DGK Deutsche Geodätische Kommission
bei der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 626

Jan-Henrik Haunert

Aggregation in Map Generalization by Combinatorial Optimization

München 2008

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5038-9

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 276, Hannover 2008

DGK Deutsche Geodätische Kommission
bei der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 626

Aggregation in Map Generalization by Combinatorial Optimization

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Ing. Jan-Henrik Haunert
geboren am 9.5.1978 in Hannover

München 2008

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission bei der C. H. Beck'schen Verlagsbuchhandlung München

ISSN 0065-5325 | ISBN 978-3-7696-5038-9

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 276, Hannover 2008

Adresse der Deutschen Geodätischen Kommission:

Deutsche Geodätische Kommission

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100

e-mail hornik@dgfi.badw.de ● http://www.dgk.badw.de

Prüfungskommission

Vorsitzender:	Univ.-Prof. Dr.-Ing. Christian Heipke
Referentin:	Univ.Prof. Dr.-Ing. Monika Sester
Korreferenten:	PD Dr.rer.nat. Alexander Wolff
	Univ.Prof. Dr.-Ing. Hansjörg Kutterer

Tag der mündlichen Prüfung: 17.09.2008

ISSN 0065-5325 ISBN 978-3-7696-5038-9

Summary

This thesis presents a new method for aggregating area objects in topographic databases. Aggregation is an important sub-problem of automatic generalization, which in cartography and geoinformatics means to derive representations of higher abstraction and smaller scale from given datasets. The developed method aims to create results of maximum quality and to ensure standards like minimal dimensions that are given with database specifications. Basically, two quality criteria are considered relevant: (1) Large-scale objects are to be represented by small-scale objects of semantically similar classes – this aim is formalized by defining a semantic distance between classes. (2) Areas in the smaller scaler need to have a geometrically simple, compact shape. For the first time, this task is consequently formulated as an optimization problem and solved by applying combinatorial optimization techniques. In a simplified form, the problem can be stated as follows:

> *Given a planar map with areas of different classes and a minimal allowed area size θ for the target scale, combine the areas into contiguous regions and define a new class for each region, such that each region has size at least θ and the area covered by objects whose classes change is minimized.*

It will be proven that this simplified problem is already NP-hard. Therefore, it is unlikely that an exact polynomial-time algorithm exists for its solution. This motivates an approach to the problem that is based on mixed-integer linear programming and heuristics. The proposed methods allow semantic class distances and different compactness measures to be considered. These criteria are incorporated into the cost function that is minimized. Furthermore, it is possible to define size thresholds for different classes, in order to preserve objects of high priority.

Due to the high complexity of the problem, the exact solution cannot be obtained for large problem instances. Nevertheless, for small instances, the approach allows the results of heuristics to be compared with optimal solutions. This enables the performance of heuristic methods to be assessed with respect to the quality of the result. The development of the presented heuristics is motivated by two aims: Firstly, large problem instances, for example, national topographic databases, need to be generalized. Secondly, local updates need to be propagated into a previously derived small-scale map without processing the complete dataset again. In addition to heuristics that eliminate certain variables in the mixed-integer programs, a heuristic is proposed that decomposes a problem instance into smaller, independent instances. With this approach both aims are achieved. Furthermore, a generalization workflow is presented that, in addition to the aggregation method, comprises methods for area collapse and line simplification.

In order to prove the applicability of the proposed method in practice, it was applied to a dataset of the official German topographic database ATKIS. A dataset of the ATKIS DLM 50 was processed, which contains similar amounts of detail as the digital topographic map at scale 1:50 000 (DTK 50). The minimal allowed size θ was defined according to existing specifications of the ATKIS DLM 250. A dataset of 22 km $\times$ 22 km with 5537 areas (a map sheet of the DTK 50) was aggregated in 82 minutes. Compared to an existing iterative approach (the so-called region-growing), the new method resulted in 20% less class change and 2% less cost for non-compact shapes.

The developed method has several advantages, in particular, it offers results of high semantic accuracy. Furthermore, the proposed optimization method is deterministic and allows (hard) constraints to be handled. These are advantages compared to existing optimization methods for map generalization, which are often based on iterative meta-heuristics like simulated annealing. How to adapt the developed approach to other generalization tasks – in particular to building simplification – is discussed as a question for future research.

Keywords: area aggregation, map generalization, mixed-integer programming

Zusammenfassung

Diese Arbeit präsentiert ein neues Verfahren zur Aggregation flächenhafter Objekte in topographischen Datenbanken. Die Aggregation ist ein wichtiges Teilproblem der automatischen Generalisierung, die in der Kartographie und Geoinformatik die Generierung von Repräsentationen höherer Abstraktion und kleineren Maßstabs aus gegebenen Datenbeständen umfasst. Das entwickelte Verfahren zielt darauf ab, Ergebnisse maximaler Qualität zu generieren und zugleich Standards wie minimale Flächengrößen zu garantieren, die durch Datenbankspezifikationen vorgegeben sind. Im Wesentlichen werden zwei Qualitätskriterien als relevant angesehen: (1) Objekte im Ausgangsmaßstab sollen im kleineren Maßstab durch Objekte semantisch ähnlicher Klassen repräsentiert werden; dieses Ziel wird durch die Definition einer semantischen Distanz zwischen Klassen formalisiert. (2) Flächen im kleineren Maßstab sollen geometrisch einfache, kompakte Formen haben. Erstmals wird diese Aufgabe konsequent als Optimierungsproblem formuliert und durch Verfahren der kombinatorischen Optimierung gelöst. Vereinfacht lässt sich das behandelte Problem folgendermaßen definieren:

> *Gegeben sei eine planare Karte mit Flächen unterschiedlicher Klassen sowie eine minimal zulässige Flächengröße θ für den Zielmaßstab. Fasse die Flächen in zusammenhängende Regionen zusammen und definiere für jede Region eine neue Klasse, so dass jede Region mindestens die Größe θ hat und die Gesamtfläche, deren Klasse sich ändert, minimal ist.*

Es wird bewiesen, dass bereits dieses vereinfachte Problem NP-schwer ist und somit kaum Hoffnung besteht, einen exakten Algorithmus mit polynomieller Laufzeit zur Lösung zu finden. Dieses motiviert einen exakten Lösungsansatz durch gemischt-ganzzahlige lineare Programmierung sowie verschiedene heuristische Verfahren. Dabei werden auch die semantischen Klassendistanzen und verschiedene Kompaktheitsmaße berücksichtigt. Außerdem können unterschiedliche Flächenschwellwerte für unterschiedliche Klassen definiert werden, um Objekte hoher Bedeutung zu erhalten.

Die exakte Lösung des Problems ist aufgrund der hohen Komplexität nicht für große Instanzen zu realisieren. Dennoch ergeben sich neue Möglichkeiten, die Leistungsfähigkeit heuristischer Verfahren an exakten Lösungen für kleine Instanzen zu messen. Die Entwicklung der vorgestellten Heuristiken wird durch zwei Ziele motiviert. Erstens sollen auch große Probleminstanzen, etwa nationale topographische Datenbanken, verarbeitet werden können. Zweitens sollen lokal begrenzte Aktualisierungen im Ausgangsmaßstab in den Zielmaßstab übertragen werden können, ohne dass eine erneute Prozessierung des gesamten Datenbestandes nötig wäre. Neben Heuristiken, welche die Anzahl der Variablen in den Probleminstanzen verringern, wird eine Heuristik eingeführt, die es ermöglicht eine Probleminstanz in unabhängige Teilinstanzen zu zerlegen. Dadurch ist es gelungen beide Ziele zu erreichen. Weiterhin wurde ein Ablaufplan für die Generalisierung entwickelt, der neben der Aggregation auch den Geometrietypwechsel und die Liniengeneralisierung umfasst.

Um die Anwendbarkeit des Verfahrens in der Praxis nachzuweisen, wurde es am Beispiel des amtlichen topographisch-kartographischen Informationssystems (ATKIS) getestet. Als Eingabe diente ein Datensatz des ATKIS DLM 50, dessen Inhalt dem einer digitalen topographischen Karte im Maßstab 1:50 000 (DTK 50) entspricht. Die minimal zulässigen Flächen θ wurden entsprechend bestehender Spezifikationen des ATKIS DLM 250 definiert. Ein Datensatz der Größe 22 km $\times$ 22 km mit 5537 Flächen (ein Blatt der DTK 50) wurde in 82 Minuten aggregiert. Im Vergleich zu einem weit verbreiteten iterativen Algorithmus (dem sogenannten *region-growing*) ergab sich eine Reduktion der Klassenänderung um 20%. Die Kosten für nicht-kompakte Regionen wurden um 2% reduziert.

Das vorgestellte Verfahren hat mehrere Vorteile, insbesondere liefert es Ergebnisse hoher semantischer Genauigkeit. Darüber hinaus ist der gewählte Optimierungsansatz deterministisch und ermöglicht die Handhabung (harter) Nebenbedingungen. Dieses sind Vorteile im Vergleich zu bestehenden Optimierungsansätzen durch iterative Metaheuristiken wie Simulated Annealing. Die gewählte Herangehensweise auf andere Generalisierungsprobleme – insbesondere die Gebäudevereinfachung – zu übertragen, wird als Aufgabe für zukünftige Forschung diskutiert.

Stichworte: Flächenaggregation, Generalisierung, gemischt-ganzzahlige Programmierung

Contents

Chapter 1

Introduction

1.1 Motivation

Topographic maps are widely used for different applications, for example, land use planning, construction, earth sciences, and recreation. Normally national mapping agencies (NMAs) provide them in different scales for the whole extent of a country. Due to the increasing use of computers during the last 40 years, analog maps are being increasingly replaced by digital maps and digital landscape models (DLMs). Automating the production of these digital representations is a primary goal of cartographic research. Though most parts of the Earth can be considered mapped, this task has not become less important: mainly due to human interactions, the landscape is permanently changing. Therefore, keeping topographic databases up-to-date is also an important task. A high degree of automation can be achieved through *map generalization*: instead of conducting redundant field surveys for maps and DLMs of different scales, representations of a reduced scale are derived from a single detailed source. This thesis contributes to the general objective of automating the generalization process.

Generalization must not be confused with pure data reduction or data compression. Instead, it aims at more abstract representations of the same physical entities. In order to accomplish this task, generalization needs to emphasize important characteristics and exclude unimportant features; programming computers to solve this problem has proved to be difficult.

A perfect generalization method would create, without any human interaction, datasets that could be directly delivered to users. Though many approaches to map generalization have found their way into applications, such perfection has not been reached. In fact it is questionable whether this aim is realistic. Generally, good experiences can support confidence in a certain generalization method, but the variability of maps is so high that we cannot expect good results for all potential situations while serving all applications. At least a visual inspection of the result is needed for quality assessment before delivering the produced maps; often manual post-processing is required. In order to minimize the manual editing and to support human controllers in the quality assessment, a generalization method should:

- guarantee results that satisfy given standards (for example, database specifications),
- offer results that are as good as possible with respect to clearly defined quality criteria, and
- include some kind of self-assessment concerning these quality criteria.

The first two aims can be achieved by finding the optimal solution subject to given constraints. This motivates an approach to map generalization by optimization. Exact optimization approaches are needed to assess the performance of heuristic methods in terms of quality. This can be measured according to the applied optimization objective.

The quality criteria or optimization objectives that are applied in existing approaches are mainly based on geometric measures. However, generalization also implies changes of semantics: in maps or DLMs of different scales, the same physical entity can be represented by objects of different classes. This especially becomes relevant in *aggregation*, which means that objects of potentially different classes are replaced by a single composite

object. To achieve that semantically similar objects are aggregated, measures of semantic accuracy need to be defined and applied in an optimization approach.

The success of map generalization strongly depends on the capability and flexibility of the applied optimization methods. Since map generalization includes discrete problems like aggregation, combinatorial optimization techniques are needed. As a very flexible deterministic and exact approach, mathematical programming has obtained attention in geographic information science. Shirabe (2003, 2005a) has concentrated on the integration of mathematical programming techniques in geographic information systems. However, this approach has seldom been addressed in the map generalization literature. This thesis presents a new optimization approach to map generalization based on this technique.

1.2 Generalization of land cover data

Figure 1.1 shows a sample of areas from the German topographic database ATKIS DLM 50, which define a *planar subdivision*, that is, a partition of the plane into areas such that different areas do not overlap and there are no gaps in between. Each area belongs to a single land cover class. This kind of representation is very commonly used to represent information about land cover or land use in a DLM. The ATKIS DLM 50 contains a similar amount of detail as an analog map of scale 1:50 000 (AdV, 2003).

Fig. 1.1: A sample of areas from the German topographic database ATKIS DLM 50 displayed at scale 1:50 000. The legend defines the encoding of land cover classes by colors, which is used throughout this thesis. This particular clipping only contains a subset of classes. For the scale 1:250 000, the minimal allowed size for settlement, forest, and farmland areas is 40 ha, which is given with the square above the legend.

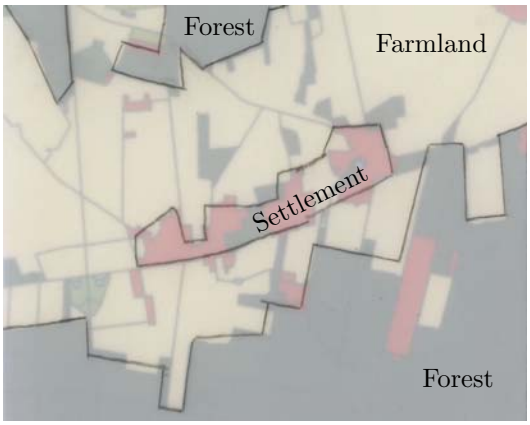


Fig. 1.2: A manually generalized map, sketched by a cartographer with a pencil on a transparency on top of the sample from Fig. 1.1; target scale of generalization 1:250 000. Courtesy of Dieter Heidorn, institute of cartography and geoinformatics, Leibniz Universität Hannover.

The generalization of land cover data needs to be done according to multiple criteria. In order to provide users with standardized products, we wish to derive a DLM of reduced scale that satisfies given requirements, for example, areas must not be smaller than a certain size. Figure 1.2 shows a manually generalized map for the sample in Fig. 1.1. An experienced cartographer was asked to derive a map at scale 1:250 000. In addition to the sample in Fig. 1.1, he was given a set of areas that exactly achieved the minimally required size, to visually compare the required area sizes (e.g., the square of 40 ha). He was asked to create contiguous areas that achieve this size, but he was not restricted in how to do this, that is, he was only asked to use his own experience. We observe that the main strategy for the defined aim was to aggregate areas of the same classes. In some cases, areas of other classes were included in a region for the target scale, for example, to ensure the contiguity of a region. Nevertheless, the class changes were kept small. The map is not cluttered with complex shapes, original boundaries were simplified, and in some cases new boundaries were introduced. The emerging problem is how to automate this process of combining areas in order to obtain such well-generalized maps and to ensure contiguous regions of sufficient size.

In this thesis, we will elaborate on the aggregation problem in map generalization in detail. Our basic approach is to consider areas in the input as minimal mapping units; each area in the target scale is defined as the union of a subset of these units. However, as the generalization task allows more variability, we will also consider other possibilities of manipulation, that is, collapse of areas and area parts to lines as well as line simplification. Generally, we need to find strategies to attack the generalization problem in a comprehensive way. Multi-agent systems are often used for this purpose (Barrault *et al.*, 2001), especially to handle different generalization methods together, which is often referred to as orchestration. Nevertheless, these multi-agent systems rely on basic algorithms for different sub-problems. Therefore, a good solution of the aggregation problem is an important achievement for generalization.

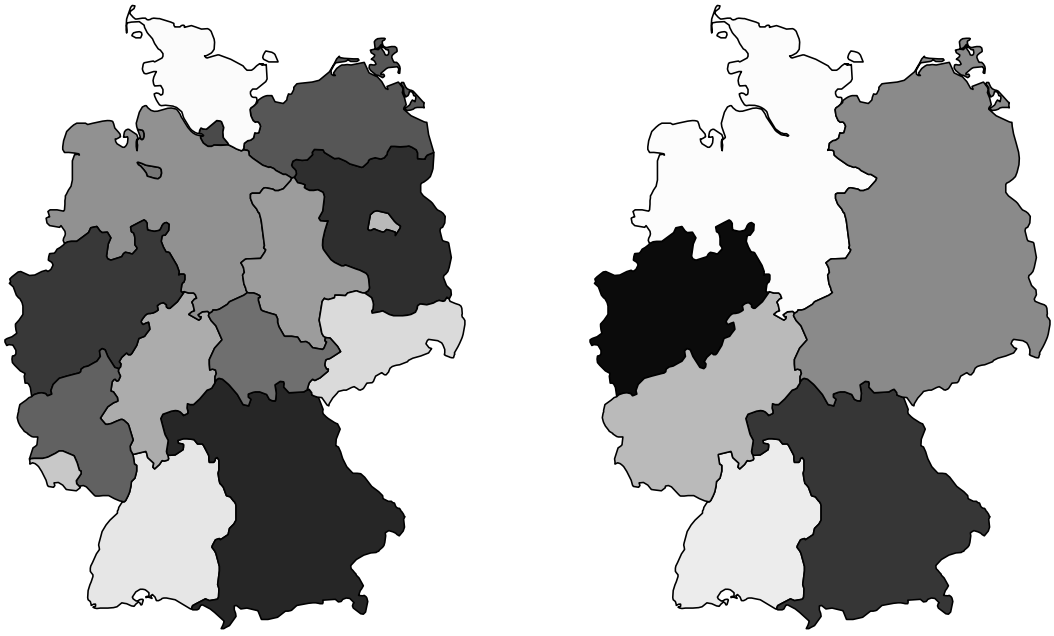
Since the aggregation of areas is primarily based on their classes, we will treat the definition and optimization of a class-based measure for semantic quality as one of our main tasks. Furthermore, we consider compactness of shapes to avoid cluttered maps. The aim for compactness complies with well-known principles of human perception, that is, the human brain preferably groups objects that are close to each other (Wertheimer, 1938).

1.3 Spatial allocation problems

Though this thesis primarily aims at improvements in map generalization, its contributions can be seen in a broader context. The aggregation of areas in a planar subdivision is closely related to *spatial allocation problems*, which Shirabe (2003) defines as follows:

> "Given a set of spatial units, such as land parcels and grid cells, how to allocate subsets of it to activities of interest while satisfying certain criteria?"

The definition applies to spatial problems in the context of operations research and management science, which includes districting problems. Usually, the "activities of interest" are political or economical in nature. Figure 1.3 shows a typical example:



(a) the 16 federal states of Germany (b) an aggregation result with 6 federal states

Fig. 1.3: A typical districting problem. Article 29 of the German constitution (Grundgesetz) regulates a potential reorganization of the German federal territory. In order to reduce expenses for administration, the aggregation of federal states has been discussed for several years. According to the web site of the German TV news "Tagesschau" a federal state needs a population between 5 million and 18 million to operate economically (ARD, 2006). The aggregation result in (b) is feasible according to that constraint. Among all feasible aggregations with 6 states it has a minimum boundary length, which can be seen as a measure for region compactness.

Several spatial properties are relevant for most spatial allocation problems. This includes size, compactness, and contiguity of shapes. Therefore, approaches to spatial allocation can often be adapted to diverse problems. The problem instance in Fig. 1.3 was solved with a mixed-integer program (MIP) for map generalization that we present in Sect. 7.4. Related problems are land acquisition (Wright *et al.*, 1983), sales territory alignment (Zoltners & Sinha, 1983), the definition of school districts (Caro *et al.*, 2004), defining regions for nature conservation (Fischer & Church, 2003), and electoral districting (Schröder, 2001). Generally meta-heuristics (Aerts & Heuvelink, 2002; Tavares-Pereira *et al.*, 2007) and mathematical programming techniques (Hess & Samuels, 1971) have been applied to spatial allocation problems. The new generalization method by mixed-integer programming presented in this thesis was influenced by formulations of Zoltners & Sinha (1983), Williams (2002), and Shirabe (2005b). We will discuss the contributions of the authors when adapting components of their methods for our own purposes. A more detailed review of approaches to spatial allocation problems is given by Shirabe (2003).

1.4 Goal of this thesis

This thesis aims at high-quality aggregation for the creation of well-generalized digital landscape models. The goal of this thesis is to answer the following research questions:

- How can we measure quality, in particular semantic accuracy, in generalization?
- According to these criteria, how can we approach generalization in order to create datasets of high quality, that comply with database specifications?
- Which techniques for combinatorial optimization and spatial allocation can we apply for aggregation, what are their benefits and limitations, and how can we improve on them?
- How competitive are these techniques under realistic conditions, that is, if large datasets need to be processed and derived data needs to be frequently updated?

A number of novel contributions to map generalization are presented in the thesis, which can be seen in the context of this general research goal. In addition to giving answers to the above general questions, we solve the following concrete problems.

- We elaborate the definition of quality in the context of generalization, which leads to a precise definition of the aggregation problem. We prove that this problem is NP-hard, meaning that there is little hope that an efficient exact algorithm exists.
- We develop a mixed-integer programming approach to aggregation. Compared to other combinatorial optimization approaches to map generalization, in particular randomized meta-heuristics like simulated annealing, it has four main benefits: (1) hard constraints can easily be defined, (2) it is deterministic and so results are reproducible, (3) it offers exact, that is, globally optimal solutions for small samples, which allows the performance of heuristics to be assessed, and (4) the exact solution does not require tuning of parameters that are not inherent to the problem.
- To process very large datasets and to efficiently process updates in the newly developed optimization approach, we propose a heuristic that decomposes a dataset into independent problem instances. When processing large datasets, these instances can be solved in parallel. When updating the source map, only those problem instances that were affected by the update need to be solved again to derive the updated generalized map.
- An existing approach to area aggregation in map generalization is based on iterative merging. We present a new, incremental algorithm that guarantees to offer the same result as the existing iterative approach of van Oosterom (1995). This method can be used for efficient updating if a change occurred in the source dataset, since in this case only a limited set of objects has to be inspected.

