneous as well as from other heterogeneous classes.

### 1.5 Scalability

Another challenge of the project is the huge amount of data. The DLM-DE LC contains ten million polygons. Each polygon consists in average of thirty points, so one has to deal with 300 million points, which is more than a standard PC can store in main memory. While fast algorithms and efficient data structures reduce the required time for the generalization, we have developed a partitioning and composition strategy in order to overcome problems due to memory limitations when processing large datasets. We store the source data for the generalization process in a spatial database system and divide it into smaller partitions, which can efficiently be handled by the CLC-generator on standard computers. The resulting CLC-datasets for the individual tiles are then composed into one dataset within the database.

To ensure consistency, i.e. to get identical results from partitioned and unpartitioned execution, some redundancy is added to the partitions in the form of overlapping border regions. This redundancy is removed in the composition phase and geographic objects residing at the border of different partitions are reconciled.

The amount of redundancy added can be controlled by the width of the border regions. As bigger regions cause longer running times of the generalization, we are interested in using values as small as possible while still ensuring consistency. Another parameter influencing performance is the number of partitions. The tiles have to be small enough to avoid memory limitations but a fine-granular partitioning leads to more composition overhead. We present experiments targeted at finding the optimal values for these parameters.
2 Related Work

CORINE Land Cover (Büttner et al. 2006) is being derived by the European States (Geoff et al. 2007). The Federal Agency of Cartography and Geodesy attempts to link the topographic database with the land-use data. To this end, transformation rules between CLC and ATKIS have been established (Arnold 2009).

As described above, the approach uses different generalization and interpretation steps. The current state of the art in generalization is described in Mackaness et al. (2007). The major generalization step needed for the generalization of land-use classes is aggregation. The classical approach for area aggregation was given by Oosterom (1995), the so-called GAP-tree (Generalized Area Partitioning). In a region-growing fashion areas that are too small are merged with neighboring areas until they satisfy the size constraint. The selection of the neighbor to merge with depends on different criteria, mainly geometric and semantic constraints, e.g. similarity of object classes or length of common boundary. This approach is implemented in different software solutions (e.g. Podrenek, 2002). Although the method yields areas of required minimum size, there are some drawbacks: a local determination of the most compatible object class can lead to a high amount of class changes in the whole dataset. Also, objects can only survive the generalization process, if they have compatible neighbors. The method by Haunert (2008) is able to overcome these drawbacks. He is also able to introduce additional constraints e.g. that the form of the resulting objects should be compact. The solution of the problem has been achieved using an exact approach based on mixed-integer programming (Gomory, 1958), as well as a heuristic approach using simulated annealing (Kirkpatrick 1983). However, the computational effort for this global optimization approach is very high.

Collapse of polygon features corresponds to the skeleton operation, which can be realized using different ways. A simple method is based on triangulation; another is medial axis or straight skeleton (Haunert & Sester, 2008).

The identification of mixed classes is an interpretation problem. Whereas interpretation is predominant in image understanding where the task is to extract meaningful objects from a collection of pixels (Lillesand & Kiefer, 1999), also in GIS-data interpretation is needed, even when the geo-data are already interpreted. E.g. in our case although the polygons are semantically annotated with land-use classes, however, we are looking for a higher level structure in the data which evolves from a spatial arrangement.
of polygons. Interpretation can be achieved using pattern recognition and model based approaches (Heinzle & Anders, 2007).
Partitioning of spatial data has extensively been investigated in the area of parallel spatial join processing. In (Zhou et al., 1998) a framework for partitioning spatial join operations in a parallel computer environment is introduced and the impact of redundancy on performance is studied. Newer work (Meng et al., 2007) presents an improved join method for decomposing spatial datasets in a parallel database system. Spatial joins only need to collect partitionwise results, maybe including duplicate elimination. Our task of generalization however needs geometric composition of results, and context dependencies have to be observed.