In the general context of spatial allocation problems, this thesis introduces the following novelties:

- The number of variables and constraints in existing mixed-integer programs for districting problems is quadratic in the number of minimal mapping units if the number of districts is unknown. The newly developed flow-based MIP gets by with a linear number of variables and constraints. It allows optimally compact and contiguous districts to be found that satisfy size constraints.
- Compactness of regions is often achieved by minimizing distances to a central facility, for example, customers living in a sales district need fast access to the central store. The flow-based MIP allows to apply a new version of this compactness measure: it minimizes travel distances along paths that are constrained to the interior of a region. For example, if an enclosure for lions is planned, they probably need fast access to their feeding trough, but they have to remain inside the enclosure.

1.5 Outline

The thesis is structured as follows. We first review basics of map generalization (Chapter 2). We put emphasis on standardization and quality assessment (Sect. 2.3), existing optimization approaches (Sect. 2.5), and aggregation (Sect. 2.7), which includes the review of an existing, iterative approach to the aggregation of areas in a planar subdivision. Then we review the basics of combinatorial optimization with a focus on mathematical programming (Chapter 3). Chapter 4 presents a new incremental version of the iterative aggregation algorithm.

The remaining part of the thesis concentrates on a new generalization method for areas that form a planar subdivision and belong to different classes. We introduce basic ideas and define relevant elements of quality (Chapter 5). This motivates a workflow that comprises collapse of areas to lines, aggregation, and line simplification. For this workflow, we developed a collapse procedure and a line simplification method, which extends an existing optimization approach. We present both methods in Chapter 6.

Then we focus on the most important contribution of this thesis, that is, the optimization approach to area aggregation (Chapter 7). We define the aggregation problem formally and present a proof that aggregating areas with minimum class change is NP-hard, if we forbid aggregates of size smaller than a defined threshold. Due to the NP-hardness of the problem it is unlikely that an efficient exact algorithm exists; therefore, we focus on mixed-integer programming and heuristics, which can be used to process large datasets and to handle updates.

We discuss results of the developed generalization approaches in Chapter 8, which also explains how the defined measures can be used for quality assessment. Chapter 9 concludes the thesis and gives recommendations for future research.

Chapter 2

Map Generalization

Map generalization is the process of deriving a map at a reduced scale from a given map. Since display devices such as sheets of paper or computer screens have limited resolutions and sizes, only selected parts of the original information can be shown. Also reading a map with too many details is difficult for users who generally intend to draw conclusions for their personal decisions: For example, the best route from Berlin to Paris can be found much faster on a map of important highways than on a map of the entire road network. Thus, the problem is to distinguish unimportant details from important features, which also depends on the purpose of the map. This work focuses on digital topographic databases, that is, digital landscape models, whose purpose is not only to directly aid users in their applications, but also to define a foundation for diverse kinds of maps.

In order to explain the context of this work, this chapter summarizes the basics of map generalization. We review common classifications of generalization sub-tasks into so-called generalization operators (Sect. 2.1) and distinguish the problems of generating a database of reduced detail and a graphical map (Sect. 2.2). Section 2.3 discusses the influence of subjectivity in map generalization and approaches to define objective requirements and quality measures. Section 2.4 explains the concept of constraints. This is an important prerequisite for optimization approaches, which are reviewed in Sect. 2.5. Section 2.6 reviews approaches for organizing the generalization workflow and Sect. 2.7 discusses approaches to aggregation, as solving this problem is the primary concern of this thesis. Finally, we discuss methods for updating generalized data (Sect. 2.8).

Several textbooks exist, giving a more comprehensive overview of all important aspects of map generalization (Buttenfield & McMaster, 1991; Müller *et al.*, 1995; Mackaness *et al.*, 2007). Weibel (1997) and Li (2007) focus on algorithms used in automation.

2.1 Generalization operators

Eliminating map objects can be an appropriate solution to reduce the clutter on a map. The whole generalization problem, however, appears to be more complex, as maps of smaller scale usually allow for alternative representations of the same physical entities. For example, a building that is too small to be displayed in the smaller target scale may alternatively be enlarged or aggregated with others. Such basic generalization processes are called *generalization operators*, which are classified by different authors. Rieger & Coulsen (1993) reveal discrepancies among different terminologies. Figure 2.1 shows the taxonomy after Hake *et al.* (1994).

Simplification reduces the amount of detail, for example, by selecting a subset of line vertices. *Enlargement* of objects can be performed to satisfy graphical limitations. This can result in graphical conflicts between different objects; in this case we need to move objects, which is termed *displacement*. *Aggregation* means to replace a group of objects by a composite object. *Selection* of important objects is a very common approach, unimportant objects are eliminated. In the taxonomy of Hake, *classification*, *typification*, and *symbolization* are subsumed by one generalization operator. Classification means, for example, that objects of type 'post office' and 'school' are replaced by objects of the more general type 'public building'. Typification is often performed to represent the patterns of objects, for example, five buildings in a row are replaced by a set of four buildings that have a similar characteristic alignment. Symbolization means, for example, to replace an object that has

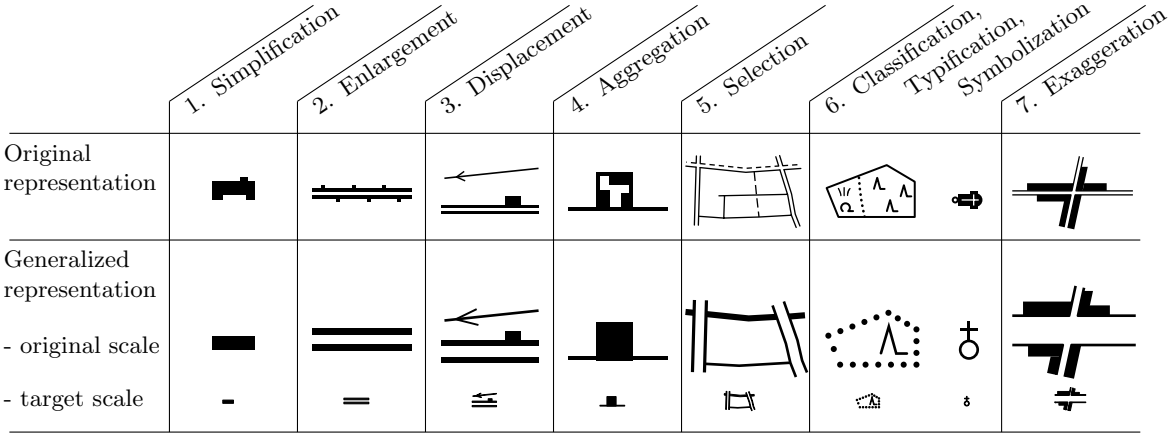


Fig. 2.1: Generalization operators after Hake *et al.* (1994).

a two-dimensional extent with a point symbol. It also comprises the collapse of areas to lines. *Exaggeration* is applied to emphasize important or characteristic features.

Researchers have successfully automated generalization operators for certain types of objects. For example, Staufenbiel (1973), Kada & Luo (2006) and Sester (2005) have developed generalization algorithms for the simplification of building ground plans. Thomson & Richardson (1995), van Kreveld & Peschier (1998) and Zhang (2005) present methods for the selection of roads. This thesis focuses on aggregation, taking multiple classes into account. To automate the generalization process it does not suffice to propose algorithms for single operators or single types of objects. We need to develop an appropriate workflow or an overall strategy to find the right algorithm for a given situation. Existing approaches to handle multiple operators are reviewed in Sect. 2.6.

In Fig. 2.1 we investigated the effects of generalization operators on samples with cartographic symbols, but map generalization is not always applied for visualization. Though the classification of generalization operators is a general concept, we need to distinguish different generalization tasks.

2.2 Model generalization and cartographic generalization

Map scale is the ratio of a distance on the map to the corresponding distance on the Earth. However, for a digital representation of the world, this definition is inapplicable, unless the data is printed or rendered on a screen. On the other hand, similarly to maps, digital landscape models are made for different purposes that require different levels of abstraction. The level of detail of a DLM often corresponds to the content of a map with a certain scale. Figure 2.2 shows the same scene in two different DLMs from the German ATKIS project. Both samples are displayed in the same extent, using the same primitive symbols. We observe that the right sample includes less details, yet preserves the most characteristic structures. In fact, the Basis DLM (Fig. 2.2(a)) is used to derive the German topographic maps at scales 1:10 000 and 1:25 000; the DLM 250 (Fig. 2.2(b)) serves as basis for the map at scale 1:250 000 (Grünreich, 2003). The ATKIS project also defines two more datasets, the DLM 50 (1:50 000) and the DLM 1000 (1:1 000 000). National mapping agencies of many countries provide similar datasets on nation-wide extents.

Though digital landscape models are related to map products, there are important differences: objects in DLMs are kept without cartographic symbols, in order to allow for multiple applications. Some applications do not require a cartographic visualization of the data, for example statistical analyses. Also a DLM can be used to derive different maps that require different symbols. Due to this consideration, generalization is often done in two steps (Brassel & Weibel, 1988):

1. *Model generalization*, also known as *database generalization* or *statistical generalization*, transforms a DLM into a less detailed DLM.

2. *Cartographic generalization* transforms a DLM into a digital cartographic model (DCM). A DCM is just the digital equivalent of an analog paper map, meaning it has complete symbols and text labels at defined positions.

Both model generalization and cartographic generalization are complex problems. The first is primarily concerned with data abstraction, the second focuses more on graphical effectiveness. This distinction, however, is not crisp: digital landscape models are mostly designed as basis for map products, so their content can be defined with regard to graphical limitations. Often this definition is given with the database specifications.

The generalization operators enlargement and displacement are primarily important for cartographic generalization, as they can be used to cope with graphical limitations. However, if a minimal allowed object size or a minimal allowed distance between objects is given with database specifications, they could also be used for database generalization.

(a) ATKIS Basis DLM (b) ATKIS DLM 250

Fig. 2.2: The same scene in topographic databases of different resolutions.

2.3 Subjectivity, quality, and specifications

It is generally acknowledged that the map generalization problem allows multiple solutions: due to subjective preferences, two cartographers will seldom come up with the same results (Spiess, 1983). This subjectivity leads to inhomogeneous maps, for example, if multiple cartographers work on the same map sheet. In fact, in his early work on line smoothing, Perkal (1966) focuses on the definition of an objective (manual) method without commenting on the possible use of computers for its accomplishment. Töpfer & Pillewizer (1966) propose empirically found rules for the number of objects that are to be selected for a reduced scale. The authors comment on the usefulness of this criterion for automation, but also aim at more objectivity in the manual processing.

The problem of how to measure the quality of a generalized dataset is still a major research issue. Usually, the quality of map generalization is assessed by comparison of the input and the output map. For example, Bard (2004) suggests measures to characterize the data before and after generalization and to assess the differences between both datasets according to these measures. Cheng & Li (2006) compare the aggregation results of a simple iterative algorithm. They consider the amount of class changes as an important quality measure. In order to measure shape similarities, Frank & Ester (2006) and Podolskaya *et al.* (2007) compare the turning functions of corresponding shapes in different datasets.

Often the problem of subjectivity and inhomogeneity is approached by standardization. National mapping agencies of many countries have developed database specifications that rather clearly define the contents of digital land scape models at different scales. Table 2.1 summarizes the definitions of two classes of the ATKIS Basis DLM, namely grassland and river. The complete specifications in German language are accessible on the

Class	Grassland	River
Superclass	Vegetation	Water
Definition	"Area of grass or lawn that is regularly grazed or mowed" (translated from German)	"Natural stream of water" (translated from German)
Selection criterion	area $\geq$ 1 ha	permanently running OR length $\geq$ 500 m
Object type	area	IF width $\geq$ 12 m THEN area ELSE line

Table 2.1: Specifications of classes grassland and river in ATKIS Basis DLM.

Germany ATKIS (AdV, 2003)		Canada National Topographic Database (Natural Resources Canada, 1996)		Australia National Topographic Database (Geoscience Australia, 2006)	
Wood, Forest		Wooded Area		Forest Or Shrub	
"Area covered by forest plants (forest trees and forest shrubs)" (translated from German)		"An area of at least 35% covered by trees or shrubs having a minimum height of 2 m"		"An area of land with woody vegetation greater than 10% foliage cover (includes trees and shrubs)"	
scale	selection criterion	scale	selection criterion	scale	selection criterion
1:25k	area $\geq$ 0.1 ha			1:25k	area $\geq$ 0.25 ha
1:50k	area $\geq$ 1 ha	1:50k	area $\geq$ 1 ha AND width $\geq$ 50 m		
				1:100k	area $\geq$ 4 ha
1:250k	area $\geq$ 40 ha	1:250k	area $\geq$ 25 ha AND width $\geq$ 250 m	1:250k	area $\geq$ 25 ha
1:1000k	area $\geq$ 500 ha				

Table 2.2: Definitions and selection criteria for forest areas in three different national databases. The Canadian specifications use the term "Guaranteed size". In the Australian specifications this is called "Minimum size for inclusion".

Internet (AdV, 2003). Similar specifications can be found for national topographic databases of other countries, for example Canada (Natural Resources Canada, 1996) or Australia (Geoscience Australia, 2006). In the US, a huge project called "The National Map" is currently being realized (Kelmelis, 2003). Afflerbach *et al.* (2004) report on differences in specifications of topographic databases in four different European countries.

Database specifications define which objects are to be included and how these are to be represented. For example, forest areas in the ATKIS Basis DLM must not be smaller than 0.1 ha. Naturally, the minimal allowed size is larger for a dataset of smaller scale. Table 2.2 shows the selection criteria for forest in the German, Australian, and Canadian datasets. The term "Guaranteed size", which is used in the Canadian specifications, unmistakably states that the defined thresholds must not be violated in any case. In addition to requirements for attributes and geometries of single objects, the specifications define requirements for relations between multiple objects. For example, different areas must not overlap. Adhering to the database specifications ensures logical consistency, which is an important element of spatial data quality (Kainz, 1995).

Though database specifications limit the freedom of action in the map generalization process, they do not suffice to solve the generalization problem, as there remains ambiguity in the possible actions for their satisfaction. For instance, a forest of 0.9 ha in the ATKIS Basis DLM must not be kept without change when applying a generalization procedure to meet the specifications of the ATKIS DLM 50, as it does not meet the required minimal size of 1 ha. However, the specifications do not define the action that is triggered if this requirement is violated. In the example, we could enlarge the forest to achieve the required size or we could eliminate it, if there are no other requirements that forbid this solution. Generally, this choice is left to the cartographer.

The requirements that are given by database specifications can be regarded as a certain type of *constraints*. The next section explains this common concept of map generalization.

2.4 Map generalization based on hard and soft constraints

Map generalization is often based on rules, which define certain actions that are performed in given situations (Shea, 1991; van Smaalen, 1996). Nickerson & Freeman (1986) specify an aggregation rule as follows:

> *"If a lake or a pond would have an area less than the minimum polygon area, do not include it, unless there are five or more small lakes within a radius of ten mm, in which case replace these small lakes with one small lake slightly larger than the minimum polygon area."*

Though this rule may be useful in practice, Beard (1991) reveals its inflexibility by showing illustrative examples. Generally, a huge number of rules will be needed to consider all possible situations. Because of this, Beard suggests the definition of *constraints*, which, in contrast to rules, are not bound to particular actions. The generalization task is to satisfy all constraints, but this can be achieved with any sequence of the available generalization operators. Weibel & Dutton (1998) distinguish five different types of constraints:

- *Graphical constraints* specify minimum size and proximity properties, which are due to graphic limits.
- *Topological constraints* define topological relationships, for example, network connectivity.
- *Structural constraints* express geometric characteristics and semantics of single features as well as structures of multiple features, for example, alignments and clusters.
- *Gestalt constraints* reflect human aesthetics and visual balance. They correspond to principles of Gestalt theory, which was developed by Wertheimer (1938).
- *Process constraints* define constraints on the generalization process itself, for example, certain operators need to be applied in a defined order. Though the definition of process constraints violates the independence of constraints from generalization operators, they are often needed for an efficient workflow.

Weibel & Dutton (1998) point out that the distinction of structural constraints and Gestalt constraints is rather vague. In Sect. 2.7 we review approaches to aggregation in map generalization that use structure or pattern recognition techniques, also considering principles of Gestalt theory.

According to Weibel & Dutton (1998), constraints are often conflicting, thus can only be satisfied to a certain degree. However, we should be aware of the fact that some constraints do not allow any compromise. This especially concerns constraints that are given by database specifications. Such constraints can only be satisfied or unsatisfied; there is no state in between. As a consequence we need the following distinction:

- *Hard constraints* must be satisfied in any case.
- *Soft constraints* allow for a trade-off if there are conflicts with other soft constraints.

Relaxing a hard constraint means to remove it or to replace it by a soft constraint.

Let's consider a simple example. We are given a set of point symbols in the plane and the following set of constraints:

(C1) The distance between two symbols must not be smaller than a given threshold.
(C2) The location of a symbol must not change.
(C3) A symbol must not be eliminated.

Obviously, if we define all three constraints as 'hard', we can end up with an over-constrained problem, that is, the problem can become unsolvable. Therefore, we should be careful in defining hard constraints. Alternatively, we can define a subset of constraints as 'soft'. For a soft constraint we need to define a cost measure that expresses its degree of satisfaction. For instance, we can relax constraint (C2) and define a cost for point movements. In this case our example becomes a typical displacement problem. We can alternatively relax constraint (C3) and charge a unit cost for each eliminated point; in this case we have a typical selection problem. If we relax both constraints, we have more freedom of action. In this case we search for a solution that defines a good trade-off: We can move a point, but if the cost for its movement exceeds a certain value, then it may be better to eliminate it.

Generally, we are interested in the solution that satisfies all hard constraints and defines the best trade-off between soft constraints, that is, the feasible solution with minimum cost. This naturally leads to the approach by optimization.

2.5 Map generalization by optimization

When discussing optimization approaches to map generalization, we can distinguish discrete and continuous optimization problems, that is, problems where the set of feasible solutions is discrete or not.

2.5.1 Continuous optimization

Problems that have been approached by continuous optimization are line smoothing (Burghardt, 2005) and displacement (Bobrich, 1996; Bader, 2001). The latter problem requires proximity conflicts between different objects to be resolved by small geometrical movements and distortions. Harrie (1999) and Sester (2005) propose to solve this problem by least squares adjustment. In this approach soft constraints are defined by the equation:

$$A \cdot x = l + v\,, \tag{2.1}$$

where x denotes the real vector of unknown parameters (the point movements) and v denotes the real vector of unknown residuals, that is, the degree of constraint satisfaction. Both, A (referred to as the design matrix) and l (the vector of observations) need to be specified in advance to define the constraints. The constraints are perfectly satisfied if $v = 0$. As this is generally not possible for all constraints, the function $v^T \cdot P \cdot v$ is minimized, where P defines the weights between different constraints. If there are non-linear constraints, these are usually replaced by their linear approximations. Sarjakoski & Kilpeläinen (1999) and Harrie & Sarjakoski (2002) show how to solve the problem for large datasets, also considering other generalization operators. Applying the same adjustment technique, Koch & Heipke (2005) and Koch (2007) additionally show how to cope with hard inequality constraints that are needed to ensure consistency between DLMs and digital terrain models. Related problems are discussed in the generalization domain, for example, a river must not run uphill (Gaffuri, 2007). Least squares adjustment allows different generalization operators to be handled, yet the existing generalization methods that are based on this technique do not take the discrete nature of map generalization into account. Usually, continuous variables are used to model a problem. These are not suited, for example, to represent whether a vertex of an original line is selected for its simplification. In their system, Sarjakoski & Kilpeläinen (1999) define a constraint that attempts to pull an unwanted vertex onto the line connecting its predecessor and successor. This is a smart workaround to also allow for line simplification, but of course it is not a solution to the discrete problem of vertex selection, which only allows two stages and none in between. Sester (2005) applies adjustment calculus to satisfy constraints in building simplification, but also points out that it does not solve the whole problem: the elimination of details is done in a first step, which is not based on optimization. The handling of hard constraints in optimization approaches is seldom addressed in the map generalization literature. Often constraints are relaxed, as they are conflicting (Harrie & Weibel, 2007). A few exceptions exist in the context of discrete optimization, which is addressed in the next section.

2.5.2 Discrete optimization

Discrete (also known as combinatorial) optimization problems in map generalization are mainly approached by meta-heuristics that iteratively improve the map, such as hill climbing (Galanda, 2003), simulated annealing (Ware & Jones, 1998; Ware *et al.*, 2003; Hardy *et al.*, 2007), genetic algorithms (Weibel *et al.*, 1995), tabu search (Ware *et al.*, 2002), or neural networks, which are applied in self-organizing maps (Allouche & Moulin, 2005; Sester, 2005).

Also in these approaches, constraints are mostly relaxed (Galanda, 2003). Hill climbing and simulated annealing, which has been introduced by Kirkpatrick *et al.* (1983), are briefly explained in Sect. 3.7 as most commonly applied representatives of meta-heuristics. Section 7.7 presents an approach to area aggregation using simulated annealing, in order to allow for a comparison with the newly developed method by mixed-integer programming. Reeves (1993) gives a general introduction to meta-heuristics. To obtain an appropriate performance, problems that appear to be continuous are often discretized and approached by iterative improvement. For example, Mackaness & Purves (2001) define a finite set of possible object shifts that define candidate solutions for cartographic displacement.

There are only a few non-heuristic and exact optimization approaches to combinatorial problems in map generalization that cope with hard constraints. Often the generalization problem is considered too complex and the

algorithms are considered too inefficient. On the other hand, it is often possible to find approximation algorithms or to introduce problem specific heuristics into exact algorithms. Chapter 3 presents the basics of these deterministic combinatorial optimization techniques with a focus on mathematical programming, which includes linear programming, integer programming, and mixed-integer programming. Especially for line simplification combinatorial optimization methods have been successful. An existing approach of de Berg *et al.* (1998) to this problem, which is based on a shortest path formulation, is reviewed and extended in Sect. 6.2. The method copes with (hard) topological constraints that ensure the simplicity of a planar subdivision. Many problems have been approached by combinatorial optimization that are related to map generalization. For example, Cromley (1986) presents an approach to cartographic label placement by integer programming. Nöllenburg & Wolff (2006) use mixed-integer programs to optimally draw metro maps, in which lines are restricted to certain directions. In particular researchers in the field of computational geometry have focused on these approaches to map generalization.

2.6 Organizing the generalization workflow

The existence of methods for different generalization operators raises the question whether the whole generalization problem can be modeled and solved in a comprehensive way. In fact the constraint-based modeling of map generalization is presently seen as the most promising approach (Harrie & Weibel, 2007). Especially optimization approaches by iterative improvement such as simulated annealing are considered capable of handling diverse kinds of constraints (Ware & Jones, 1998). However, Galanda (2003) notes that simulated annealing "is not suited as an approach to model a comprehensive generalization process". A reason for this is seen in its high computational costs. In recent years the application of multi-agent systems has become a popular approach to model the generalization problem in a comprehensive way. In this approach, geographic entities such as buildings and roads are considered as communicating individuals (agents) that try to attain certain goals and therefore satisfy constraints (Barrault *et al.*, 2001). Galanda (2003) uses hill climbing for optimization in a multi-agent system for polygon generalization. This optimization technique, however, only yields locally optimal solutions. Generally, a trade-off needs to be found: An integrated approach allows a better respect of the dependencies between different generalization operators. However, the problem needs to be decomposed into subtasks, if more advanced optimization techniques than hill climbing are to be applied. Different general approaches can be observed that are used to break down the high complexity of the generalization problem:

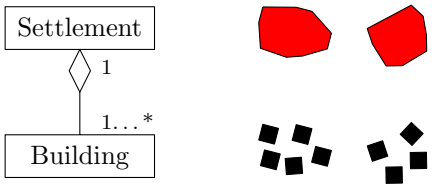
- *Sequencing of different generalization operators*: As explained in Sect. 2.1 many algorithms have been developed for certain generalization operators, for example, selection and simplification. In the simplest case we can apply the operators in sequence. For instance, we first select buildings for the target scale and then simplify their outlines. Experts can often find appropriate sequences, yet we generally need to be aware of possible dependencies between different operators. Imagine that we need to satisfy a size constraint for buildings. If we first eliminate all buildings that are smaller than a given threshold, then in a second step simplify their outlines, we can again end up with buildings below threshold, as simplification may imply changes of sizes.
- *Sequential processing of different types of objects*: Similarly, we can apply the same operator in sequence to different classes of objects. For example, in a first step, we resolve proximity conflicts between different roads by displacement. Then we resolve conflicts between buildings and roads, and between different buildings, with the same operator, this time keeping the roads fixed. In this example, we first process those objects whose positional accuracy is of a higher importance.
- *Spatial partition*: A common approach is to partition a dataset into smaller, manageable patches. For example, different tiles or map sheets are processed independently. However, this artificial separation will result in inconsistencies between adjacent map sheets. A more appropriate method is to partition the dataset according to existing structures of the landscape. Timpf (1998) introduced the trans-hydro graph, which comprises roads, railways, and rivers. Its faces are often defined as independent generalization zones.
- *Different levels of analysis:* Ruas (1999) proposes to define constraints on different levels of analysis, for example, on single buildings (micro level), building blocks (meso level), and the set of all buildings on the map (macro level). Constraints on higher levels of analysis are evaluated first to guide constraints on component objects such as single buildings, which then ensure their constraints autonomously.

- *Gradual reduction of scale*: Usually, the complexity of a generalization problem increases with the difference between the original scale and the target scale. An obvious approach to process large differences in scale is to proceed in multiple steps. In each step the constraints for an intermediate scale are satisfied. This is done until the final scale is reached. This approach is commonly used for the aggregation of areas, which is explained in Sect. 2.7.2.

Some of these general ideas were taken up to develop the generalization methods that are presented in this thesis. A sequencing of different generalization operators is proposed in Sect. 5.4, comprising collapse of areas to lines, aggregation, and line simplification. This workflow is similar to the workflow proposed by Jaakkola (1998), which, however, was designed for the generalization of land cover data in the raster model. Furthermore, Jaakkola (1998) did not apply optimization methods in his approach. Section 7.6.3 presents a heuristic aggregation method that spatially decomposes a problem instance into manageable pieces by introducing intermediate scales.

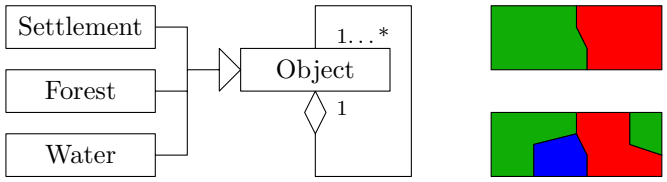
2.7 Aggregation

Aggregation denotes generalization processes that result in "part-of" relationships between objects in the original scale and the target scale. Figure 2.3 illustrates a typical aggregation problem in map generalization. The large-scale map contains a set of spatially disjoint objects (buildings), which need to be aggregated to form composite objects of another class (settlements) in the smaller scale. A second problem is shown in Fig. 2.4. In both scales, objects form a planar subdivision. An object in the original scale can become part of a composite object of any class in the target scale. For example, a lake can become part of a forest and a forest can become part of a settlement. We discuss the first problem and the related problem of grouping lines in Sect. 2.7.1. Sect. 2.7.2 deals with the aggregation of areas in a planar subdivision.



(a) conceptual level (b) object level

Fig. 2.3: Aggregation of spatially disjoint objects of a certain type.



(a) conceptual level (b) object level

Fig. 2.4: Aggregation of areas in a planar subdivision (b).

2.7.1 Aggregation of spatially disjoint objects and line features

The aggregation of spatially disjoint objects of the same category is often divided into two parts: objects that form groups are found, and composite objects for the target scale are defined from these groups, for example, the geometric outlines of composite objects need to be defined. Generally, the detection of groups can be done in a first step of generalization, which is often referred to as data enrichment (Neun *et al.*, 2004; Lüscher *et al.*, 2007). Algorithms that are applied later need to preserve the structures that were found in the data enrichment process. Grouping of objects is often done according to principles of human perception, which, in early works of psychology, were subsumed by laws of Gestalt theory. For example, Wertheimer (1938) emphasizes the importance of proximity and similarity for perceptual grouping. Though this gives ideas about important criteria, their formalization for automated generalization and the definition of appropriate procedures for their satisfaction are generally open problems. The grouping of buildings is investigated by Regnauld (1996), Boffet & Serra (2001) and Yan *et al.* (2008). However, the problem also needs to be solved for other polygon or point features that are distributed on the plane, for example, islands (Steiniger & Weibel, 2007). Anders (2003, 2004) proposes a graph clustering method as a basic tool to find groups of points in map generalization. The extraction

of building groups based on their centroids is shown as an example. Ware *et al.* (1995) propose a so-called *adopt merge operator* based on a conforming Delaunay triangulation to define the geometric outline of a composite object that is defined by a set of disjoint polygons. Peter & Weibel (1999) use the term *amalgamation* for this operator.

Another problem is the grouping of line features, for example, roads or rivers. This is often done according to the principle of 'good continuation': consecutive line segments of a network are grouped into so-called strokes, where the turning angles at junctions are kept small (Thomson & Brooks, 2002). Heinzle & Anders (2007) apply pattern recognition techniques to find groups of roads that are important for a smaller scale, for example, roads that form grid structures, star-like structures or ring structures. Considering the aggregation of areas in a planar subdivision, the method of Heinzle *et al.* (2006) for the detection of ring structures is of particular relevance, as their approach is based on properties of faces in the road network. According to their definition, each contiguous shape without holes that contains a given center and that can be obtained by union of a subset of road network faces is a feasible ring structure. Hence, the grouping of road segments is done by aggregation of faces in the network. In order to obtain the most circular ring, a compactness measure based on geometric moments is optimized. In fact, this is a special case of the spatial allocation problem discussed by Shirabe (2005a). Heinzle *et al.* (2006) solve this combinatorial optimization problem by explicit enumeration, but only handle a small number of faces. To obtain an acceptable performance, the road network is thinned out in a first step, such that no more than an empirically determined number of faces persists.

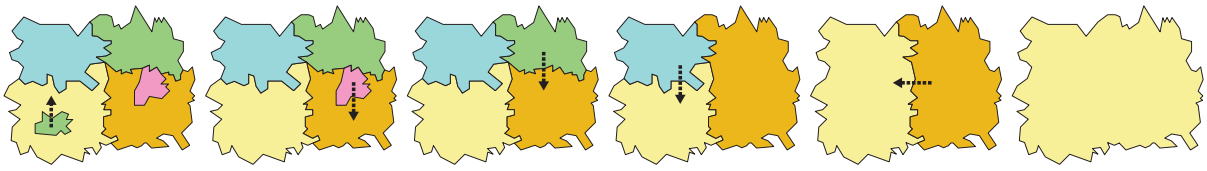
2.7.2 Aggregation of areas in a planar subdivision

Different authors have proposed generalization methods for areas in a planar subdivision. This representation is commonly used for areas of different land cover classes, for example, in the ATKIS datasets (recall Fig. 1.1). To ensure size constraints for the target scale some areas need to be aggregated with others. The approach of van Oosterom (1995) is most commonly applied; this is specified in Algorithm 1. We refer to this method as *region-growing*. In each iteration, the smallest area is selected and merged with one of its neighbors. Figure 2.5 shows a sample at different iterations. In fact, van Oosterom (1995) proposed to select areas according to a defined importance value, which can be different from the size. However, we use size and importance synonymously, in order to better demonstrate the approach.

The algorithm defines the generalization process as a gradual transformation in scale. At each step the map satisfies a certain size threshold, which increases in each iteration. The algorithm stops, if defined thresholds are reached. Alternatively, we can continue the process until only one area remains as it is shown in Fig. 2.5.

Algorithm 1 Iterative aggregation of areas (region-growing)

1: $S \leftarrow$ set of areas below threshold
2: **while** $S \neq \emptyset$ **do**
3: $\quad a \leftarrow$ smallest area in S
4: $\quad$ merge a to most compatible neighbor
5: $\quad S \leftarrow$ set of areas below threshold
6: **end while**



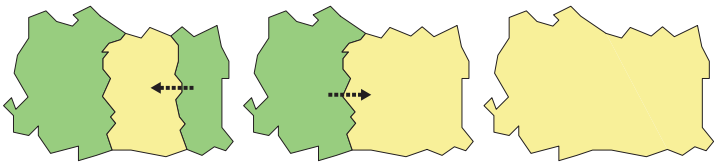
(a) step 0 (b) step 1 (c) step 2 (d) step 3 (e) step 4 (f) step 5

Fig. 2.5: Six steps of Algorithm 1; graphics are modified after van Oosterom (2005).

Existing implementations mainly differ in the definition of compatibility, which is used to select the neighbor in line 4. Cheng & Li (2006) compare two common settings. The first option is to select the neighbor with the longest common boundary. The second option is to select the neighbor that is semantically most similar; in their model, this criterion is based on fuzzy class membership values of areas. Different other models for the treatment of semantics exist. Podrenek (2002) defines priority lists for class changes, for example, a farmland area is rather merged with grassland than with water. Yaolin *et al.* (2002) define a semantic similarity matrix for classes. A similar model is applied in Sect. 5.3, yet in a global approach.

Several variations of Algorithm 1 exist that apply a different order of merges. For example, Bobzien (2001) defines an inner loop in Algorithm 1, such that the area resulting from the merge in line 4 is again merged with a neighbor, until it satisfies the given area threshold. The approach of van Smaalen (2003) is to iteratively treat pairs of classes, whose order is defined by a class adjacency index. This is based on the total length of all boundaries separating the members of any given pair of classes. In each iteration, those adjacent objects of both classes become merged that are selected according to a second measure, referred to as object aggregation factor. This is based on boundary lengths and areas of single objects. As information on land cover is often given in the raster data model, special generalization methods for raster data sets have been developed. In this context, Schylberg (1993) and Jaakkola (1997a,c) apply iterative methods, which are similar to Algorithm 1.

The gradual transformation defined by Algorithm 1 can be used for zooming and progressive transmission of datasets, for example, in web cartography (van Oosterom, 1995, 2005). Zooming from small to large scale, the maps in Fig. 2.5 can be shown in succession, ordered from right to left. However, merging small areas to neighbors is also commonly used to produce a single output dataset (Cheng & Li, 2006). A gradual transformation between both datasets is not needed in this case. In Chapter 4 we will see an additional feature of the iterative method: if a change happens to the original dataset, only certain parts of the generalized map will need to be updated. Despite this benefit, we need to discuss the performance of this algorithm in terms of quality. Clearly it satisfies the hard size constraints and produces contiguous regions, while ensuring that the output is a planar subdivision. However, the rigid structure of the algorithm does not allow soft constraints to be resolved optimally, for example, to minimize class changes. Figure 2.6 shows a second sample processed with Algorithm 1. The dominating color in the input is green, yet the resulting area in step two is yellow. This result does not depend on our definition of compatibility: in each iteration the smallest area has only one neighbor.



(a) step 0 (b) step 1 (c) step 2

Fig. 2.6: Three steps of Algorithm 1, leading to a large amount of class changes.

Cheng & Li (2006) use global quality measures to compare results that were obtained with two different definitions of compatibility. For example, they compare the results according to the amount of class change, which they refer to as a measure of "semantic consistency". However, the new method that is presented in Chapter 7 goes an important step further: A global quality measure is not only used for assessment, but for optimization, in order to obtain results of maximum quality under given hard constraints. In Sect. 8.3 results of Algorithm 1 are compared with those of the newly developed aggregation method.

2.8 Multiple Representation Databases (MRDBs) and their update

So far we have considered datasets of different scales as products that are independently offered to support different applications of users. However, to make the updating process efficient, datasets of different scales are often maintained in the same database (Kilpeläinen, 2000). Also, for many applications there is not a single, optimal scale. For example, Walsh *et al.* (2004) point out "that multithematic and multiscale perspectives are essential to effectively address complex human-environment interactions and their linkage to landscape form

and function". Generally, a database that keeps datasets (layers) of different themes or scales for the same spatial extent is referred to as *multiple representation database*. In the case of different scales, the term *multiple resolution database* is often used. Correspondences between objects of different datasets are explicitly stored as links in an MRDB. These can be exploited for multi-scale analyses; the updating problem can be seen as a special analysis task, which is explained in Sect. 2.8.2. MRDBs are often designed with the architecture of a federated database system (Anders & Haunert, 2004; Mantel & Lipeck, 2004). Generally, this is a collection of cooperating but autonomous component database systems (Sheth & Larson, 1990).

Multiple resolution databases are often used for visualization tasks that require fast zoom functionalities. Cecconi (2003), Hampe *et al.* (2004), and Hampe (2007) present their application in web cartography. As previously generalized results can be stored in an MRDB, the computational overhead during user interactions is greatly reduced. The link structure can be used to query objects that appear when changing the level of detail. For example, aggregation relationships of cardinality 1:m can explicitly be stored. When zooming in, components of composite objects can be efficiently accessed.

Before reviewing methods for MRDB-update, we need to explain two different ways to set up an MRDB.

2.8.1 MRDB set-up

Hampe *et al.* (2003) distinguish two different ways to set up an MRDB:

- Set-up by generalization
- Set-up by matching

An MRDB with different DLMs can be set up by applying methods of model generalization; an MRDB with different DCMs can be set up by cartographic generalization. Normally the dataset with the largest scale is given as input; all other datasets are derived consecutively. In this case, correspondences between features result as a byproduct of the generalization process and can be directly transferred to the link structure of the MRDB. Furthermore, generalization rules and operators that are used for the MRDB set-up can be documented and provided for later updates. Also the sequence in which generalization rules and operators are applied, as well as the order in which different features are processed, can be recorded.

The second possibility for an MRDB set-up is to use existing datasets that were independently collected. In this case, links can be acquired with automatic matching techniques. Before matching or linking homologous objects of different datasets, correspondences between abstractions need to be found. This process often leads to a global schema or multi-scale schema. Devogele *et al.* (1996) define a multi-scale schema by introducing scale transition dependencies between classes from different schemata. For the matching task, attributes, geometric properties and/or topological relationships between features of different datasets can be compared. These three aspects are evaluated sequentially by Cobb *et al.* (1998) for each candidate match to exclude false matches. Walter & Fritsch (1999) present a solution for the matching of roads in datasets of different thematic domains, that is, a topographic database and a dataset for car navigation. Mustière & Devogele (2008) have developed a matching method for networks of different levels of detail. We can exploit such links to define a rubber-sheeting transformation between both representations, which allows objects to be transmitted from one representation into the other (Haunert, 2005b).

Often inconsistencies between different datasets hinder their matching. This concerns geometrical distortions and, which is a more severe problem, topological differences between both datasets, for example, there is no exact isomorphism between different representations of the road network. Even correspondences between schemata are often missing. Thus, links found by matching are less complete and more erroneous than links recorded during generalization. Furthermore, the set-up by generalization allows to record "rich" links, which express functional dependencies between different datasets and not only correspondences between objects. On the other hand, generalization knowledge is implicitly given by links, which were found by matching. Sester *et al.* (1998) and Dunkars (2004) apply data mining techniques to extract generalization rules from linked datasets.

2.8.2 Incremental generalization for update

The close connection between datasets of different scales in an MRDB can be used for incremental updating. Normally it is assumed that changes in the physical world are captured in the largest scale and propagated into the smaller scales in the MRDB. This can be done without processing the complete dataset, since only directly changed and potentially influenced objects need to be generalized. This approach is generally called *incremental generalization* (Kilpeläinen & Sarjakoski, 1995), which is in contrast to *complete generalization*. Different types of updates need to be considered; Anders & Bobrich (2004) distinguish insertion, deletion and change of objects.

Assuming that updates are permanently inserted into the most detailed layer of an MRDB, we could at any time create updated layers of smaller scales, if we had an automatic method for complete generalization. Even then, the complete generalization of large datasets is very time-consuming. Since updates usually have a limited spatial extent, the computational effort can be reduced with an incremental generalization method. Skogan & Skagestein (2005) claim: "The two methods, however, should produce identical target dataset[*sic*] using the same source datasets as input." Of course, in this case the result is independent of the order in which updates are processed. Skogan and Skagestein present a system that records "productions" during an MRDB set-up by generalization. Similarly to a link, a production stores correspondence relationships between objects. Additionally, it provides information about an applied generalization rule, and thus defines functional dependencies. These are exploited to find objects that need to be updated to accomplish the defined task. However, the presented system only handles a small set of simple rules. It requires a large number of productions to be maintained; that might become infeasible due to storage limitations.

The claim for identical outputs of incremental and complete generalization methods only makes sense, if a complete generalization method is given that yields well-defined results. This, however, does not apply to an MRDB that was set up by matching. Even in the case that the MRDB was set up by generalization, the derived scales are not well-defined, for example, if nondeterministic methods like simulated annealing were applied. For these cases, we need another notion of incremental generalization: we seek an updated dataset of reduced scale that sufficiently satisfies the defined generalization constraints. In fact, most approaches to incremental generalization do not satisfy the strict claim of Skogan & Skagestein (2005). For example, Harrie & Hellström (1999) and Dunkars (2004) present prototype systems for incremental generalization of roads and buildings. These use special rules for the update scenario. In both cases, the authors note that the result after several updates depends on the order in which the updates were inserted. Though the authors consider this as a deficit of their methods, their primary concern is to satisfy constraints. For example, Dunkars (2004) discusses topological constraints and proximity conflicts.

2.9 General remarks

As we have seen, many researchers consider area aggregation in map generalization as an important problem. Furthermore, optimization is generally considered as one of the most promising approaches to map generalization. Surprisingly, the area aggregation problem has not been approached by optimization yet. This motivates to develop a new aggregation method by optimization.

Hard constraints are relaxed in most optimization approaches to map generalization. Most researchers justify this approach by stating that different constraints are conflicting, that is, without relaxing the constraints the map generalization problem would be over-constrained. However, this argument does not legitimate a relaxation of all hard constraints. In particular, constraints that express standards from database specifications must be satisfied in any case.

Often we can easily satisfy such standards, for example, we can apply the simple iterative aggregation algorithm to ensure that all areas have sufficient size. However, we need to handle hard constraints in optimization approaches if we want to create high-quality maps. In the context of map generalization, this problem indeed has not been investigated in sufficient depth. Solutions exist mainly for geometric problems that have been addressed by researchers in the field of computational geometry.

Many researchers applied geometric measures as optimization objectives for map generalization, for example, point movements should be small. In contrast we discuss how to optimize a measure of semantic accuracy that is based on class distances.

Chapter 3

Combinatorial Optimization

In order to explain the theoretical background of the developed generalization methods, this chapter reviews basics of combinatorial optimization. This includes general definitions and examples (Sect. 3.1), complexity theory (Sect. 3.2), and general approaches to NP-hard problems (Sect. 3.3). Approximation algorithms are explained in Sect. 3.4. Section 3.5 explains linear programming. Mixed-integer programming is explained in Sect. 3.6. Section 3.7 presents the basics of local search. Section 3.8 concludes with a comparison of mixed-integer programming and simulated annealing, since this thesis presents solutions for aggregation in map generalization based on these approaches.