3 Generalization Approach

3.1 Data and index structures
An acceptable run time for the generalization of ten million polygons can only be reached with efficient algorithms and data structures. For topology depending operations a topologic data structure is essential. For spatial searching a spatial index structure is needed; furthermore, also structures for one-dimensional indexing are used.
In the project we use an extended Doubly Connected Edge List (DCEL) as topologic structure. A simple regular grid (two-dimensional hashing) is used as spatial index for nodes, edges and faces. For the DLM-DE a grid width of 100 m for points and edges (<10 features per cell) and 1000 m for faces (40 faces per cell) leads to nearly optimal speed.

3.2 Topological cleaning
Before starting the generalization process, the data has to be imported into the topological structure. In this step we also look for topological or semantic errors. Each polygon is checked for a valid CLC class. Small sliver polygons with a size under a threshold of e.g. 1 m² will be rejected. A snapping with a distance of 1 cm is done for each inserted point. With a point in polygon test and a test for segment intersection overlapping polygons are detected and also rejected. Holes in the tessellation can be easily found by building loops of the half-edges which not belong to any face. Loops with a positive orientation are holes in the dataset.
3.3 Generalization operators

**Dissolve**

The dissolve operator merges adjacent faces of the same class. For this purpose the edges which separate such faces will be removed and new loops are built.

**Aggregate**

The aggregation step aims at guarantying the minimum size of all faces. The aggregation operator in our case uses the simple greedy algorithm described by Oosterom (1995). It starts with the smallest face and merges it to a compatible neighbor. This fast algorithm is able to process the dataset sequentially. There are different options to determine compatible neighbors. The criterion can be:

- the semantic compatibility (semantic distance),
- the geometric compactness
- or a combination of both.

<table>
<thead>
<tr>
<th>CiC</th>
<th>111</th>
<th>112</th>
<th>121</th>
<th>122</th>
<th>123</th>
<th>124</th>
<th>131</th>
<th>132</th>
<th>133</th>
<th>134</th>
<th>141</th>
<th>142</th>
<th>143</th>
<th>144</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>112</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>121</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>122</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>124</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>131</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>132</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>133</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>134</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>141</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>142</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>211</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 1. Small extract of the CLC priority matrix

The semantically nearest partner can be found using a priority matrix. We use the matrix from the CLC technical guide (Bossard, Feranec & Otahel 2000) (Figure 1). The priority values are from an ordinal scale, so their differences and their values in different lines should not be compared. The matrix is not symmetric, as there may be different ranks when going from one object to another than vice versa (e.g. settlement → vegetation). Priority value zero is used if both faces have the same class. The higher the
priority value, the higher is the semantic distance. Therefore the neighbor with the lowest priority value is chosen.

As geometric criterion the length of the common edge is used. A shorter perimeter leads to better compactness. So the maximum edge length has to be reduced to achieve a better compactness.

The effects of using the criteria separate are shown in a real example in Figure 2. The semantic criterion leads to non-compact forms, whereas the geometric criterion is more compact but leads to a large amount of class change. The combination of both criteria allows merging of semantically more distant objects, if the resulting form is more compact. This leads to Formula 1.

$$distance(A,B) = \frac{\text{priority}}{\text{length}}$$  \hspace{1cm} (1)

The formula means that a b-times longer shared edge allows a neighbor with the next worse priority. The base b allows to weight between compactness and semantic proximity. A value of b=1 leads to only compact results, a high value of b leads to semantically optimal results. Using the priority values is not quite correct; it is only a simple approximation for the semantic distance.

Another application of the aggregation operation is a special kind of dissolve that stops at a defined area size. It merges small faces of the same class to bigger compact faces using the geometric aggregation with the condition that only adjacent faces of the same class are considered.
Split

In addition to the criterion of minimal area size also the extent of the polygon is limited to a minimum distance. That demands for a collapse operator to remove slim, elongated polygons and narrow parts. The collapse algorithm by Haunert & Sester (2008) requires buffer and skeleton operations that are time consuming. Therefore - as faster alternative - a combination of splitting such polygons and merging the resulting parts with a geometric aggregation to other neighbors is used. Instead of shrinking the slim parts to their medial axes we split it at suited points and use the aggregation step to merge the slim polygons with another neighbor.

To find the narrows we use a constrained Delaunay triangulation of the polygon. Each triangle is checked for edges and heights smaller than a threshold. These edges or heights will be used for splitting. (see Figure 3)

![Figure 3. Data before and after a 100 m split operation.](image)

24x-Filter

In CORINE land-cover there is a group of classes which stands for heterogeneous land-uses. The classes 242 and 243 are relevant for Germany. Class 242 (complex cultivation pattern) is used for a mixture of small parcels with different cultures. Class 243 is used for land that is principally occupied by agriculture, with significant areas of natural vegetation.

Heterogeneous classes are not included in the DLM-DE. To form these 24x-classes an operator for detecting heterogeneous land-use is needed. The properties of these classes are that smaller areas with different, mostly agricultural land-use alternate within the minimum area size (actually 25 ha in CLC). For the recognition of class 242 only the agriculture areas (2xx) are relevant. For 243 also forest, semi- and natural areas (3xx, 4xx) and lakes (512) have to be taken into account.
The algorithm calculates some neighborhood statistics for each face. All adjacent faces within a distance of the centroid smaller than a given radius and with an area size smaller than the target size are collected by a deep search in the topological structure. The fraction of the area of the majority class and the summarized fractions of agricultural areas (2xx) and (semi-)natural areas (3xx, 4xx, 512) are calculated. In the case the majority class dominates (>75 %) then the majority class becomes the new class of the polygon. Otherwise there is a check, if it is a heterogeneous area or only a border region of larger homogeneous areas.

For that purpose the length of the borders between the relevant classes is summarized and weighted with the considered area. A heterogeneous area is characterized by a high border length, as there is a high number of alternating areas. To distinguish between 242 and 243 the percentage of (semi-natural) areas has to be significant (>25 %).

**Simplify**

The simplify-operator removes redundant points from the loops. A point is redundant, if the geometric error without using this point is lower than an epsilon and if the topology does not change. Therefore we implemented the algorithm of Douglas & Peucker (1973) with an extension for closed loops and a topology check.

### 3.4 Process chain

In this section the use of the introduced operators and their orchestration in the process chain is shown. The workflow for a target size of 25 ha is as follows:

1. import and clean data and fill holes
2. dissolve faces < 25 ha
3. split faces < 100 m
4. aggregate faces < 1 ha geometrically (base 1.2)
5. reclassify faces with 24x-filter (r=282 m)
6. aggregate faces < 5 ha weighted (base 2)
7. aggregate faces < 25 ha semantically
8. simplify polygons (tolerance 20 m)
9. dissolve all

During the import step (1) semantic and topology is checked. Small topologic errors are resolved by a snapping. Gaps are filled with dummy objects. These objects will be merged to other objects in the later steps.
A first dissolve step (2) merges all faces with an adjacent face of the same CLC class which are smaller than the target size (25 ha). The dissolve is limited to 25 ha to prevent polygons from being too large (e.g. rivers that may extend over the whole partition). This step leads to many very non-compact polygons. To be able to remove them later, the following split-step (3) cuts them at narrow internal parts (smaller than 100 m). Afterwards an aggregation (4) merges all faces smaller than 1 ha (100 m × 100 m) to geometrically fitting neighbors.

The proximity analysis of the 24x-filter step (5) re-classifies agricultural or natural polygons smaller than 25 ha in the 25 ha (corresponding to a radius of 282 m) surrounding as heterogeneous (24x class).

The next step aggregates all polygons to the target size of 25 ha. First we start with a geometric/semantically weighted aggregation (6) to get more compact forms, second only the semantic criterion is used (7) to prevent large semantic changes of large areas.