Comprising popular heuristic methods like hill climbing and simulated annealing, local search is considered as the currently dominating optimization approach to map generalization (Galanda, 2003; Hardy *et al.*, 2007; Ware & Jones, 1998; Ware *et al.*, 2003). Aerts & Heuvelink (2002) and Tavares-Pereira *et al.* (2007) apply local search methods to spatial allocation problems.

A comprehensive introduction to combinatorial optimization is given by Papadimitriou & Steiglitz (1998). The textbook of Garey *et al.* (1974) has become a definitive book on computational complexity theory. Michiels *et al.* (2007) give a theoretical introduction to local search. More practical introductions are given by Reeves (1993) and Michalewicz & Fogel (2004), who present popular heuristic approaches.

3.1 Combinatorial problems

Generally, an *instance of an optimization problem* is defined by a pair (S, f), with S being the *set of feasible solutions* and $f : S \to \mathbb{R}$ being the *objective function*. Assuming that the problem is a minimization problem, f is also referred to as *cost function*. In this case, a solution $s^* \in S$ is *globally optimal*, if and only if $f(s^*) \leq f(s)$ for all $s \in S$. An *optimization problem* is a set of problem instances. If S is finite or infinite countable for all problem instances, the problem is usually called a combinatorial optimization problem. Combinatorial optimization problems are often defined in terms of graphs. This section introduces some types of graphs and two combinatorial problems, namely the shortest path problem and the problem MinimumVertexCover.

3.1.1 Graph definitions

A *graph* $G(V, E)$ is defined by two sets: a *node set* or *vertex set* V and an *edge set* $E \subseteq \big\{\{u, v\} \mid u, v \in V\,, u \neq v\big\}$, meaning that an edge $e \in E$ is an unordered pair of two distinct nodes. We call the nodes u and v of the edge $e = \{u, v\}$ *endpoints* of e. To illustrate a graph we can draw each node as a point at an arbitrary location in the plane. An edge can be represented by a (not necessarily) straight line connecting both endpoints. A drawing of a graph with no intersecting lines is called a *planar drawing* of the graph. If such a planar drawing exists for a certain graph, the graph itself is called *planar*. Figure 3.1 shows two drawings of a planar graph. The graph in Fig. 3.2 is non-planar. This graph is commonly referred to as K_5. A graph that contains an edge for each pair of nodes is called *complete*. Generally, the complete graph with n nodes is denoted by K_n.

A *directed graph* $G(V, A)$ or simply *digraph* is defined by a node set V and a set A of *directed edges* or *arcs*, that is, $A \subseteq \big\{(u, v) \mid u, v \in V\,, u \neq v\big\}$, see Fig. 3.3.

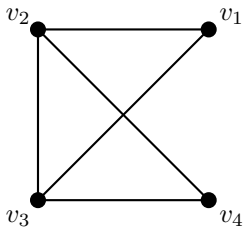
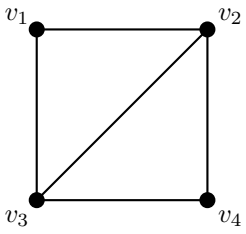


Fig. 3.1: Two drawings of the same planar graph. The right drawing is planar.



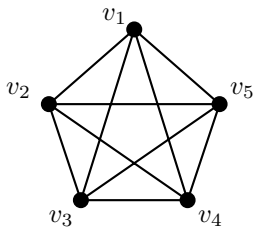


Fig. 3.2: A non-planar graph.

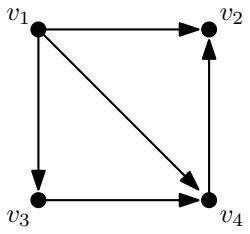


Fig. 3.3: A digraph.

3.1.2 The shortest path problem

Obviously, the concepts of graphs and digraphs are useful for diverse network or routing tasks, for example, the *shortest path problem*, which finds an application in car navigation systems. An instance of this problem is given with a graph $G(V, E)$, edge lengths $l : E \to \mathbb{R}^+$, an origin $u \in V$, and a destination $v \in V$. A *path* is defined as a sequence of nodes $p = (w_1, w_2, \ldots, w_k)$ without repetition, such that $\{w_i, w_{i+1}\} \in E$ for $i = 1\,, 2\,, \ldots\,, k-1$. The set of feasible solutions S contains all paths from u to v in G. The cost function f is defined as:

$$f(p) = \sum_{i=1}^{k-1} l(\{w_i, w_{i+1}\})\,. \tag{3.1}$$

The graph representation allows a problem to be investigated on an abstract level, separated from the specific application. Therefore, algorithms for the solution of the problem can easily be transferred to other domains. In Sect. 6.2 we will formalize the line simplification problem in map generalization as a shortest path problem. Though car navigation and line simplification do not seem to be related at first glance, both problems can be solved with the same algorithms.

We know from experience that computers can efficiently solve shortest path problems: Modern web services provide optimal nationwide routes within seconds. However, there are many other problems that are not as good-natured. The next section introduces an example of such a problem.

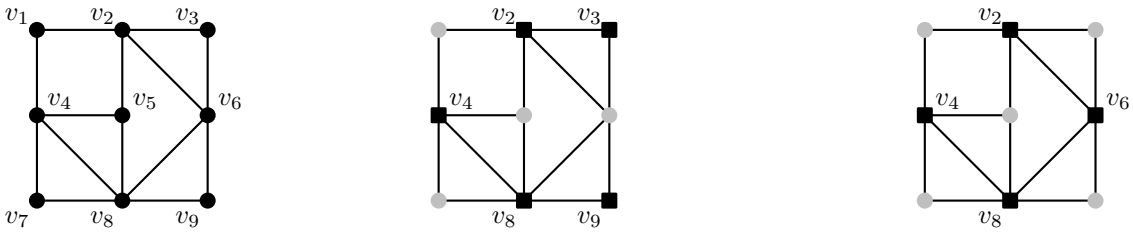
3.1.3 The problem MinimumVertexCover

In order to introduce the problem MinimumVertexCover, let's consider the following land surveying task: We are asked to measure a certain set of distances (represented by edges E) between stations (vertices V). Our equipment allows a distance to be measured between two stations by visiting only one of them, for example, we use a laser range-finder and each station is permanently equipped with a reflector. We want to measure all required distances, but, in order to minimize our workload, we aim to visit as few stations as possible.

The set S of feasible solutions of our problem contains those subsets of stations that correspond to a *vertex cover* of the graph $G(V, E)$. A vertex cover of G is a subset of V that contains at least one of both endpoints of each edge in E. Figure 3.4 shows a graph with two different vertex covers.

The *cardinality* of a set X is the number of elements in X, which is denoted by $|X|$. The problem MinimumVertexCover is to find a vertex cover of minimum cardinality, meaning that the cost function is defined by $f(C) = |C|$ for any subset C of V. The optimal solution for the example in Fig. 3.4(a) is the vertex cover $C = \{v_2, v_4, v_6, v_8\}$ in Fig. 3.4(c). Its cardinality is $|C| = 4$. It is not possible to find a vertex cover of G with cardinality less than four. Accordingly, we need to visit at least four stations in our land surveying problem.

Often a combinatorial problem is expressed as a *recognition problem* or *decision problem*. For the optimization problem MinimumVertexCover we define the related decision problem VertexCover. Given a graph G and an integer $c > 0$, this problem is to answer the following question: Does there exist a vertex cover of G with cardinality c or less? Thus, we just ask for a 'yes' (there exists a vertex cover of cardinality c or less) or a 'no'. Obviously, we can solve the decision version of the problem by solving its optimization version.



(a) a graph $G(V, E)$ (b) a vertex cover of G with cardinality 5 (c) a vertex cover of G with cardinality 4

Fig. 3.4: A graph with two vertex covers. Vertices included in vertex covers are displayed by black squares.

In contrast to the shortest path problem, VertexCover turns out to be very complex. We can solve the problem in a naive way by explicitly enumerating and testing all subsets of V with cardinality c. However, the number of such sets is $\binom{|V|}{c}$, say $|V|$ choose c, which even for small graphs can be extremely high. The next section explains how problems are classified according to their complexity.

3.2 Computational complexity theory

According to Papadimitriou & Steiglitz (1998) algorithms are "precise and universally understood sequences of instructions that solve any instances of rigorously defined computational problems." Usually we are interested in *efficient* algorithms, which we introduce in Sect. 3.2.1. However, we will fail to find efficient algorithms for certain problems like VertexCover. Section 3.2.2 explains how this differs from 'easy' problems like the shortest path problem.

3.2.1 Time complexity of algorithms

Generally, we assume that a problem instance is given by a string of a defined alphabet using a defined encoding scheme. The length of this string is referred to as size of the instance. Usually, algorithms need more time to solve a problem if we increase the size of the input, for example, the number of vertices in the problem VertexCover. We could try to express the behavior of an algorithm by a function that, for each possible input size, defines the maximal number of required basic processing steps, each of which takes one unit of time. However, this function might depend on the computer that is used to run the algorithm and the way the input is encoded. Therefore, the running-time behavior of an algorithm is commonly expressed by a class of functions that expresses the growth rate of the running time.

Formally, we say a function $f(x)$ is $\mathcal{O}(g(x))$ as $x \to \infty$, if and only if there are numbers $x_0 \in \mathbb{R}$ and $M \in \mathbb{R}^+$ such that $|f(x)| \leq M \cdot |g(x)|$ for each $x > x_0$.

For example, the shortest path problem can be solved in time $\mathcal{O}(|V| \log |V| + |E|)$ using the algorithm of Dijkstra (1959). Hence, without knowing the exact values of M and x_0, we know that there exists some pair of numbers M and x_0 such that the number of calculations for instances of size greater than x_0 is bounded by $M \cdot (|V| \log |V| + |E|)$.

In order to classify algorithms according to their efficiency, we distinguish algorithms whose required time is bounded by a polynomial *(polynomial time algorithms)* and algorithms whose required time is not bounded by a polynomial *(exponential time algorithms)*. We know from demonstrative examples that exponential functions grow extremely fast: If we put a grain of rice on the first square of a chess board and, for each subsequent square, double the number of grains, we need more rice than there exists in the whole world. Of course, we could find similar examples for polynomials of very high degree. However, time bounds for polynomial time algorithms that appear in practice seldom exceed degrees two or three. Furthermore, for large arguments, an increasing exponential function always surpasses a polynomial function, no matter how high we choose the degree of the polynomial. Therefore, the definition of a polynomial time algorithm as being efficient, is widely accepted.

3.2.2 Complexity of combinatorial problems

In order to classify combinatorial problems according to their complexity, we could distinguish problems that can be solved by polynomial time algorithms and problems that cannot be solved by polynomial time algorithms. We can prove that a problem falls into the first class by finding an efficient algorithm. However, if we do not find an efficient algorithm, this does not necessarily mean that no such algorithm exists. In fact, proving the intractability of a problem in this sense is extremely difficult. Because of this, another classification is commonly used.

Similar to our first idea, the complexity class $\mathcal{P}$ is defined to consist of all decision problems that can be solved by a *deterministic Turing machine* in polynomial time. This formal mathematical model is close to our general idea of a computer, meaning that the class $\mathcal{P}$ definitely contains the decision version of the shortest path problem. A second class, the class $\mathcal{NP}$, is defined to include all decision problems that can be solved by a *non-deterministic Turing machine* in polynomial time. Formal definitions of both, deterministic and non-deterministic Turing machines, are given by Garey & Johnson (1979).

For our discussion the following explanation of the class $\mathcal{NP}$ suffices, which uses the idea of a certificate-checking algorithm: Let's assume that an omniscient intelligence (an oracle) wants to convince us that the answer to our problem instance is 'yes'. In the example of the problem VertexCover it could provide us a set of vertices V' that is a vertex cover with the required cardinality. We could easily convince ourselves that this vertex set has indeed the demanded characteristics, or in other words, we could specify an efficient algorithm that checks the given certificate. In order to do so, we would need to check two things: Firstly, we would need to check whether V' contains no more vertices than the allowed number c. Secondly, we would need to check whether V' contains an endpoint for each edge. To solve both problems, we first need to iterate over the vertices in V'. In each iteration, we increase a counter and mark the selected vertex as 'visited'. After this loop, we iterate over the edges in E to check whether each edge has a visited endpoint. If this is true and the counter does not exceed c, the set V' meets our requirements. The algorithm requires $\mathcal{O}(E)$ time, that is, there is an efficient certificate-checking algorithm for VertexCover.

Because of this positive characteristic of the problem VertexCover, it is included in the class $\mathcal{NP}$. The class $\mathcal{NP}$ seems to be very general; it is clear that all problems in $\mathcal{P}$ are included in $\mathcal{NP}$, thus $\mathcal{P} \subseteq \mathcal{NP}$. However, to date it is not known whether both classes are in fact the same, that is, whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$.

So far we know that VertexCover is in $\mathcal{NP}$. In order to explain the difference to the shortest path problem, we need to introduce the concept of NP-hard problems. A problem is said to be NP-hard, if it has the following, very interesting property: Finding an efficient algorithm for its solution would imply that *all* problems in $\mathcal{NP}$ can be solved efficiently. Because of this, we can say that an NP-hard problem is at least as hard as any problem in $\mathcal{NP}$. An NP-hard problem that is in $\mathcal{NP}$ is said to be *NP-complete*. We can say that such a problem is among the hardest problems in $\mathcal{NP}$. As the set $\mathcal{NP}$ is only defined for decision problems, an optimization problem can be NP-hard, but not NP-complete. Figure 3.5 illustrates the discussed complexity classes assuming $\mathcal{P} \neq \mathcal{NP}$.

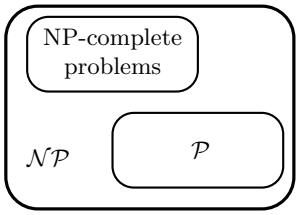


Fig. 3.5: The classes $\mathcal{P}$ and $\mathcal{NP}$, assuming $\mathcal{P} \neq \mathcal{NP}$.

In a fundamental proof, Cook (1971) showed that a certain combinatorial problem called Satisfiability is NP-hard. With this achievement, proving NP-hardness for other problems has become much easier: it suffices to show that efficiently solving the problem at hand allows *one* NP-hard problem to be solved efficiently, for example, Satisfiability. In fact, the problem VertexCover was proven to be NP-hard using this approach. Table 3.1 illustrates the history of proofs that led to this achievement. We say Satisfiability *polynomially reduces to* 3-Satisfiability, 3-Satisfiability polynomially reduces to VertexCover and so on. The proof

by Garey *et al.* (1974) shows that VertexCover remains NP-hard in the case when the graph G is restricted to be planar. We refer to this special version of the problem as PlanarVertexCover. In Sect. 7.3 we will use the NP-hardness of PlanarVertexCover to prove that the discussed aggregation problem is NP-hard. The presented proof is a typical example of such a reduction.

problem	reference to reduction
general definition of $\mathcal{NP}$	
↓	Cook (1971)
Satisfiability (SAT)	
↓	Cook (1971)
3-Satisfiability (3SAT)	
↓	Karp (1972)
VertexCover	
↓	Garey *et al.* (1974)
PlanarVertexCover	
↓	Section 7.3
AreaAggregation	

Table 3.1: Some NP-hard problems; the arrows represent reductions found by cited authors. The proofs for SAT, 3SAT and VertexCover can all be found in the textbook by Garey & Johnson (1979).

3.3 Approaches to solve NP-hard optimization problems

If we can prove that a certain optimization problem is NP-hard, it is extremely unlikely that there is a polynomial time algorithm for its solution. However, we can see such a proof as a success, since it shows that our failure to find an efficient algorithm is not due to our personal inability! If we do not want to give up the attack on the problem, we still have multiple options: we can apply (slow) exact algorithms, approximation algorithms, or heuristics.

When applying exact algorithms, we can often solve problem instances that appear in practice, in modest time. If we decide on this option, one possibility is to use mathematical programming, which allows existing implementations of advanced algorithms to be applied. In this context, the term "programming" must not be confused with the implementation of computer programs. Instead it is a general approach to formalize and solve a combinatorial optimization problem. The general term "mathematical programming" comprises, among others, linear programming and mixed-integer programming. The latter is of particular importance, as it allows so solve NP-hard problems. An advantage of this approach is that once a problem has been transferred into the required form existing software can be applied for the solution. The commercial solver CPLEX can be considered as state-of-the art software for this task (ILOG, 2008). Also public-domain software exists, for example, the free program lp_solve (LPSolve, 2008). Formalizing a problem as a mathematical program is useful, as we will directly profit from improvements to the general algorithms.

When giving up the claim for exact optimality, one option is to find an efficient *approximation algorithm*, which guarantees a solution of cost within a certain factor of the minimum. This is probably the most elegant approach to NP-hard problems. It is especially useful if an approximation factor close to one can be guaranteed. In fact, some approaches allow a problem to be solved with an approximation factor arbitrarily close to one. Unfortunately, some NP-hard problems are even hard to approximate.

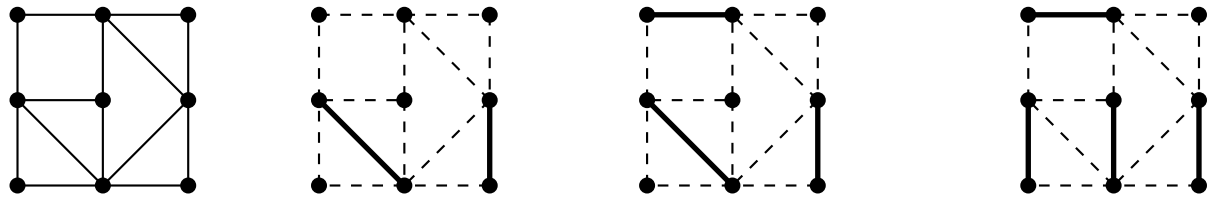
If we completely give up the claim for a performance guarantee, we can apply *heuristics*, which often yield relatively good solutions in reasonable time. Often we can agree on a 'local' optimum instead of insisting on a globally optimal solution. However, unless we define a neighborhood of a solution, the term local does not have any meaning.

The introduction of approximation algorithms will lead to the definition of another complexity class. Therefore, we first address this approach in more detail (Sect. 3.4). Sections 3.5 and 3.6 introduce linear programming and mixed-integer programming, respectively. Section 3.7 reviews some general concepts of local search and a simulated annealing approach, which is often used to avoid getting stuck in poor local optima.

3.4 Approximation algorithms

This thesis does not present any new approximation algorithms for generalization. However, an overview of combinatorial optimization should not leave this approach out. Approximation algorithms for line simplification have been presented, among others, by Bose *et al.* (2006) and Gudmundsson *et al.* (2007).

In order to explain a simple approximation algorithm for MinimumVertexCover, we first need to define the term *matching*: given a graph $G(V, E)$, a matching of G is a subset of the edge set E that contains no two edges with a common endpoint. A matching is called *maximal* if no edge of E can be added such that this property persists. Note that this is not necessarily a *maximum* matching, which consists of as many edges as possible. Figure 3.6 illustrates these definitions with examples.



(a) a Graph $G(V, E)$ (b) a matching of G (c) a maximal matching of G (d) a maximum matching of G

Fig. 3.6: A graph and three matchings. Edges included in matchings are displayed by solid lines.

Obviously, if the matching $E' \subseteq E$ is maximal, then the endpoints of edges $e \in E'$ define a vertex cover of G. This vertex cover has cardinality $2|E'|$. As each vertex cover of G contains at least one endpoint of each edge $e \in E'$, a vertex cover of G has cardinality at least $|E'|$. To conclude, a maximal matching E' defines a feasible solution to the problem MinimumVertexCover. Though this solution might not be optimal, we know that such a vertex cover never contains more than twice as many nodes as the vertex cover of minimum cardinality. Therefore, an algorithm that yields a maximal matching is a factor-2 approximation algorithm for MinimumVertexCover. We can define this algorithm with a simple iteration through the edge set; in this case it needs time $\mathcal{O}(|E|)$.

For many applications, an approximation factor of two is certainly rather meaningless. In our land surveying problem from Sect. 3.1.3, we would possibly visit twice as many stations as necessary! On the other hand, the results of heuristics, though often useful in practice, can be arbitrarily bad in theory. Certainly, we are interested in finding efficient algorithms with approximation factors as close to one as possible. An interesting fact is that many NP-hard problems can be approximated arbitrarily well, meaning that, for any $\varepsilon > 0$, we can find a polynomial time algorithm that guarantees an approximation factor of $(1 + \varepsilon)$. We can define a general algorithm that requires ε and an instance of the optimization problem as input. If it ensures the approximation factor of $(1 + \varepsilon)$ and if the required time is bounded by a polynomial in the instance size, it is called a *polynomial-time approximation scheme* (PTAS). Unfortunately, when providing a PTAS with a small ε-value, the degree of the time bound is often extremely high. As we are concerned about the increase of the time complexity with respect to ε, we also give the definition of a *fully polynomial-time approximation scheme* (FPTAS): a PTAS is called FPTAS if its running time is bounded by a polynomial in the instance size and in $1/\varepsilon$. Though PTAS and FPTAS are known for many NP-hard problems, it is also known that, unless $\mathcal{P} = \mathcal{NP}$, some NP-hard problems cannot be approximated arbitrarily well. Such problems are called *APX-hard*. For example, it is NP-hard to approximate MinimumVertexCover with a factor smaller than $7/6$ (Håstad, 2001), thus MinimumVertexCover is APX-hard.

3.5 Linear programming

Linear programming is a special optimization problem. An instance of this problem is called a *linear program* (LP). In contrast to most other mathematical programming problems, linear programming can generally be solved efficiently. The algorithm of Karmarkar (1984), for example, runs in polynomial time – assuming that the input numbers are not too large. The set of feasible solutions S of an LP is neither finite nor countable, thus it is questionable whether linear programming should be classified as a combinatorial problem. However, some algorithms use the fact that there is an easily accessible, finite set of potentially optimal solutions. One of these methods is the *simplex algorithm* by Dantzig (1963). Though it is not a polynomial time algorithm, it usually performs well in practice. Linear programming is often seen as a sub-problem of integer and mixed-integer programming. Therefore, the simplex algorithm is an important subroutine in algorithms for many combinatorial problems. This section defines two different forms of LPs and sketches the simplex algorithm.