The simplify step (8) smoothes the polygon outlines by reducing the number of nodes. As geometric error tolerance 20 m (0.2 mm in the map) is used. The finishing dissolve step (9) removes all remaining edges between faces of same class.

### 4 Partitioning method

To create an appropriate partitioning for the derivation of land-use data from the DLM-DE, we divide the minimum bounding rectangle (mbr) of the dataset into a grid of same-sized rectangles. The number of partitions is specified for the x- and y-dimension, respectively. Alternatively, a predefined partitioning scheme can be used, whose geometries have to be imported into the database before processing the topographic data, and are intersected with the dataset mbr to identify the relevant partitions.

For each tile three phases are executed called the partitioning, generalization and composition phase. The second phase is described in detail in the previous section. Below we present the details of the first and the third phase.

The partitioning is defined on the mbr instead of the exact shape of the whole dataset for performance reasons. Empty partitions resulting from a non-rectangular shape of the dataset are identified in the partitioning phase and skipped.
4.1 Partitioning phase

For the current partition to be processed, first its rectangular geometry is enlarged by adding a user-defined width to its borders. In the following the area defined by the original rectangle is called interior, the area defined by the enlargement is called border region and the complement of the enlarged rectangle is called exterior (Figure 4). Then all DLM-objects intersecting the enlarged rectangle are selected and clipped at its border. The resulting dataset is exported from the database to be processed by the CLC-generator.

Please note that each geographic object in a border region also resides in the interior of another partition, which means that these objects are exported and generalized more than once in different partitions. The idea of our partitioning method is that for each area in the interior of the exported partition enough context is provided to take a correct generalization decision using only data exported. Areas residing in the exterior are considered too far away to influence the generalization in the interior. Possibly wrong decisions in the border region are removed during composition.

Fig. 4. Partitioning grid with interior (dark grey), border region (light grey) and exterior (white) of one partition.
4.2 Composition phase

The result of the generalization phase, which is a valid CLC dataset for the current partition, is then imported back into the database. We expect that areas in the border region may be generalized incorrectly because of missing context information in the generalization phase. So the results from the whole border region are thrown away by only selecting areas residing in the interior, clipping these areas at the border of the interior and adding them to the CLC objects of already finished partitions. Thus the final result will not contain any gap, since each area in the border region also resides in the interior of another partition and is accepted when that partition is processed.

However, because CLC objects are clipped, they do not extend across partition borders, which means that adjacent areas from different partitions but assigned the same land-use information are represented by two or more polygons. We identify these situations by executing a spatial join searching for objects from the current partition and from already composed neighboring partitions that have the same CLC class and that have a piece of the partition border in common. Reconciliation is done by aggregating (dissolving) each group of objects, which are in this way associated, into one object (see Figure 5).

4.3 Implementation issues

We have implemented the partitioning and composition phase on top of an Oracle 11g database with Spatial Data Option installed. All operations ac-
cessing data from more than one partition are executed using SQL-statements, so that the database system manages the computation resources and we don’t need to deal with memory limitations by ourselves. To avoid unnecessary but expensive disk fetches and geometrical computations we use Oracle’s built-in spatial index type, which is an implementation of the R*-tree (Beckmann et al., 1990), to access the imported DLM data and generated CLC data. To exchange data with the CLC-generator we use the simple and popular shapefile format by ESRI.

5 Results

5.1 Runtime and memory use of the generalization step

The implemented algorithms are very fast but require a lot of memory. Data and index structures need up to 160 Bytes per point on a 32 bit machine. The run-time of the generalization routines was tested with a 32 bit 2.66 GHz Intel Core 2 processor with a balanced system of RAM, hard disk and processor (windows performance index 5.5). The whole generalization sequence for a 45 km × 45 km dataset takes less than two minutes. The most time expensive parts of the process are the I/O-operations which take more than 75 % of the computing time. We are able to read 100 000 points per second from shape files while building the topology. The time of the writing process depends on the disk cache. In the worst case it is the same as for reading.

5.2 Semantic and geometric correctness

To evaluate the semantic and geometric correctness we did some statistics comparing input, result and a CLC 2006 reference dataset, which was derived from remote sensing data.
Figure 6 shows the input data (DLM-DE), our result and the CLC 2006 of the test area Dresden. The statistics in Figure 7 verifies that our result matches with DLM-DE (75 %) better than the reference dataset (60 %). This is not surprising as for CLC 2006 different data sources were used. Because of the removing of the small faces our generalization result is a bit more similar to CLC 2006 (66 %) than CLC 2006 to the input dataset.

Figure 7. Percentage of area for each CLC class (bars) and percentage of match (A0) and \(\kappa\)-values for the Dresden dataset.
Table 2 shows that our polygons are only a bit smaller, more complex and less compact than the CLC 2006 polygons. The structure index values (diversity, dominance and homogeneity) (Liu et al. 2010) indicate that the structure was preserved during the process.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DLM-DE</th>
<th>Result</th>
<th>CLC 2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygons</td>
<td>91717</td>
<td>1244</td>
<td>876</td>
</tr>
<tr>
<td>Points per Polygon</td>
<td>23</td>
<td>62</td>
<td>75</td>
</tr>
<tr>
<td>Area per Polygon</td>
<td>2.3 ha</td>
<td>167 ha</td>
<td>238 ha</td>
</tr>
<tr>
<td>Perimeter per Polygon</td>
<td>0.6 km</td>
<td>10.1 km</td>
<td>9.0 km</td>
</tr>
<tr>
<td>Avg. Compactness</td>
<td>50 %</td>
<td>29 %</td>
<td>33 %</td>
</tr>
<tr>
<td>Diversity</td>
<td>2.8</td>
<td>2.7</td>
<td>2.6</td>
</tr>
<tr>
<td>Dominance</td>
<td>1.9</td>
<td>1.7</td>
<td>1.9</td>
</tr>
<tr>
<td>Homogeneity</td>
<td>0.60</td>
<td>0.61</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Table 2. Statistic of the test dataset Dresden (45 km × 45 km)

The percentage of the CLC classes is similar in all datasets (Figure 7). There are some significant differences between the DLM-DE and CLC 2006 within the classes 211/234 (arable/grass land) and also between 311/313 (broad-leaved/mixed forest) and 111/112 (continuous/discontinuous urban fabric). We assume that this comes from different interpretations and different underlying data sources. The percentages in our generated dataset are mostly in the middle. The heterogeneous classes 242 and 243 are only marginally included in the input data. Our generalization generates a similar fraction of these classes. However, the automatically generated areas are often not at the same location as in the manually generated reference dataset. We argue though that this is the result of an interpretation process, where different human interpreters would also yield slightly different results.

Input (DLM-DE) and the result match with 75 %. This means that 25 % of the area changes its class during generalization process. This is not an error; it is an unavoidable effect of the generalization. The \(\kappa\)-values 0.5-0.65 which stand for a moderate up to substantial agreement should also not be interpreted as bad results, because it is not a comparison with the real truth, or with a defined valid generalization, respectively.
5.3 Stability of generalization results

To test the influence of the generalization parameters to the result we made some experiments with our test datasets. To get an impression of its influence and to optimize the generalization, we changed each parameter separately in small steps. The result of the changed generalization was then compared with the input data and the CLC reference dataset. Also the statistics (Table 2) was taken into account.

To simulate an update process and its effects on the generalized data, we used two different versions of the DLM-DE (see Figure 8). The land-use of these two datasets differs in nine percent of the area (ground truth). Both datasets were generalized with the same parameters; the land-use of the generalization results differs in 13% of the area. 20% of these differences in generalized data are correct and 20% are false (different classes). The other 60% are false positive – they occur at areas where no differences are in ground truth. 30% of the real changes are missing (false negative) (see Figure 9).