3.5.1 The canonical form of linear programs

The *canonical form* of an LP is defined as follows:

Given an $m \times n$ integer matrix A, an m-vector of integers b, and an n-vector of integers c,

$$\begin{aligned} \text{minimize} \quad & z = c^T \cdot x \\ \text{subject to} \quad & A \cdot x \geq b\,, \\ & x \geq 0\,, \\ \text{with} \quad & x \in \mathbb{R}^n. \end{aligned}$$

The vector of variables x represents solutions to the problem in terms of positive real numbers. Constraints being imposed on these variables define the set of feasible solutions. Each row of the matrix A defines a linear combination of variables, which is bounded below by the value in the same row of vector b. Geometrically, such a constraint defines a half space. Defining multiple constraints, the set of feasible solutions S becomes an intersection of many half spaces. If this set is bounded and not empty, it is a convex polytope. The vector c defines the optimization objective. Each value of this vector expresses a cost that is charged proportionally to a corresponding variable. The cost $f(x)$ of a solution $x \in S$ is equal to z. Since all solutions on a line perpendicular to c have the same cost, the aim is to find a feasible point on such a line closest to the origin.

Figure 3.7 shows a geometric interpretation of the example:

$$\begin{aligned} \text{minimize} \quad & z = x_1 + 2x_2 && (3.2)\\ \text{subject to} \quad & -16x_1 - 12x_2 \leq -35\,, && (3.3)\\ & 16x_1 - 12x_2 \leq 17\,, && (3.4)\\ & -2x_1 + 8x_2 \leq 21\,, && (3.5)\\ \text{with} \quad & x_1, x_2 \in \mathbb{R}_0^+\,. && (3.6) \end{aligned}$$

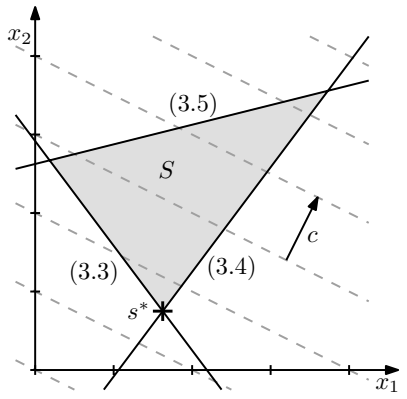


Fig. 3.7: A geometric interpretation of a linear program.

In this example, the vector x has two dimensions, that is, $x^T = (x_1, x_2)$. The set of feasible solutions is the shaded polygon. The dashed gray lines in Fig. 3.7 are lines of equal cost.

It is easy to see that, if the set of feasible solutions is bounded and not empty, the optimal solution s^* of an LP can always be found in one of the vertices of the convex polytope, which is defined by the constraints: the first vertex hit when sweeping the gray lines in the direction of c is the optimal solution. As the number of vertices is finite, an LP can be considered as a combinatorial problem. A possible approach to solve an LP is to enumerate all vertices. This, however, is very inefficient as the number of vertices increases exponentially in the size of the LP. Section 3.5.4 explains the simplex algorithm, which searches more intelligently for the best vertex.

The canonical form of an LP is very general. It also allows maximization problems, unbounded variables, and equality constraints to be expressed:

- A maximization problem can be transformed into a minimization problem, simply by multiplying the objective function by -1.
- A variable that is not restricted to positive numbers can be substituted by a difference of two variables, each being bounded below by 0.
- An equality constraint can be expressed by two inequality constraints.

However, in order to explain the simplex algorithm, we introduce the standard form of an LP in the next section.

3.5.2 The standard form of linear programs

The *standard form* of an LP is defined as follows:

Given an $m \times n$ integer Matrix A, an m-vector of integers b, and an n-vector of integers c,

$$\begin{aligned} \text{minimize} \quad & z = c^T \cdot x \\ \text{subject to} \quad & A \cdot x = b\,, \\ & x \geq 0\,, \\ \text{with} \quad & x \in \mathbb{R}^n. \end{aligned}$$

In contrast to the canonical form, the standard form expresses constraints by equations and not by inequalities. In Sect. 3.5.1 we introduced a transformation rule, which allows an equality constraint to be expressed by means of two inequality constraints. In order to show that the standard form of an LP is as general as the canonical form, we also need to introduce a rule for the opposite direction:

- A constraint $a_1 \cdot x_1 + \ldots + a_n \cdot x_n \leq b$ can be expressed by $a_1 \cdot x_1 + \ldots + a_n \cdot x_n + s = b$, with $s \geq 0$ being an additional variable, referred to as *slack variable*.

With this, our example from Sect. 3.5.1 becomes:

$$\begin{aligned} \text{minimize} \quad & z = x_1 + 2x_2 \\ \text{subject to} \quad & -16x_1 - 12x_2 + x_3 = -35\,, \\ & 16x_1 - 12x_2 + x_4 = 17\,, \\ & -2x_1 + 8x_2 + x_5 = 21\,, \\ \text{with} \quad & x_1, \ldots, x_5 \in \mathbb{R}_0^+\,. \end{aligned}$$

In this case, the matrix A is defined by:

$$A = \begin{pmatrix} -16 & -12 & 1 & 0 & 0 \\ 16 & -12 & 0 & 1 & 0 \\ -2 & 8 & 0 & 0 & 1 \end{pmatrix}.$$

3.5.3 Basic feasible solutions

This section explains an important relationship between a basis of A and a solution of an LP in standard form. We first define the matrix B to include all columns of such a basis, meaning that it includes m linear independent columns of A. In our example, a basis is given with:

$$B = \begin{pmatrix} -16 & -12 & 1 \\ 16 & -12 & 0 \\ -2 & 8 & 0 \end{pmatrix} .$$

The matrix B is nonsingular and has dimension $m \times m$. All columns that do not belong to this basis form the matrix N, whose dimension is $m \times (n - m)$. Consequently, we reorder the variables x in the same way into vectors x_B and x_N and their associated costs into vectors c_B and c_N. With this, we can write the equation $A \cdot x = b$ as:

$$B \cdot x_B + N \cdot x_N = b$$

and the cost z as:

$$z = c_B^T \cdot x_B + c_N^T \cdot x_N .$$

We can now express the variables x_B by:

$$x_B = B^{-1} \cdot b - B^{-1} \cdot N \cdot x_N . \tag{3.7}$$

The cost z becomes:

$$z = c_B^T \cdot B^{-1} \cdot b + \left(c_N^T - c_B^T \cdot B^{-1} \cdot N\right) \cdot x_N . \tag{3.8}$$

A solution of Eq. (3.7) is found with:

$$\begin{aligned} x_B &= B^{-1} \cdot b , \\ x_N &= 0 . \end{aligned}$$

Such a solution is called a *basic solution*; the variables x_B are called *basic variables*. Note that a basic solution may contain variables with negative values; in this case, it is not a feasible solution of the LP. We refer to a basic solution x as a *basic feasible solution* if the inequalities $x_B \geq 0$ hold.

In our example we have:

$$x_B = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = B^{-1} \cdot b = \begin{pmatrix} 0 & 1/13 & 3/26 \\ 0 & 1/52 & 2/13 \\ 1 & 19/13 & 48/13 \end{pmatrix} \cdot \begin{pmatrix} -35 \\ 17 \\ 21 \end{pmatrix} = \begin{pmatrix} 97/26 \\ 185/52 \\ 876/13 \end{pmatrix} \approx \begin{pmatrix} 3.73 \\ 3.56 \\ 67.38 \end{pmatrix} ,$$

which is basic feasible, as $x_B \geq 0$. The point $(3.73, 3.56)$ with cost $z = 141/13 \approx 10.85$, given by this solution, turns out to be the upper right vertex of the feasible region S in Fig. 3.7. An important fact is that all vertices of S correspond to basic feasible solutions like this. This fact is used by the simplex algorithm, which is explained in the next section.

3.5.4 The simplex algorithm

In Sect. 3.5.1 we argued that an optimal solution of an LP can be found in one of the vertices of the convex polytope that defines the set of feasible solutions. Hence, we proposed to solve an LP by searching through all vertices. We can now alternatively think of this process as searching for an optimal basis of A. Starting from a basic feasible solution, the simplex algorithm iteratively exchanges one column of the basis. In each iteration, a non-basic variable enters the basis and and a basic variable leaves the basis. Geometrically, this pivoting step means to move along an edge of the feasible region S to an adjacent vertex.

To refer to a single variable, let x_u be the value in row $u \in \{0, 1, \ldots, n\}$ of vector x. We define $\beta(v)$ such that $x_{\beta(v)}$ corresponds to row $v \in \{0, 1, \ldots, m\}$ of vector x_B. We define $\mu(w)$ such that $x_{\mu(w)}$ corresponds to row $w \in \{0, 1, \ldots, n - m\}$ of vector x_N.

Assume we try to increase the non-basic variable $x_{\mu(j)}$. We can do this, remaining feasible, until one of the basic variables becomes zero according to Eq. (3.7). At this point, we have found a new basic feasible solution.

In order to formalize this procedure, we introduce the $m \times (n-m)$ matrix X, whose values $X_{i,j}$ express the amount by which the basic variable $x_{\beta(i)}$ decreases, if the non-basic variable $x_{\mu(j)}$ increases by one. According to Eq. (3.7), this is:

$$X = B^{-1} \cdot N \,. \tag{3.9}$$

The amount θ_j by which $x_{\mu(j)}$ maximally can be increased is then:

$$\theta_j = \min \left\{ \frac{x_{\beta(i)}}{X_{i,j}} \;\middle|\; i = 1 \ldots m,\ X_{i,j} > 0 \right\} . \tag{3.10}$$

This means that by setting $x_{\mu(j)} := \theta_j$, one of the basic variables becomes zero and the constraint $x \geq 0$ is still satisfied. In our example we have:

$$X = B^{-1} \cdot N = \begin{pmatrix} 1/13 & 3/26 \\ 1/52 & 2/13 \\ 19/13 & 48/13 \end{pmatrix}$$

and

$$\theta_1 = \min \left\{ \frac{97}{26} : \frac{1}{13}, \frac{185}{52} : \frac{1}{52}, \frac{876}{13} : \frac{19}{13} \right\} = \min \left\{ \frac{97}{2}, 185, \frac{876}{19} \right\} = \frac{876}{19} , \tag{3.11}$$

$$\theta_2 = \min \left\{ \frac{97}{26} : \frac{3}{26}, \frac{185}{52} : \frac{2}{13}, \frac{876}{13} : \frac{48}{13} \right\} = \min \left\{ \frac{97}{3}, \frac{185}{8}, \frac{219}{12} \right\} = \frac{219}{12} . \tag{3.12}$$

Equation (3.11) means that, if we increase the non-basic variable $x_{\mu(1)} = x_4$, then $x_{\beta(3)} = x_3$ is the first basic variable that becomes zero. This happens at $x_4 = 876/19$. According to Eq. (3.12), the same basic variable leaves the basis when choosing $x_{\mu(2)} = x_5$ for pivoting; x_5 enters the basis at $219/12$.

In order to decide which of the non-basic variables to choose for entering the basis, we need to find out whether our choice will result in a reduction of cost. We introduce the $1 \times (n-m)$ matrix C whose value $C_{1,j}$ is the derivative of z with respect to the non-basic variables $x_{\mu(j)}$. According to Eq. (3.8) this is:

$$C = c_N^T - c_B^T \cdot B^{-1} \cdot N \,. \tag{3.13}$$

For our example we obtain:

$$C_{1,1} = 0 - \frac{1}{13} - 2 \cdot \frac{1}{52} = -\frac{3}{26} .$$

$$C_{1,2} = 0 - \frac{3}{26} - 2 \cdot \frac{2}{13} = -\frac{11}{26} .$$

We can now choose any non-basic variable $x_{\mu(j)}$ with $C_{1,j} < 0$ to enter the basis. This step changes the cost by $\theta_j \cdot C_{1,j}$. Let's first neglect the degenerated case $\theta_j = 0$, that is, we assume $\theta_j > 0$. In this case, the pivot results in a basic feasible solution of less cost. If we iteratively perform such improvements until no variable with $C_{1,j} < 0$ exists we obtain the globally optimal solution.

In our example, both possible pivots result in a decrease of cost. We are free to choose one of them. As usual we choose the non-basic variable with smallest value $C_{1,j}$, that is the variable $x_{\mu(2)} = x_5$. We can now calculate the new basic feasible solution according to Eq. (3.7). This leads to:

$$x'_B = \begin{pmatrix} x_1 \\ x_2 \\ x_5 \end{pmatrix} = \begin{pmatrix} 13/8 \\ 3/4 \\ 73/4 \end{pmatrix} = \begin{pmatrix} 1.625 \\ 0.75 \\ 18.25 \end{pmatrix} .$$

Applying Eq. (3.13) with $(x'_N)^T = \{x_3, x_4\}$, we obtain:

$$C'_{1,1} = \frac{11}{96} ,$$

$$C'_{1,2} = \frac{5}{96} .$$

As both values are greater than zero, no pivot will result in reduced cost. Therefore, the solution is optimal. This can also be seen in Fig. 3.7: The point $(1.625, 0.75)$ is optimal with cost $z = {}^{25}/_{8}$. Note that, choosing the pivot x_4 instead of x_5 we would first reach the vertex $(0.18, 2.67)$. However, we would find the optimum after one more iteration.

To process the iterations efficiently, it is very convenient to maintain the coefficients in Equations (3.7) and (3.8) in a table: the *simplex tableau*. For our example, the first basic feasible solution corresponds to the following tableau:

$-{}^{141}/_{13}$	0	0	0	$-{}^{3}/_{26}$	$-{}^{11}/_{26}$
${}^{97}/_{26}$	1	0	0	${}^{1}/_{13}$	${}^{3}/_{26}$
${}^{185}/_{52}$	0	1	0	${}^{1}/_{52}$	${}^{2}/_{13}$
${}^{876}/_{13}$	0	0	1	${}^{19}/_{13}$	${}^{48}/_{13}$

If we numerate the rows by 0 to 3, row 1 corresponds to the equation ${}^{97}/_{26} = x_1 + {}^{1}/_{13} \cdot x_4 + {}^{3}/_{26} \cdot x_5$, which is equivalent to the first row in Eq. (3.7). Rows 2 and 3 of the tableau complete this vector equation. Row 0 contains the matrix C, that is, it can be read as $z - {}^{141}/_{13} = -{}^{3}/_{26} \cdot x_4 - {}^{11}/_{26} \cdot x_5$ according to Eq. (3.8). If we wish to pivot on x_5, we simply need to multiply the last row with ${}^{13}/_{48}$, yielding 1 in the bottom right cell. We can multiply the obtained row with ${}^{11}/_{26}$ and add it to the zeroth row. If we do the same with values $-{}^{3}/_{26}$ and $-{}^{2}/_{13}$ for the first row and second row, respectively, we obtain the final tableau:

$-{}^{25}/_{8}$	0	0	${}^{11}/_{96}$	${}^{5}/_{96}$	0
${}^{13}/_{8}$	1	0	$-{}^{1}/_{32}$	${}^{1}/_{32}$	0
${}^{3}/_{4}$	0	1	$-{}^{1}/_{24}$	$-{}^{1}/_{24}$	0
${}^{73}/_{4}$	0	0	${}^{13}/_{48}$	${}^{19}/_{48}$	1

We directly obtain the solution of the LP in the left column and see that it is optimal as all values in C' are greater than zero.

Let's now consider the case that, for some j, Eq. (3.10) results in $\theta_j = 0$. This means that one of the basic variables is smaller than zero. The current basic feasible solution is called *degenerate*. Though the derivation of cost $C_{1,j}$ might be smaller zero, choosing the non-basic variable $x_{\mu(j)}$ for pivoting does not reduce the cost, as $\theta_j \cdot C_{1,j} = 0$. In fact, by accepting a sequence of such steps, we might return to our starting point. The algorithm will *cycle* and never terminate. A simple rule for pivoting that guarantees to avoid cycling is given by Bland (1977): Always choose the non-basic variable $x_{\mu(j)}$ with $C_{1,j} < 0$ in the lowest numbered column of the tableau to enter the basis. If there is a tie in the calculation of θ_j, that is, the minimum in Eq. (3.10) is obtained for more than one basic variable, remove from the basis the basic variable among them that is in the lowest numbered column.

Finally we need to discuss how to generally find an initial basic feasible solution to start the simplex algorithm. For this we introduce artificial variables $x^a \in \mathbb{R}^m$. In a first step the following LP can be solved:

$$\text{minimize} \quad z^a = e^T \cdot x^a \quad \text{subject to} \quad \begin{bmatrix} A & I \end{bmatrix} \cdot \begin{bmatrix} x \\ x^a \end{bmatrix} = b\,, \quad x, x^a \geq 0\,, \quad x \in \mathbb{R}^n\,, \quad x^a \in \mathbb{R}^m\,, \tag{3.14}$$

with I being an $m \times m$ identity matrix and e being an m-vector of ones. If $b \geq 0$ the solution $x = 0\,, x^a = b$ is basic feasible, thus we can start to solve this LP with the simplex algorithm. Otherwise we can simply ensure this condition by multiplying some constraint equations by -1. If we obtain the optimum $z^a = 0$ with $x^a = 0$, we have found a feasible solution for the original LP. We can then use this to start the simplex algorithm again, this time without the artificial variables. Note that we do not need this two-stage procedure if we obtained the LP in standard form from an LP in canonical form, since the slack variables always constitute a basic feasible solution.

3.6 Mixed-integer programming

An *integer linear program* or simply *integer program* looks very similar to an LP:

$$\begin{aligned} \text{minimize} \quad & b^T \cdot x \\ \text{subject to} \quad & A \cdot x \geq r\,, \\ & x \geq 0\,, \\ \text{with} \quad & x \in \mathbb{Z}^n. \end{aligned}$$

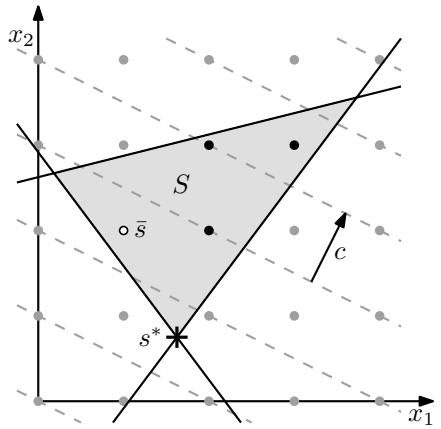


Fig. 3.8: A geometric interpretation of an integer linear program.

In contrast to the continuous variables in an LP, the variables in an IP are restricted to integer values. A graphical interpretation is depicted in Fig. 3.8; the solution $\bar{s}$ is globally optimal. Though the definitions of IPs and LPs are very similar, the computational complexity of integer programming is much higher. In fact the problem of solving IPs is NP-hard. However, a lot more problems can be expressed as IPs. Despite the high complexity, several general algorithms have been developed for the solution of IPs, which have been found to be useful for applications. This section explains the ideas behind some of these methods and presents an IP for MinimumVertexCover. First however we define the term mixed-integer program (MIP) as a combination of an LP and an IP. It may contain continuous as well as integer variables. Basically, a MIP can be solved with the same techniques as an IP.

3.6.1 Approaches to solve mixed-integer programs

Most approaches to solve mixed-integer programs can be subsumed by the term *implicit enumeration*. This means that all possible solutions are systematically evaluated without explicitly evaluating all of them. Usually the first step to solve an IP is to solve its LP relaxation. This is obtained from the IP by relaxing the integrality requirements for the variables, meaning that fractional values are allowed. Like any other LP the obtained problem can be solved, for example, with the simplex algorithm. Obviously, the optimum of an IP is found with this approach, if the optimal solution of its LP relaxation is integer. However, this case only occurs for special problems. Usually, rounding the values of the LP solution does not directly lead to the optimal solution of the IP, as can be seen in Fig. 3.8. The four integer solutions next to the optimal solution s^* of the LP are not even feasible. However, rounding is often used to divide a problem into two subproblems; this procedure is called *branching*.

A branching step means to select one variable x_i with fractional value x_i^0 in the solution of the LP relaxation and to split the original problem into two subproblems, each being defined by addition of one of the constraints $x_i \leq \lfloor x_i^0 \rfloor$ and $x_i \geq \lceil x_i^0 \rceil$, meaning that the value x_i^0 is rounded down and up, respectively. Again, the LP relaxations of both subproblems are solved. The branching procedure continues until each leaf node of the resulting tree represents an IP whose LP relaxation is either infeasible or has an optimal solution that is integer. Figure 3.9 illustrates this approach. In our example we obtain two subproblems, each by adding one of the constraints:

$$x_1 \leq 1 \qquad \text{and} \tag{3.15}$$

$$x_1 \geq 2\,. \tag{3.16}$$

The branching routine terminates at a finite number of iterations but requires an enormous computational effort. In order to avoid the tree becoming too large, we can use the fact that the LP relaxation offers a lower *bound* for the IP. Suppose that at a certain node of the branching tree, an integer solution with objective value z^0 is found, each tree node with lower bound greater or equal z^0 can be *killed*, which means that the branching

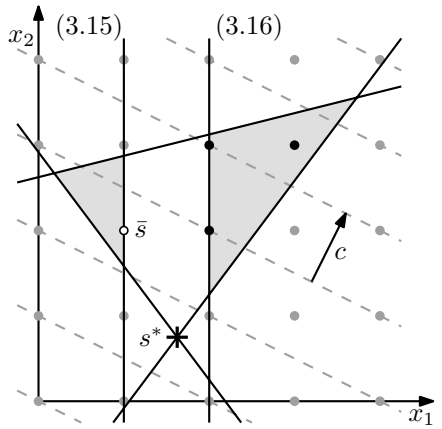


Fig. 3.9: The optimal solution of the LP relaxation s^* and the resulting sub problems when branching at variable x_1.

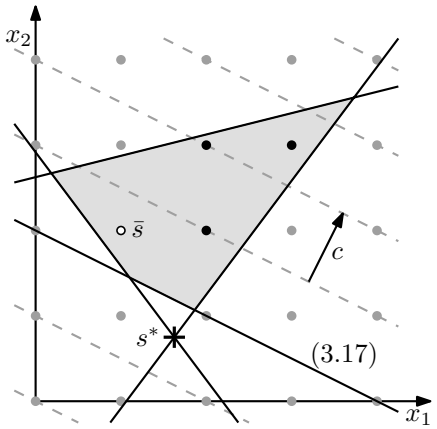


Fig. 3.10: The MIP after addition of the cut in Eq. (3.17).

process can be terminated in these nodes. This is simply because the bound guarantees that no solution better than z^0 can be reached when branching at such a node. The term *branch-and-bound* is used for the general approach of dividing a problem into subproblems and finding bounds for their costs. Since in the explained example a bound is found by solution of the LP relaxation, this special version of the approach is often referred to as LP-based branch-and-bound. The success of this method clearly depends on whether the bounds are *tight* or not, which means that the solution of the LP relaxation needs to be a good approximation for the solution of the IP.

Another approach to solve IPs and MIPs is based on *cutting plane algorithms*. A *cutting plane* or simply a *cut* is an additional constraint, that is, an inequality, that removes the solution of the LP relaxation from the feasible region of the IP, but does not exclude any integer solution. An example of a cut is shown in Fig. 3.10. The general approach of cutting plane algorithms is to iteratively solve the LP relaxation and to add a cut, until the solution of the LP relaxation satisfies the integrality constraints. In this case, the optimal solution of the MIP is found. Different methods exist to find cuts; one of the most prominent is the algorithm of Gomory (1958). We briefly explain this approach for the given example. In Sect. 3.5.4 we obtained the following equation in the final simplex tableau:

$$z = \frac{25}{8} + \frac{11}{96} \cdot x_3 + \frac{5}{96} \cdot x_4.$$

As x_3 and x_4 are positive numbers, rounding their coefficients down leads to:

$$z \geq \frac{25}{8}$$

and so

$$x_1 + 2x_2 \geq \frac{25}{8}\,.$$

As the variables x_1 and x_2 are restricted to integer values, the expression on the left side is integer. So, we can round the right side up, yielding the constraint:

$$x_1 + 2x_2 \geq 4\,, \tag{3.17}$$

which turns out to be the cut in Fig. 3.10. We could find additional cuts by proceeding in a similar way with the other rows of the simplex tableau. We can also solve the more constrained LP to find more cuts. An important fact is that we do not need to restart the simplex algorithm for this, though the current solution becomes infeasible. This is because the new simplex tableau gives a feasible solution to the *dual LP*, which is closely related to the *primal LP* that actually is to be solved. If we continue to solve the dual LP, we will obtain the optimal solution of the primal LP as well. More information on duality is given by Papadimitriou & Steiglitz (1998). Cutting plane algorithms usually have a slow convergence. Often a more efficient approach is to combine branch-and-bound and cutting plane algorithms in a method termed *branch-and-cut* (Mitchell, 2002). Adding cuts, the LP relaxation becomes tighter, thus applying branch-and-bound on the more constrained problem can

result in a significantly smaller branching tree. Currently, this method is implemented in the most powerful general purpose optimizers such as the software CPLEX (ILOG, 2008).

As the solution of a MIP is generally NP-hard, existing solvers have an exponential-time performance. Thus, problems that allow better solutions should not be approached by (mixed-)integer programming. Also there are often several alternatives to express the same problem as a mathematical program; choosing among them can result in significant differences in terms of performance. Generally the number of variables and constraints should be kept small. However, the tightness of the LP relaxation is often considered as a more important criterion.

3.6.2 Modeling with decision variables: an IP for MinimumVertexCover

In order to demonstrate how easily combinatorial problems can be modeled as integer programs, this section presents a simple IP for the problem MinimumVertexCover from Sect. 3.1.3. In fact, we only need binary variables or *decision variables* to model the problem. This special case of an IP appears very often and it is sometimes referred to as a *binary program*. We can express a binary variable simply by an integer variable $x \in \mathbb{Z}$ with constraints $x \geq 0$ and $x \leq 1$. In our example, the following variables express the solution:

$$x_v \in \{0,1\}\,, \qquad \text{with } x_v = 1 \text{ if and only if node } v \in V \text{ is included in the vertex cover}\,.$$

In our land surveying application, $x_v = 1$ would denote a visit to station v. In order to express our aim for a minimum number of selected vertices, we need to count the number of variables that are set to one. This is done by summing the values of all variables. The objective becomes:

$$\text{minimize} \quad \sum_{v \in V} x_v. \tag{3.18}$$

It remains to ensure that the selected vertices indeed define a vertex cover of G. For this, we need constraints that ensure one selected endpoint for each edge:

$$x_u + x_v \geq 1 \qquad \text{for all } \{u,v\} \in E. \tag{3.19}$$

This avoids that both variables, x_u and x_v, are set to 0. The IP formulations that appear in Chapter 7 are based on the same principles.

3.7 Local search

Local search is often used to attack NP-hard optimization problems. It is widely applied as a heuristic method, that is, there is no guarantee of performance. Nevertheless, there are well known facts about the performance of certain local search methods. For example, it is known that simulated annealing converges to the set of optimal solutions if certain conditions are satisfied. This section briefly reviews theoretical aspects of local search in general (Sect. 3.7.1) and simulated annealing in particular (Sect. 3.7.2). Section 3.7.3 discusses the handling of hard constraints in local search methods.

3.7.1 The neighborhood function and iterative improvement

A fundamental concept of local search is the *neighborhood function* $N : S \to 2^S$. The set $N(s) \subseteq S$ of a solution $s \in S$ is called the *neighborhood* of s. For example, the neighborhood of a feasible solution of MinimumVertexCover can be defined as the set of feasible solutions that can be obtained by adding or removing a single node.

Similar to our definition of a globally optimal solution in Sect. 3.1, a solution $s^* \in S$ is called *locally optimal* with respect to N, if and only if $f(s^*) \leq f(s)$ for all $s \in N(s^*)$. The *neighborhood graph* is a directed graph whose node set is defined by S and whose arc set contains an arc (s_1, s_2) for a pair $s_1, s_2 \in S$ if and only if $s_2 \in N(s_1)$. Algorithm 2 defines a simple iterative improvement procedure that yields a local optimum. The algorithm requires an initial feasible solution, which can even be NP-hard to find. Though a trivial start

Algorithm 2 Iterative improvement.

1: $s \leftarrow$ some initial solution
2: **loop**
3: $s' \leftarrow \bar{s} \in N(s)$ such that $f(\bar{s})$ is minimal
4: **if** $f(s') < f(s)$ **then** $s \leftarrow s'$
5: **else return** s
6: **end if**
7: **end loop**

solution often exists, there is no general answer to this problem. The rule in line 3 to select a solution from the neighborhood is known as *best improvement*. Also the procedure is often referred to as *steepest-ascent hill climbing*, though this term rather implies a maximization problem.

Usually, the neighborhood function is not given with the problem statement. If we wish to develop a local search method for a problem, it is our task to find an appropriate neighborhood function. Sometimes an *exact* neighborhood can be defined, which means that each local optimum is globally optimal as well. This special case applies to the simplex algorithm, as it finds the global optimum by iteratively moving to a more profitable adjacent vertex. The neighborhood $N(s) = S$ is exact for all problems. However, finding a local optimum would have the same complexity as the original problem. As an exact neighborhood for an NP-hard problem implies an exponential running time of local search, we give up the claim for global optimality. In this case, designing a neighborhood function is mainly based on practical considerations. Generally, a trade-off needs to be found: A large neighborhood may be hard to explore, but a too small neighborhood can produce many poor local optima. At least we should require that the neighborhood size is polynomial in the size of the problem instance. Otherwise it may require exponential time to attest that a solution is locally optimal. Furthermore, a good solution (preferably the optimal solution) should be reachable from any initial solution via the neighborhood graph. This condition is ensured, if the neighborhood graph is *strongly connected*, meaning that each two solutions can be reached from each other. The maximal distance on the graph between two nodes (the *diameter* of the graph) should be small.

3.7.2 Simulated annealing

The problem with iterative improvement is that we can get trapped in poor local optima. Several meta-heuristics exist, which use different ideas to escape them. An overview is given by Reeves (1993). Often these methods are inspired by phenomena in physics or biology. A typical example is *simulated annealing*, which simulates a controlled cooling process that is applied in metallurgy to minimize the number of defects in a material (Kirkpatrick *et al.* (1983)). Simulated annealing is a randomized algorithm, which extends Algorithm 2. This is specified by Algorithm 3.

Algorithm 3 Simulated annealing algorithm according to Michiels *et al.* (2007).

1: $s \leftarrow$ some initial solution
2: $k \leftarrow 1$
3: **repeat**
4: generate an $s' \in N(s)$
5: **if** $f(s') \leq f(s)$ **then** $s \leftarrow s'$
6: **else if** $\exp\left(\frac{f(s)-f(s')}{T_k}\right) > \text{random}[0,1)$ **then** $s \leftarrow s'$
7: **end if**
8: $k \leftarrow k + 1$
9: **until** stop criterion

Instead of selecting the best neighbor, in each iteration a neighbor is selected at random (line 4). Normally the selection is made according to an equal distribution. The generated neighbor is generally accepted if this

improves the solution (line 5), but even if it means an increase of cost, it is accepted with a certain probability (line 6). The parameter T_k defines the *temperature* at iteration k. For large values of T_k the probability of accepting a more expensive neighbor is relatively high. The temperature T_k is decreased during the simulation, for example by defining:

$$T_k = \alpha^k \cdot c_0 \,, \tag{3.20}$$

with constants $\alpha \in (0, 1)$ and $c_0 \in \mathbb{R}^+$. If T_k approaches zero, only improvements are accepted.

As explained by Michiels *et al.* (2007), a run of simulated annealing can mathematically be modeled as a Markov chain. Given that $i \in S$ is the solution in step $k \in \mathbb{N}$, we can express the probability that the neighbor $j \in N(i)$ is generated and the probability that this move is accepted. Consequently, if we are given an initial solution, we can express the probability that the algorithm arrives in a given step at a certain solution. This idea has led to a proof that, under certain conditions, simulated annealing converges to the set of globally optimal solutions S^*. Formally this can be expressed by:

$$\lim_{k \to \infty} \mathbb{P}\{X(k) \in S^*\} = 1 \,, \tag{3.21}$$

with $X(k)$ being the solution in step k. Four conditions suffice to ensure this result (Michiels *et al.*, 2007):

(S1) The neighborhood graph must be finite.
(S2) The neighborhood graph must be strongly connected.
(S3) The neighborhood graph must be *symmetric*, meaning that $s' \in N(s)$ implies $s \in N(s')$.
(S4) The control parameter T_k needs to be cooled down slowly enough.

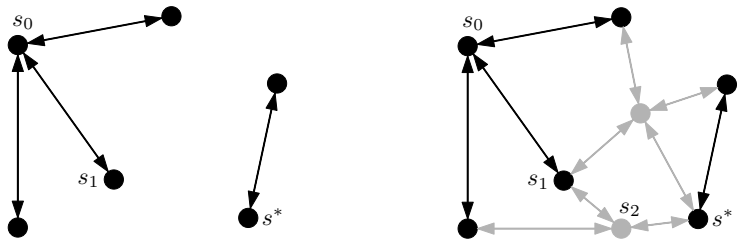
A formal definition of the last requirement is given by Hajek (1988).

Of course, explicitly enumerating all solutions to find the global optimum would be still better than an infinite run of simulated annealing. However, Michiels *et al.* (2007) note: "The relevance of the convergence result for simulated annealing is more subtle, however. It states that, when moving through a neighborhood graph, simulated annealing searches its way to the set of globally optimal solutions". It is therefore believed that the above requirements should be respected when designing a neighborhood function.

3.7.3 Handling hard constraints

In many optimization problems the set of feasible solutions is implicitly given with a set of (hard) constraints. The handling of hard constraints by meta-heuristics is a difficult problem. Michalewicz & Fogel (2004, Chapter 9) discuss this problem in detail for evolutionary algorithms. Most of the considerations can be generalized, that is, they also apply to simulated annealing and other local search methods.

Figure 3.11(a) displays the neighborhood graph for an instance $I = (S, f)$ of a combinatorial optimization problem. The black nodes in Fig. 3.11(a) form the set of feasible solutions S. Arcs display the neighborhood function N that was defined in order to approach the problem by local search. Apparently, the graph is not connected: given the start solution s_0 we cannot reach the globally optimal solution s^* via a feasible path. We may end up with a poor locally optimal solution, for example, s_1.



(a) original problem (b) relaxed problem

Fig. 3.11: Solutions (nodes) and their neighborhoods (arcs directed to neighbors) for an instance of a combinatorial optimization problem. Originally, the neighborhood graph is not connected (a). Relaxing a constraint the neighborhood graph becomes connected (b). Now the optimal solution s^* can be reached from s_0.

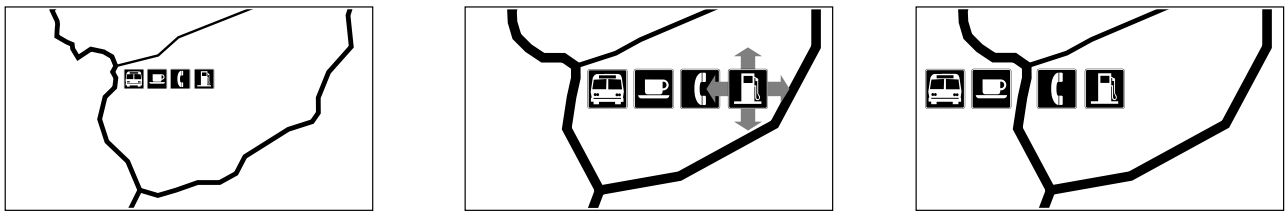
Finding a neighborhood that connects all feasible solutions can be difficult. Therefore, a hard constraint often needs to be relaxed, which means that additional solutions are permitted. Relaxing a certain constraint in the example of Fig. 3.11 yields the new solution set S' with $S' \supseteq S$. The two additional solutions that are in S' but not in S are displayed as gray nodes in Fig. 3.11(b). We introduce the cost function f' and the neighborhood function N' for the relaxed problem. Normally, the definitions of f and N are simply generalized to define the functions f' and N', that is, $f'(s) = f(s)$ and $N'(s) \supseteq N(s)$ for $s \in S$. We call (S', f') the instance of the relaxed problem. Now, in the instance of the relaxed problem, there is a path from s_0 to s^*, that is, $(s_0\,, s_1\,, s_2\,, s^*)$. However, though s_2 is not a feasible solution of I, it could be locally or even globally optimal in S' according to the cost function f'. In conclusion: if we relax a hard constraint, we run the risk of generating infeasible solutions. Else we run the risk to end up with poor local optima. There are two approaches to this dilemma:

1. We can define the cost function f' such that infeasible solutions (solutions in $S' \setminus S$) are penalized, thus it becomes more likely to end up with a feasible solution in S.
2. We can define a 'repair' procedure that, for any solution in $S' \setminus S$, yields a feasible solution in S.

In both approaches, we need to make crucial design decisions. Which penalty should we charge for an infeasible candidate? How should we repair an infeasible candidate, in order to obtain a good feasible solution? When should we invoke the repair procedure, for example, each time an infeasible candidate becomes selected or only if our final solution is infeasible? Abramson *et al.* (1996) discuss these design decisions for a simulated annealing approach to the general problem of solving binary programs. However, though the authors emphasize the generality of their approach, simulated annealing is usually more efficient when tailored to a specific problem.

Let's discuss these design issues in the context of generalization. In this application, local search methods are usually applied to iteratively improve the input map (Ware & Jones, 1998). However, as database specifications often define hard size constraints for the target scale, the input map is not a feasible solution. Hence, we can only start our local search method with the input map, if we relax these hard constraints. Additional costs or penalties need to be charged instead, for example, if a region is smaller than its required size. We can put high weights on such soft constraints, but this approach does not guarantee compliance with database specifications. Alternatively, we could apply a repair procedure to the input map that yields a feasible initial solution. For example, an empty map or a map that only contains one big area of a single class will usually comply with database specifications. The disadvantage is that such a solution will be poor in terms of quality, that is, the cost for the solution would be extremely high. Consequently, our search will require many iterations to find a good result. Furthermore, even if we find a feasible start, it can be impossible to reach good feasible solutions: again, if we reject infeasible candidates during the search, there may be no feasible path to the optimal solution.

These considerations become apparent if we take a look at the label placement problem in Fig. 3.12. In the input map (Fig. 3.12(a)), each label clearly corresponds to a location in the road network: a user will certainly understand that all facilities are on the left road, close to the road junction. If we generalize this map to a smaller scale, we aim to preserve clear correspondences between labels and the actual facility locations. Furthermore, for the sake of clarity, we do not allow intersections of labels and roads. In terms of constraints, we could define a soft constraint ("correspondences between labels and locations need to be clear") and a hard constraint ("a label and a road must not intersect").



(a) an input map of large scale (b) an initial solution for a smaller scale (c) a potentially optimal solution

Fig. 3.12: Cartographic label placement in map generalization is often approached by iterative improvement. Labels and roads must not intersect. Let's assume that we somehow find the initial solution in (b) and that the neighborhood is defined by small incremental label shifts, which, for one label, are illustrated by gray arrows. With this approach, the result (c) cannot be obtained when rejecting infeasible candidates during the simulation. This is because some labels need to cross a road in order to reach their optimal position.

Let's assume that we somehow found a cost function f that expresses the aim for clear correspondences, that is, the degree of non-satisfaction of the soft constraint. Furthermore, we have a repair procedure that yields a feasible start solution s_0 (Fig. 3.12(b)). In this solution, the correspondences of labels and locations are not clear: is the gas station on the left road or on the right road? Consequently, the cost for this feasible solution would be high. We can now try to iteratively improve the map. For example, we can pick a label at random and apply a small shift to a new location. The neighborhood $N(s_0)$ contains all maps that can be produced by applying such a move to the current map s_0. For a similar problem, Monnot *et al.* (2007) apply a neighborhood structure, which is also based on small incremental shifts of labels.

The map in Fig. 3.12(c) is a good solution, supposably the optimal solution s^* of the problem: the correspondences are clear. However, with our definition of the neighborhood, we cannot reach this solution without accepting infeasible candidates during our search. This is because two labels need to cross a road, in order to reach their optimal position. Again, this raises several questions. How should we penalize a map with intersecting labels and roads? How and when should we repair a map with intersecting labels and roads? As there are no general answers, the idea of a perfect general-purpose heuristic for map generalization is an illusion.

3.8 Concluding remarks

Mixed-integer programming and simulated annealing are two very different approaches to combinatorial optimization problems. As this thesis presents the application of both approaches to aggregation in map generalization, we recap their main differences.

Simulated-annealing is an iterative meta-heuristic. Though it theoretically converges to the set of globally optimal solutions, it only offers local optima in practice. A general drawback of simulated annealing is the tuning of parameters that are not inherent to the problem (we call such parameters "tuning parameters"). For example, a cartographer is probably able to specify parameters expressing his preferences for generalization, but setting up an annealing schedule and finding an appropriate design for the neighborhood function requires much experimentation. In contrast to simulated annealing, mathematical programming does not require tuning parameters. Furthermore, it is an exact approach, that is, it offers the globally optimal solution. Though we cannot sustain this statement under time constraints (solving a MIP is generally NP-hard), we can summarize the general advantages of mixed-integer programming compared to simulated annealing as follows:

+ Optimally solving small problem instances by mixed-integer programming allows the performance of heuristics to be assessed.
+ Mixed-integer programming does not require tuning parameters, that is, parameters that are not inherent to the problem.
+ Mixed-integer programming allows linear constraint equations and inequalities to be introduced. In contrast, simulated annealing often requires hard constraints to be relaxed, in order to produce solutions sufficiently close to the optimum.
+ The approach by mathematical programming is deterministic. Randomized algorithms like simulated annealing are not. Using a pseudo-random number generator, each time with the same initiation, would ease this problem. However, a reproduction of experimental results is still hard for other researchers.
+ Mixed-integer programming allows existing software to be applied.

On the other hand, the mathematical programming approach has a severe limitation:

− The possibility of expressing constraints and objectives is limited, often restricted to linear expressions.

We will see in Sect. 7.2 that this limitation especially affects the possibility of expressing the compactness of a shape.

Chapter 4

An Incremental Aggregation Algorithm for Efficient Updating

This chapter presents an incremental version of Algorithm 1 (see page 23), that is, the region-growing procedure for aggregation of areas in a planar subdivision. It summarizes the results of three publications (Haunert & Sester, 2005; Haunert, 2006; Haunert *et al.*, 2006). The presented method yields the same result as Algorithm 1, but allows updates to be efficiently propagated. This means that, if the large-scale map changes and we previously aggregated certain areas of the map, we do not need to re-generalize the whole map. Instead, we can retain most parts of the previously derived small-scale map without changes. The method copes with a general definition of compatibility of two adjacent areas, which subsumes existing definitions based on classes, sizes and lengths of common boundaries.

In Sect. 2.7.2 we argued that the region-growing procedure does not allow high-quality results to be produced. Nevertheless, due to the popularity and frequent application of the algorithm, the incremental version is an important achievement. In some applications the quality of the generalized map is not the primary concern. For example, if we want to support a method for progressive data submission, we might favor the algorithm, as it defines a gradual reduction of detail. The incremental approach can be used to update data structures that are designed for this problem, for example, the tGAP data structure of van Oosterom (2005).