This example shows that changes in the input data produce more and also different changes in the generalized data. The causes of these changes are the classification and the aggregation step. In these generalization operations decisions are made based on thresholds. A small change can switch between the states under or over the threshold and produce a very different result. Because of the local decisions of the generalization algorithms this often leads also to changes in the local environment. Changes of the input data have only an influence in a limited environment.
Fig. 8. Two versions of input DLM-DE (left), their generalization results (right) and the differences between the versions (below). 9% changes of the input data produce 13% differences in the generalized data.

Fig. 9. Overlay of the differences between input and output data.
5.4 Partitioning experiments

To study the impact of the border-width on consistency, we selected the Dresden dataset of 45 km × 45 km. We first generalized the dataset at once using only one partition and no border-regions. We considered the result as a reference, because it cannot contain errors induced by partitioning. Second we generalized the same dataset many times while dividing it into four partitions and varying the border-width from zero up to 3.5 km. Each of the results was compared to the reference by computing a diff-dataset showing all areas that are assigned different CLC classes. The results are shown in Figure 10 by plotting the sum of differing areas over the width of the partition borders.

These results show the need of adding redundancy to the partitions. While we have a total of almost 2000 ha differently classified area when using no border regions, this error decreases very fast (please note the logarithmic scale on the vertical axis) with increasing border width. At 2.5 km the diff contains only 0.16 ha (8·10⁻⁷ of the overall generalized area), and at 3.5 km the result matches the reference completely. The running time of the pure generalization rises from 115 to 157 seconds only.

![Diagram](image)

Fig. 10. Total error and running time of the generalization plotted against the border width.
We could prove the capability of our approach to handle large datasets in another experiment, in which we also investigated the connection between partition size and computation time. We generalized the DLM-DE of Lower Saxony, which contains 1.4 million polygons, many times using a different number of partitions. Given the results of the previous experiment, we selected a constant border width of 2.5 km. The running times of the three phases are shown in Figure 11.

Most noticeable in the experiment is the strong increase of running time for the generalization phase when using large partitions. While it takes only 36 minutes to generalize all partitions of the 5 × 5 grid, using only nine partitions (3 × 3) raises the running time to 90 minutes. Using even larger partitions (e.g. a 2 × 2 grid) is not possible in our test environment due to a lack of free memory available for the CLC-generator. Increasing partition size also has a slightly negative effect on performance in the partitioning phase. Decreasing partition size from 4250 to 1062 km² does not alter the running time of any phase significantly. We conclude that partition sizes between 2000 and 4000 km² are a good choice for our test environment.

![Figure 11](image.jpg)

**Fig. 11.** Total running times of the three phases over all partitions. Number of partitions is given in brackets below the partition size.
Partitioning was tested on an Intel Core 2 Duo (2.53 GHz) machine with 2 GB RAM running Windows 7. The database server was also locally installed on the same computer.

6 Conclusions and Outlook

The whole process was designed separating the generalization from scalability issues by only exchanging data in a common file format. This way the CLC-generator could be developed as a stand-alone program without involving a database system but using efficient geometric processing. The database system is only used for partitioning and composition, where data from multiple partitions have to be accessed. Thus geometric computations that are rather expensive in the database, are restricted to intersection (clipping, spatial selections and joins) and aggregation (reconciling).

We plan to generalize our partitioning concept to a database service that can also be used to solve scalability problems in other localizable computations on large sets of spatial data. We hope this can be done in many practically relevant situations without major changes to the source code of spatial data computations and without major performance overhead caused by partitioning and composition.

Our next project aim is to derive the CORINE land cover change layer from different versions of DLM-DE. The change layer cannot be generated by intersecting CORINE land cover datasets, because the minimum mapping unit of the change layer is only five hectare in contrast to 25 ha for the land cover dataset. The EEA is only interested in real changes and not in so called technical changes (changes that are produced by the generalization). Resulting from our experiments in Section 5.3, we plan to intersect versions of the high resolution data DLM-DE and then to filter and aggregate the detected changes.

References


Anonymous Authors