Later in this thesis, we will concentrate on the problem of how to create aggregation results of high quality. The approach by combinatorial optimization that will be developed is fundamentally different to the iterative region-growing procedure. It is important not to extend the results of this chapter to the combinatorial optimization approach: applying the incremental algorithm does not generally yield the same result as re-generalizing the updated map by optimization. We will discuss in Sect. 7.6 how to also deal with updates in the optimization approach.

The chapter first explains requirements that need to be satisfied to allow for the developed incremental approach (Sect. 4.1). Section 4.2 introduces the new algorithm. Section 4.3 presents results that were obtained with this method, applying a definition of compatibility according to Podrenek (2002). Finally, Sect. 4.4 concludes with general remarks on incremental generalization.

4.1 Requirements for an incremental approach

To develop an incremental version of Algorithm 1, we need to insist on several basic requirements. This includes the following specifications of the 'smallest area' and the 'most compatible neighbor':

- The 'smallest area' is defined according to a strict total order '$<$' on the set of input areas and their potential aggregates. This implies that, given any two areas a and b, we can unambiguously decide whether $a < b$ or $b < a$.
- The compatibility of area a to its neighbors $N(a)$ is given with a strict total order '$\prec_a$' on $N(a)$. This means that, given two neighbors $b, c \in N(a)$, we can unambiguously decide whether b is more compatible with a than c. This case can be expressed by $c \prec_a b$.

Obviously, without ensuring that the 'smallest area' and the 'most compatible neighbor' are well-defined, we cannot even ensure that Algorithm 1 produces the same results when applied twice to the same input map. For example, if we select the areas according to their sizes, we need to consider the unlikely case that two areas have exactly the same size. For this purpose, it is useful to define an ID for each area. If we cannot decide according to the size which area to select first, we can simply select the one with the smaller ID. In any case, the orders $\prec_a$ and $<$ that are applied by Algorithm 1 need to be known. The incremental method uses the same criteria to select and merge objects.

In order to ensure that an update has a limited effect on the target dataset, we add two requirements:

- If we change another area c, this must not influence whether $a < b$ or $b < a$.
- If we change another area d, this must not influence whether $b \prec_a c$ or $c \prec_a b$.

If we define how to resolve ambiguous cases, then our model allows different methods to be subsumed, for example, the method of van Oosterom (1995) who proposes to use an importance function and a collapse function that yields local compatibility measures. Selecting a neighbor according to class similarities or boundary lengths as proposed by Cheng & Li (2006) also satisfies the requirements.

However, we can imagine a setting that does not satisfy the requirements: Let's assume that we defined that an area is preferably merged with the most frequent class in the dataset. If we change the class of area d, the new class may globally become the most frequent one. This can imply that, instead of c, area b becomes the most compatible neighbor of a. In other words, $b \prec_a c$ changes to $c \prec_a b$, which does not satisfy the last requirement. Criteria for iterative merging that are based on a global analysis of the dataset are proposed by van Smaalen (2003). Obviously, under such conditions, it would be extremely difficult to define an incremental method that offers the same result as a complete processing of the dataset. However, if the compatibility is defined according to rules that depend on global properties of the dataset, it would be reasonable to update the rules only after large changes. If we do not change the rules that define the compatibility and if the compatibility only depends on local properties, then we can apply the incremental approach.

4.2 An incremental aggregation algorithm

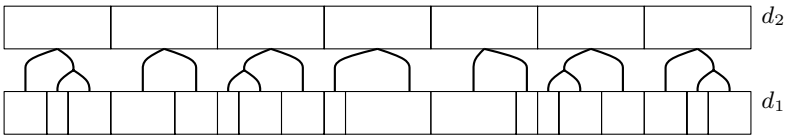
Let's assume that we are given two datasets: a planar subdivision of areas at large scale (dataset d_1) and a generalized planar subdivision (dataset d_2) that was obtained by applying Algorithm 1 to d_1, see Fig. 4.1(a). In each iteration, the area with minimum size was selected and merged with its most compatible neighbor. If we observe a change in the physical world we first update d_1, which yields dataset d_1^*. Generally, this can include insertions and deletions of objects as well as changes of geometries and classes. In any case we want to propagate the updates to the smaller scale. We could apply Algorithm 1 again to the whole large scale map, which yields the updated generalized dataset d_2^* in Fig. 4.1(b). However, we should avoid this approach if we wish to keep the processing time short. Presumably, the influence of the update is restricted to a certain region. Therefore, we can define a more efficient, incremental version of Algorithm 1.

In order to do so, we define two regions of influence (see Fig. 4.2):

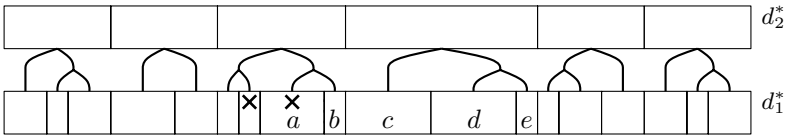
- The *interior region* R_1 is the union of areas in d_2 that have an outdated component in d_1, for example, in the more recent dataset d_1^*, a component's class was changed.
- The *boundary region* R_2 is the union of areas in d_2 that share a boundary with the interior region but do not belong to the interior region.

Algorithm 4 defines the incremental approach. Let's first consider the case that the lines (7)–(21) are ignored. In this case, the algorithm performs exactly the same steps as Algorithm 1 when applied to the areas in the region $R_1 \cup R_2$. We now explain that this approach does not guarantee the same result as a complete processing.

Applying Algorithm 1 only to $R_1 \cup R_2$ while keeping the rest of the map unchanged, we can obtain results that differ from the result of a complete processing. For example, area d in Fig. 4.1(b) is not contained in $R_1 \cup R_2$; nevertheless, it is influenced by the update: c and d are merged. This, however, is only possible, as an area



(a) two datasets of different scales before update: dataset d_2 was derived with Algorithm 1 from dataset d_1



(b) two areas changed (marked with X), which yields the updated input map d_1^*; applying Algorithm 1 to the complete dataset d_1^* yields d_2^*

Fig. 4.1: Influence of an update to the result of Algorithm 1. The bold lines define trees, which reflect how the areas are merged. Due to the update of area a, area c is no longer the most compatible neighbor of b in dataset d_1^*. Hence, b is merged with a and not with c. As a consequence, c does not reach the required size for the target scale and needs to be merged with its right neighbor, that is, area d.

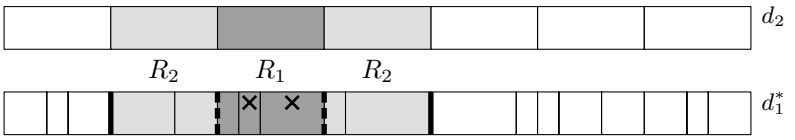


Fig. 4.2: Definition of influence regions R_1 (interior region) and R_2 (boundary region) in Algorithm 4. Both regions are shown in the dataset d_1^* (bottom) and the dataset d_2 (top).

Algorithm 4 Iterative aggregation in case of update (incremental version of Algorithm 1)

$R_1 \leftarrow$ union of areas in d_2 whose components have changed *//Interior region*
$R_2 \leftarrow$ union of areas in d_2 that are adjacent to R_1 but not contained in R_1 *//Boundary region*

$u \leftarrow$ smallest area of d_1^* in region $R_1 \cup R_2$ below threshold for target scale
while u is not Null **do**
    $v \leftarrow$ most compatible neighbor of u in $R_1 \cup R_2$
    merge u with v

    if (u in R_1 **and** v in R_2) **or** (u in R_2 **and** v in R_1) **then** *//Separation of R_1 and R_2 violated*
        if u in R_2 **then**
            $w \leftarrow$ area in dataset d_2 containing u
        else
            $w \leftarrow$ area in dataset d_2 containing v
        end if
        $R_3 \leftarrow$ union of neighbors of w in d_2 that are not contained in $R_1 \cup R_2$
        $t \leftarrow$ smallest area of d_1^* in R_3 below threshold for target scale
        while t is not Null **and** $t < u$ **do**
            merge t with most compatible neighbor in R_3
            $t \leftarrow$ smallest area of d_1^* in region R_3 below threshold for target scale
        end while
        $R_1 \leftarrow R_1 \cup \{w\}$ *//Expand interior region*
        $R_2 \leftarrow (R_2 \setminus \{w\}) \cup R_3$ *//Expand boundary region*
    end if

    $u \leftarrow$ smallest area of d_1^* in region $R_1 \cup R_2$ below threshold for target scale
end while

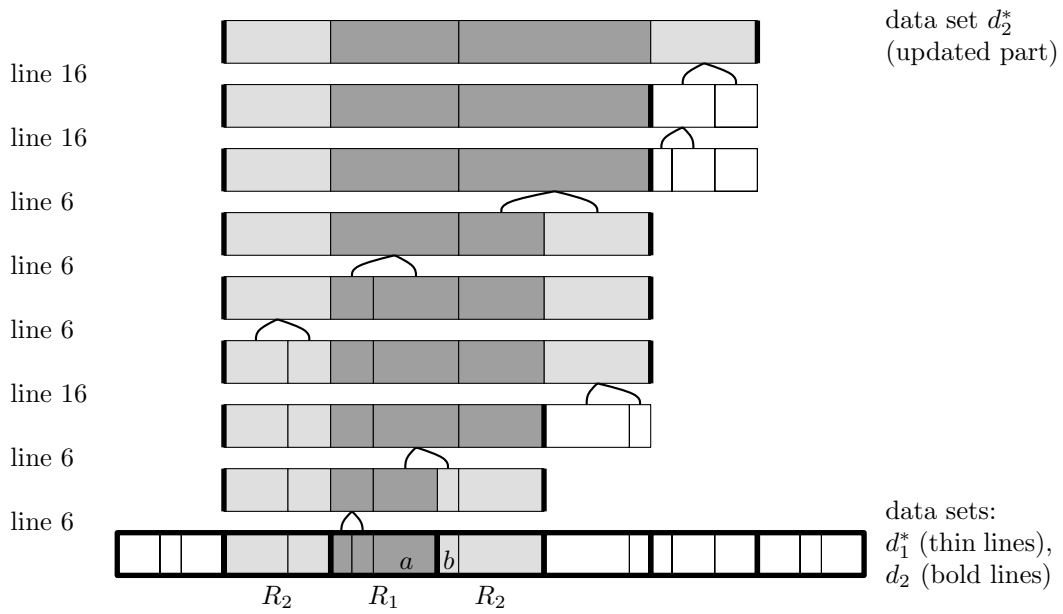


Fig. 4.3: Different steps of Algorithm 4 (from bottom to top). Each merge is labeled with the line number in the algorithm that defines the merge. The algorithm starts to aggregate areas of dataset d_1^* in the same way as Algorithm 1. However, instead of considering all areas, only the areas in the regions R_1 and R_2 are considered. This restriction is valid, until an area from R_1 is merged with an area of R_2. In the second step of the example, area b (contained in R_2) is merged with area a (contained in R_1). Consequently, the regions need to be expanded. The result equals the result of Algorithm 1, but only a subset of areas is evaluated.

from R_1 was earlier merged with an area from R_2 (b was merged with a). In other words, the update in the interior region R_1 first needs to influence the merges in the boundary region R_2 before it can affect how the areas in the exterior of $R_1 \cup R_2$ are merged. Hence, we need to expand the regions, whenever we observe that an area from R_1 is merged with an area from R_2.

The lines (7)–(21) handle this case, that is, a merge violates the separation of R_1 and R_2. We can say with certainty that, before this merge happens, all areas outside of $R_1 \cup R_2$ are merged in the same way as before the update. This is assured by our requirements in Sect. 4.1: a change of an area can influence how its neighbors are merged. Other areas can only become affected if their neighbors were affected earlier.

If we observe that a merge of two areas u and v violates the separation of R_1 and R_2, we cannot guarantee anymore that the areas outside of $R_1 \cup R_2$ are merged in the same way as before the update. To handle this case, we first select the composite area w in region R_2 that was affected by this merge (lines (8)–(12)). We will need to expand the region R_1 to w and the region R_2 to the neighbors of w, which define the region R_3 (line (13)). Before we continue to merge the areas in $R_1 \cup R_2$, we need to make sure that the areas in the added region satisfy the size threshold for the current aggregation level. This means, we need to catch up with merges in R_3 that, when applying Algorithm 1, would happen earlier than the merge of u and v. This is performed in an interior loop (lines (15)–(18)). Figure 4.3 illustrates this process with an example. We needed to expand the regions R_1 and R_2 two times. However, we still needed to process only a part of the whole dataset.

Obviously, we can construct examples with small updates that imply large changes of the generalized map. In theory, Algorithm 4 can require the same computational effort as Algorithm 1, that is, the final extent of the region R_1 can cover the whole dataset. Generally, there are $\mathcal{O}(n)$ merges to perform, with n being the number of areas in the input. In each step, the most compatible neighbor of a selected area needs to be found, which takes $\mathcal{O}(n)$ time, if the neighbors are explicitly enumerated. Hence, both algorithms require $\mathcal{O}(n^2)$ time. Nevertheless, in practice the set of influenced areas is hopefully small if we only apply small updates.

Finally, we explain a small improvement of Algorithm 4 that can be applied if the history of merges is stored in a database. More precisely, if the tree in Fig. 4.1(a) is explicitly given. In this case, we can replace the interior loop (lines (15)–(18)) with a select statement that returns the areas in the added region R_3 at the current level of aggregation. For example, this is useful for efficiently updating the tGAP structure that is used for progressive submission of vector data.

4.3 Experiments with the incremental aggregation method

We applied the iterative algorithms for area aggregation to an existing dataset of the ATKIS DLM 50. We defined the importance of an area according to its size and applied size thresholds according to existing specifications of the ATKIS DLM 250 (AdV, 2003). In this experiment, we defined the compatibility of neighbors based on classes, using a priority list for class changes; this approach was proposed by Podrenek (2002). The priority list for the class 'open forest' defines the following rules:

- An open forest area is preferably merged with another open forest area.
- If there is no such neighbor, then it is merged with a forest area.
- If there is neither an adjacent open forest area nor an adjacent forest area, then it is merged with a grassland area (and so on).
- In the case that there are several neighbors of the most compatible class, the selected area is merged with the largest among them.

Remaining ambiguities are resolved by selecting the area with the smallest ID. However, it is very unlikely that there are two areas of exactly the same size. We do not further discuss this set-up in detail, as the next chapter introduces a new approach that applies a global measure of semantic accuracy instead of a priority list for merges. At this point, it is sufficient to see that the approach of Podrenek (2002) allows our incremental method to be applied: the priority list defines the order $\prec_a$.

Figure 4.4 shows a sample that was processed with the developed method. To find this illustrative example we tried to achieve a maximum impact in the target scale with a small change in the source dataset. Figure 4.4(a) shows the sequence of merges that are applied by Algorithm 1, which yields the result in Fig. 4.4(b). The same is shown for the dataset after adding a hypothetical update (Figures 4.4(c) and 4.4(d)). In this example, a small grassland area changed to open forest. Consequently, before and after update, the same area is merged with different neighbors (step 2). Merging the area to a forest in the updated map implies that the forest to its right side gets a neighbor of the same class. Hence, also this area is handled differently (step 3). In the fourth step of both examples, the same forest area is merged to different neighbors. In both cases, the largest neighbor of class 'farmland' was chosen. The example shows that a small change in the input map can indeed imply large changes in the result: only changing the class of a small area caused four composite regions to change their shapes. However, changes of similar extent did usually not influence more than two composite areas.

(a) dataset d_1: ATKIS DLM 50 before update

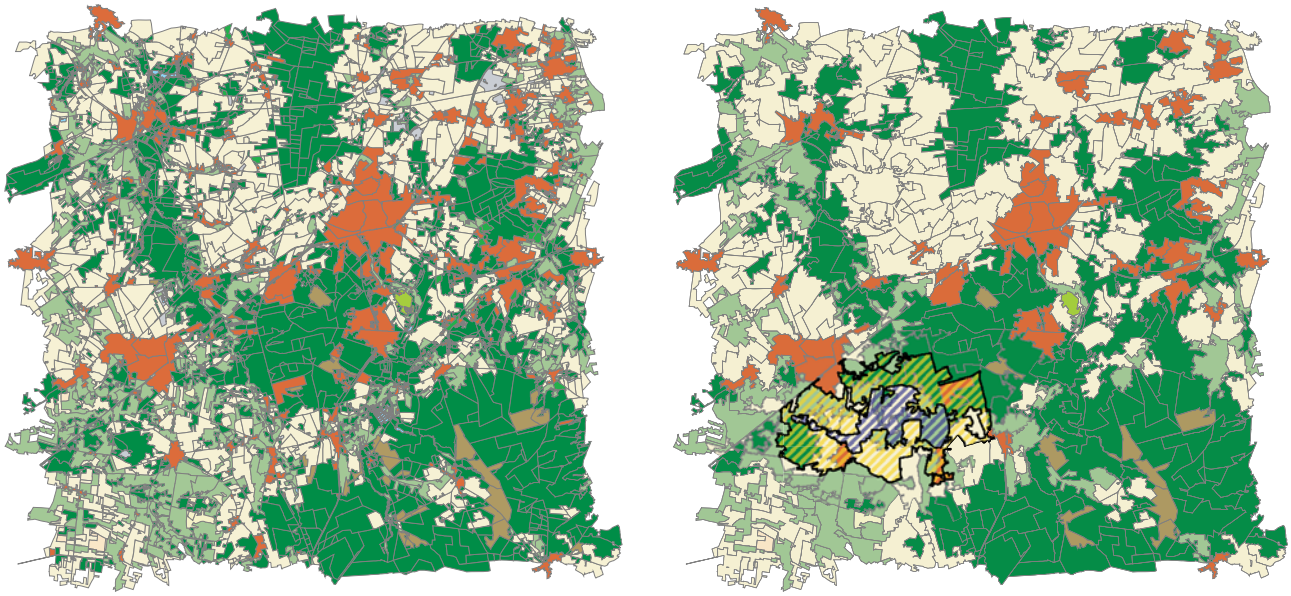
(b) dataset d_2: result of Algorithm 1 when applied to d_1

(c) dataset d_1^*: ATKIS DLM 50 after update

(d) dataset d_2^*: result of Algorithm 1 when applied to d_1^*

Fig. 4.4: Influence of an update to the result of Algorithm 1. A grassland area in d_1 (bold outline) changed to open forest in d_1^*. The blue arrows display merges that are applied by Algorithm 1 in increasing order of numbers (not all merges shown). The result satisfies given size thresholds for the ATKIS DLM 250.

The significance of the incremental algorithm becomes apparent with Fig. 4.5, which shows a dataset that corresponds to a sheet of the German topographic map at scale 1:50 000. Figure 4.5(a) shows the input data set (ATKIS DLM 50) and Fig. 4.5(b) shows the result of the iterative procedure together with the final influence regions R_1 and R_2 that resulted from the update in Fig. 4.4. Together, both regions contain 348 areas of the ATKIS DLM 50. Only these areas were evaluated by Algorithm 4. The whole dataset contains 5461 areas. Hence, though the update had a relatively high influence, only 6% of the map needed to be processed. In particular, the result allows an assumption to be made that processing updates of limited size takes constant time in practice.



(a) a dataset of the ATKIS DLM 50 that corresponds to the map sheet "Buchholz in der Nordheide" of the topographic map at scale 1:50 000; the dataset has an extent of 22 km × 22 km and contains 5461 areas.

(b) a dataset that was derived with Algorithm 1 from the ATKIS DLM 50, applying specifications of the ATKIS DLM 250; the update in Fig. 4.4 is propagated with Algorithm 4, yielding the influence regions R_1 (blue hatched) and R_2 (yellow hatched)

Fig. 4.5: A worked example of the incremental area aggregation method (Algorithm 4). Applying the update in Fig. 4.4 to the source map (a) implies that four composite areas change in the target scale (b). The resulting influence regions R_1 and R_2 are shown at their final extents. The region R_1 contains all composite areas that were actually affected by the update, that is, it contains four areas in the example. The region R_2 contains 21 additional composite areas that were considered as potentially affected. Algorithm 4 only processed the areas in R_1 and R_2; other areas were not evaluated.

4.4 Concluding remarks

The definition of an incremental version of the iterative area aggregation algorithm is an important achievement. Since it guarantees the same result as the original algorithm, the result is independent of the order of updates. Though an update of the large-scale map can imply that the whole small-scale map needs to be changed, we argued that there is reasonable hope for an influence region of limited extent. In this case, the incremental algorithm allows the map to be updated without considering the whole dataset. This is of particular importance if large databases are to be updated, for example, digital landscape models of a national extent. In practice we can expect a constant-time performance with the incremental method. However, tests with realistic updates are needed to find out the usual running time of the method, for example, updates that occur when a new road is built.

An interesting question is whether the proposed approach can be generalized, in order to also develop other incremental generalization algorithms. Of course, Algorithm 4 is designed only for the aggregation of areas.

However, the definition of two zones, that is, an interior region that contains obviously influenced objects and a boundary region that contains potentially influenced objects could be a useful idea for other problems. Probably, for other generalization operators, we cannot achieve an incremental method that guarantees the same result as the global method. However, this requirement could be relaxed: often we would agree with a result that is of a similar quality to the output of a global method. With this concession, we can imagine several applications of our two-zone approach.

Let's consider a displacement operator that aims to satisfy proximity constraints with minimum movements of objects, for example, the distance between two buildings must exceed a given threshold. If we modify a set of buildings, other buildings may need to move to satisfy the constraint. Again, these movements can imply conflicts with other buildings, which might cause the whole map to change. In this case, it is certainly reasonable to first operate with the set of modified buildings and the buildings in a certain surrounding. As long as we do not observe an influence to the buildings in this surrounding, we may assume that also other buildings are not affected. Otherwise, we could simply restart the method with a larger set of buildings. Defining such an incremental method requires appropriate influence regions to be defined, which depends on the specific generalization algorithm that is to be applied. In order to define appropriate influence regions we could employ the partition defined by roads, rivers, and railways, which is proposed by Timpf (1998).

If we require that the result of an incremental method is to be of similar quality to the result of an existing global method, we need to answer the following question: How can we compare the results of two generalization methods with respect to their quality? We could answer this question if we had quantitative measures that express the quality of a generalized dataset. We introduce a quantitative measure of semantic accuracy in the next chapter. This will be applied as cost function in an optimization approach to find a solution of maximum quality. However, we could apply the same measure to compare the results that were obtained with different generalization methods.

Chapter 5

A Generalization Workflow Driven by Data Quality

To formalize the requirements of the generalization problem under examination, we discuss the relevant elements of spatial data quality. We assume that all constraints in generalization are needed to ensure quality. A similar view was chosen by Jaakkola (1997b,c) to introduce requirements for the generalization of land cover data in the raster model. However, Jaakkola did not apply optimization techniques in his generalization system. A summary of this discussion on relevant quality aspects has been published (Haunert & Sester, 2008b). This chapter explains general notions and quality criteria for the generalization of areas in a planar subdivision and presents a workflow comprising collapse, aggregation, and line simplification.

5.1 Quality of spatial data

According to Morrison (1995) there are seven elements of spatial data quality: lineage, positional accuracy, attribute accuracy, completeness, logical consistency, semantic accuracy, and temporal information. Most of these are affected by map generalization, for example, when applying displacement or simplification algorithms to lines, their positional accuracy is reduced. Selection of objects affects completeness. Concerning the generalization of areas in a planar subdivision, we especially need to consider logical consistency and semantic accuracy. We will discuss these elements in detail in Sect. 5.2 and Sect. 5.3, respectively. First, however, let's discuss the other elements of data quality and possible implications for generalization.

Lineage is meta-information about the history of a dataset, for example, concerning its source and transformations that have been applied. For map generalization this means that we should provide information about the algorithms that have been applied and their parameters together with the generalized data. However, this also has implications on how to generalize. We can conclude that we should favor deterministic approaches, which allow a dataset to be reproduced, if the set of parameters is known. Otherwise the content of a dataset will be influenced by chance. The reason for changes between the small-scale map and the large-scale map will be hard to understand, even if information about the algorithms that have been applied is provided.

Positional accuracy is most often defined by standard deviations of coordinates. Ensuring positional accuracy in map generalization is an important problem, as coordinates of points can change, for example by displacement. We can extend positional accuracy to a *geometrical accuracy*, which also comprises accuracy of angles and distances. The objective for a minimal standard deviation leads to the optimization approach by least squares adjustment, which we reviewed in Sect. 2.5.1. We do not approach the displacement of objects, but consider the simplification of lines, which also affects the positional accuracy. A common approach is to introduce a maximal allowed error, which means that the old line needs to be within a defined buffer of the generalized line. This is a hard constraint in the optimization approach of de Berg *et al.* (1998) for the simplification of lines that define a planar subdivision. We review and extend this method in Sect. 6.2.

Attribute accuracy can be an important objective for map generalization. When aggregating objects with different values for the same quantitative attribute, we probably would like to minimize the mean of the differences or the mean of the squared differences between the original values and the value for the composite object. Similarly,

the accuracy of a qualitative attribute could be measured based on semantic distances to the original attribute values. Section 5.3 explains a measure based on semantic distances between land cover classes. Though this is seen in the context of semantic accuracy, the accuracy of attributes could be defined in a straightforward way. An aggregate can be composed of objects that belong to different classes with different attribute definitions. However, for areas of different land cover classes, we require that an aggregate at least partly corresponds to a large-scale object of the same class. Otherwise it is questionable whether attribute values for the small-scale object can reasonably be defined at all.

Completeness could be defined for a data model (to what degree is the reality reflected by the model?) and for the data itself (is the data complete with respect to the model?). Usually the data model for the target scale is given in map generalization. However, we need to take care of the completeness of data, for example, when applying a selection operator. Excerpts from typical data models were shown in Table 2.2 (page 18). We interpreted the selection criteria as a prohibition to keep small areas in the target scale. Additionally, we can understand the given thresholds as instruction to keep areas that have sufficient size, meaning that their classes must not be changed. As we will see in Sect. 7.6.2, the proposed approach allows hard constraints to be introduced that ensure completeness in this sense. However, the basic approach is to define soft constraints that hinder the changes of classes. This is discussed in the context of semantic accuracy in Sect. 5.3.

Temporal information is needed to inform users about the currency of the dataset. This element of quality is related to the updating task from Sect. 2.8, which stated that the currency of all layers in an MRDB should be the same. If we additionally assume that each object has an individual time stamp that defines the date of its acquisition, we need to consider how to represent this information in the generalized map. Normally, this does not have any consequence on how to generalize, but we can define the currency of a composite object simply according to its most dated component.

5.2 Logical consistency

Kainz (1995) defines logical consistency as follows:

> "A spatial dataset is said to be logically consistent when it complies with the structural characteristics of the selected data model and when it is compatible with the attribute constraints defined for the set."

We first select an appropriate data model and then discuss the implications for generalization.

5.2.1 Selected data model

A planar subdivision is a suitable and common data model to represent land cover in topographic databases. This is defined as follows:

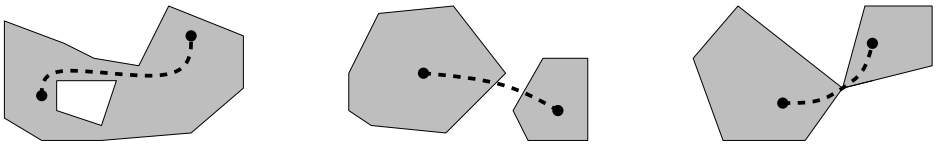
- A *planar subdivision* is an exhaustive partition of the plane into non-overlapping regions.

Though the extent of a spatial dataset is limited, we can define an exterior region to satisfy the requirement of being exhaustive. The specifications of the German ATKIS datasets allow a few exceptions from the definition of a planar subdivision. For example, the space between a pair of carriageways is an undefined area. If a pond lies within a park, the polygons representing both features can intersect. We ignore these exceptions in the sequence to find a common basis. Generally, we can define a default class to fill up holes and new classes, if areas of different classes overlap. For example, if areas of classes 'park' and 'lake' overlap, their intersection is defined as a new area of class 'park_lake'.

The German ATKIS specifications allow multi-part shapes in special cases. Again we neglect these exceptions and claim that regions need to be contiguous, which is defined as follows:

- A region is said to be *contiguous*, if it contains a path between each two interior points.

Figure 5.1 illustrates this definition. Figure 5.1(c) shows a region of two parts that only touch each other in a common point. Each path between points in different parts intersects the boundary of the region. According to the definition of containment by Egenhofer (1989), such a path is not contained in the region. Consequently the region is not contiguous.



(a) a contiguous region (b) a non-contiguous region (c) a non-contiguous region

Fig. 5.1: Definition of contiguity using a path between two interior points.

In a topographic database each region of the planar subdivision belongs to a single land cover category (or class). In the following discussion we denote the set of classes by Γ. We assume that this is the same in both the input map and the output map. In addition to the structural characteristics of the data model, the definition of Kainz (1995) claims to satisfy attribute constraints. This includes any hard size constraints defined in the database specifications (recall Table 2.2). We simply define area thresholds θ as a function of the category, that is $\theta : \Gamma \to \mathbb{R}^+$. Note that without the requirement for contiguity, the definition of a minimal object size becomes almost meaningless. An object could have sufficient size even in the case that each component part is smaller than the allowed threshold. These parts could be scattered over the whole dataset; of course, this is not intended by generalization.

Finally, we need to discuss two different types of maps with areas of different categories: categorical coverages and choropleth maps. Their distinction can be summarized as follows (Frank *et al.*, 1997):

- A *categorical coverage* is a planar subdivision solely defined by categories of a theme over the plane. A category is given for each point, then contiguous regions are defined to be as large as possible under the restriction that each region contains points of only one category.
- A *choropleth map* is a planar subdivision whose regions are mapped on the categories of a theme. The regions are given mapping units, for example, political districts. These are assigned to categories.

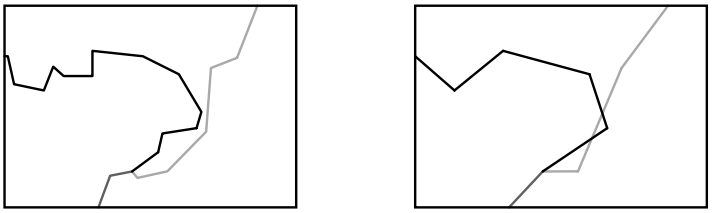
Galanda (2003) points out that categorical coverages are commonly used in topographic databases for the representation of land cover classes. However, if we inspect the sample in Fig. 1.1, we observe adjacent areas having the same class. By definition, these do not exist in a categorical coverage. The additional boundaries are useful, as they represent entities in the physical world, for example, roads. Consequently, each area is a reasonably defined object that has a relatively compact shape. Similar conditions were found in a digital dataset of OS MasterMap from Great Britain: according to Revell (2007) regions of the same class are not merged if they are separated by a river or a fence. In order not to loose the information given by additional boundaries, we do not insist on the strict definition of a categorical coverage.

The developed methods were not tested for a dataset that satisfies the strict definition of a categorical coverage. However, if the input meets this definition, we probably can produce conditions that are similar to our setting by splitting polygons at roads. As we can apply the developed method for both kinds of maps, that is, categorical coverages and choropleth maps, we only assume that the input dataset is a planar subdivision, each of its regions belonging to a single class. The same characteristic is required for the target dataset, in which each region must satisfy the size threshold defined for its class.

5.2.2 Implications for generalization

When generalizing topographic information in the defined data model we need to take care that the subdivision remains exhaustive. This implies that we cannot apply a selection operator without refilling the free space that is left by eliminated objects. Secondly, we need to ensure that the regions of the subdivision do not intersect. This problem occurs when applying standard line simplification routines, for example the algorithm of Douglas & Peucker (1973). Figure 5.2 shows an example with resulting intersections. More advanced line simplification methods exist, which ensure that no intersections occur in the generalized map. We review and extend the approach of de Berg *et al.* (1998) in Sect. 6.2.

With the requirement for simplicity of the subdivision, it is also clear that a certain area aggregation approach does not work: we cannot solve the problem independently for objects of different classes. For example, if we independently group forest areas and settlement areas, we will end up with intersections between forests and settlements. Therefore, the areas in the subdivision need to be handled in a more comprehensive approach.



(a) original map (b) map after line simplification

Fig. 5.2: Line simplification violating simplicity of the subdivision.

As we will see in Sect. 7.4, it is not trivial to express the requirement for contiguous regions in an optimization approach. A graph-theoretic, general solution to this problem is presented. However, a stricter definition of contiguity is introduced and applied to obtain an acceptable performance.

To satisfy the hard size constraints, we could apply existing methods, for example Algorithm 1, which iteratively merges areas. Alternatively, we could apply a collapse routine that divides an area among its neighbors, for example the method proposed by Bader & Weibel (1997). A similar method is presented in Sect. 6.1. Though these methods produce logically consistent results, we should define some additional aims for generalization before deciding on a certain solution. More precisely, the requirement for logical consistency defines hard constraints, but the formalization of soft constraints is still needed to approach the problem by optimization. As discussed in Sect. 3.7.3, the satisfaction of hard constraints is problematic with most optimization approaches in map generalization. This, however, is essential to produce logically consistent results.

5.3 Semantic accuracy

Semantic accuracy is an essential element of quality. It ensures that the meaning and function of geographic objects is correctly reflected in the dataset. Salgé (1995) gives the following definition:

> "The purpose of Semantic Accuracy is to describe the semantic distance between geographical objects and the perceived reality."

This definition is very general, but we can take the idea of a semantic distance literally: we introduce a distance function to measure semantic changes caused by generalization. This approach is similar to those of Bard (2004) and Frank & Ester (2006) who generally propose measures to evaluate the similarity of the input map and the output map. In other words, the input map is considered to be a sufficiently good reference of the "perceived reality". As semantics of objects are mainly given by class memberships, we introduce a semantic distance between classes in Sect. 5.3.1. Additionally we consider the shapes of objects in Sect. 5.3.2.

5.3.1 Semantic distance between classes

We use 'semantic distance' synonymously with 'semantic dissimilarity'. Consequently, we could derive semantic distances from semantic similarity measures, which have been proposed by different authors. Yaolin *et al.* (2002) introduce a symmetric semantic similarity matrix to compare object types of areas before and after reclassification in generalization. Rodríguez & Egenhofer (2004) propose an asymmetric similarity measure for classes of geographic objects. In both approaches, semantic similarity values $\rho : \Gamma^2 \rightarrow [0, 1]$ for classes are automatically derived from the given data model, taking class hierarchies into account and comparing attribute definitions. Ahlquist (2005) uses a similarity measure based on fuzzy membership functions to assess land cover changes over time. Schwering (2008) reviews different approaches for defining semantic similarity measures. She also discusses whether a semantic distance should have the properties of a metric (identity, symmetry, and triangle inequality) and concludes that the definition of a metric would be too restrictive. We adhere to this position, which we will support with examples from the context of map generalization later in this section.

We formally define the function $d : \Gamma^2 \rightarrow \mathbb{R}_0^+$ as the semantic distance between classes in the set Γ. Our only assumption about this function is that $d(\gamma_1, \gamma_2) = 0$ if $\gamma_1 = \gamma_2$. A high value $d(\gamma_1, \gamma_2)$ means that the classes

$\gamma_1 \in \Gamma$ and $\gamma_2 \in \Gamma$ are dissimilar, which for example applies to settlement and grassland. Instead, farmland and grassland are semantically close as they are both classes of vegetation. Since we aim for semantic accuracy in map generalization, we would rather accept a change of a grassland area into farmland than into settlement. In order to define a global measure of semantic accuracy, it is most natural to associate a cost with such basic change operations. We define this cost to be proportional to the size of the area that changed its class and the distance between the classes before and after generalization. The *total cost for class change* f_{class} is the sum of these cost terms:

$$f_{\text{class}} = \sum_{v \in V} w(v) \cdot d(\gamma(v), \gamma'(v)) \,, \tag{5.1}$$

with V being the set that contains an element for each area in the overlay of the input map and the generalized map, $w : V \to \mathbb{R}^+$ being their sizes, $\gamma : V \to \Gamma$ being their classes in the input map, and $\gamma' : V \to \Gamma$ being their classes in the generalized map. Our aim is to minimize the cost f_{class}. Dividing f_{class} by the size of the dataset, we obtain the *average class distance*:

$$\bar{d} = \frac{\sum_{v \in V} w(v) \cdot d(\gamma(v), \gamma'(v))}{\sum_{v \in V} w(v)} \,. \tag{5.2}$$

This measure is useful since it allows the following interpretation without knowing the size of the dataset: Let's assume that we obtain $\bar{d} = d_0$ for our generalized dataset. We can say that our dataset is of the same quality as a hypothetical dataset, each of its areas having a class at semantic distance d_0 from the original class.

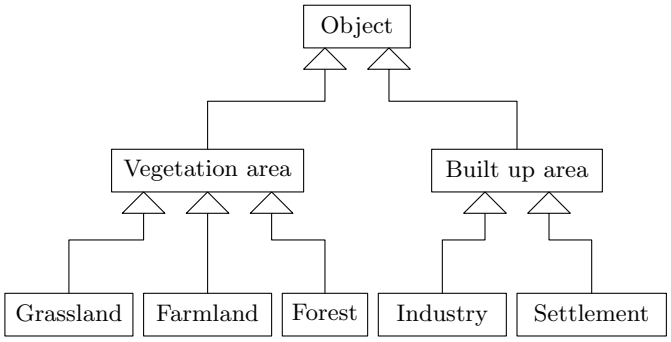
Allowing d to be asymmetric or to violate the triangle inequality can be useful in several applications: in map generalization we often want to exaggerate objects of important or infrequent classes. We can express this objective by defining a long distance from an important class to a less important one and a short distance for the opposite direction. This would imply that changes of important classes into unimportant classes result in a relatively low semantic accuracy compared to changes in the other direction. Consequently, we do not insist on symmetric semantic distances. Furthermore, for example we could imagine a map of three classes: 'playground', 'traffic' and a class 'default'. Confusing any class with the default class (or vice versa) might be unproblematic, thus we would define a relatively short semantic distance. However, 'playground' and 'traffic' should not be mixed up, thus we would define a very long distance in between. In this example, the triangle inequality does not hold, since $d(\text{playground}, \text{default}) + d(\text{default}, \text{traffic})$ may be less than $d(\text{playground}, \text{traffic})$.

Let's assume that for each area $v \in V$ we are given a quantitative attribute. This case can be subsumed by our model, for example, by defining $\Gamma \subset \mathbb{Z}$. In this case, we can define $d(\gamma(v), \gamma'(v)) = |\gamma(v) - \gamma'(v)|$, which means that we wish to minimize the average difference between attributes of the input map and the generalized map. However, we can also define $d(\gamma(v), \gamma'(v)) = (\gamma(v) - \gamma'(v))^2$ in order to charge higher costs for large differences. Hence, the definition of d allows different definitions to be subsumed. However, in the case that Γ contains qualitative classes, the distances d cannot be defined by differences of quantitative values. We now discuss how to find an appropriate setting for d in practice.

Of course the class distance values could be directly defined by experts. This, however, requires values to be specified in a matrix of $|\Gamma| \times |\Gamma|$ elements, which can be difficult, even for experts. Alternatively, the values of d could be derived from a given data model, for example, using the method of Rodríguez & Egenhofer (2004). The obtained similarity values could be translated into distances, for example, by defining $d = 1/\rho - 1$. This yields the distance $d = 0$ if $\rho = 1$ (perfect similarity). If ρ approaches zero, the distance d approaches positive infinity. The special case $\rho = 0$ (no similarity at all) can be treated as a hard constraint, that is, class $\gamma_1 \in \Gamma$ must never change to $\gamma_2 \in \Gamma$. The method yields objective class distance values. However, it requires a rich data model and a detailed class hierarchy to accurately distinguish different classes. Because of this, the following mixed approach is chosen, which exploits class hierarchies in a given data model, but also requires input of expert knowledge.

Figure 5.3(a) shows a class hierarchy from the ATKIS specifications. We consider the diagram as a graph whose nodes correspond to classes and whose edges correspond to links, that is, class inheritance relationships. Defining unit edge lengths, the distance value $d(\gamma_1, \gamma_2)$ can be defined as the length of the shortest path in this graph from $\gamma_1 \in \Gamma$ to $\gamma_2 \in \Gamma$. This method yields the class distances in Fig. 5.3(b). The approach is reasonable, since similar classes have common superclasses. Accordingly, we assign short distances to such pairs.

In order to further differentiate between different classes, experts can incorporate additional knowledge. In Fig. 5.4(a) we have introduced an additional class 'open vegetation' to express that farmland and grassland have

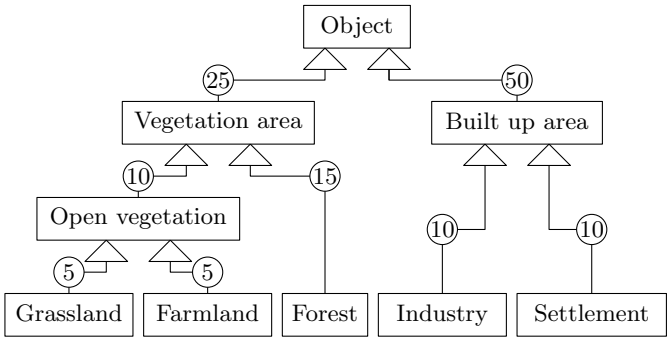


(a) class hierarchy defined in the ATKIS specifications

	Settlement	Industry	Farmland	Grassland	Forest
Settlement	0	2	4	4	4
Industry	2	0	4	4	4
Farmland	4	4	0	2	2
Grassland	4	4	2	0	2
Forest	4	4	2	2	0

(b) semantic distance matrix that can be derived from class hierarchies in (a)

Fig. 5.3: A class hierarchy that was given in the database specifications and a derived distance matrix.



(a) class hierarchy from Fig. 5.3(a) after augmentation with expert knowledge; numbers in circles refer to the degrees of specialization, which are exploited to calculate class distances

	Settlement	Industry	Farmland	Grassland	Forest
Settlement	0	20	100	100	100
Industry	20	0	100	100	100
Farmland	100	100	0	10	30
Grassland	100	100	10	0	30
Forest	100	100	30	30	0

(b) semantic distance matrix that can be derived from class hierarchy and degrees of specialization in (a); the results in this thesis were obtained using this setting

Fig. 5.4: A more sophisticated class hierarchy and the derived distance matrix.

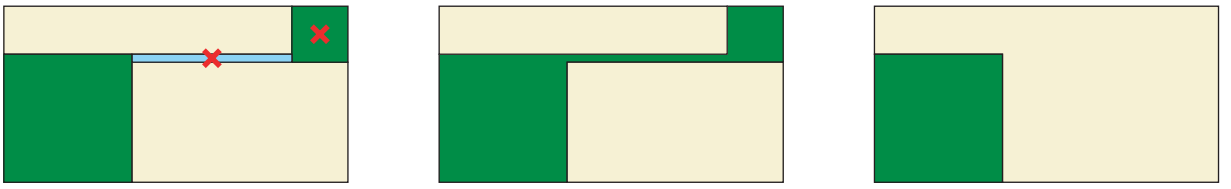
a common property that is not shared by forest. Additionally, non-uniform edge lengths have been defined; these express the degree of specialization between a superclass and a subclass. Again, we could exploit the data model to define these values, for example, we could set the edge length equal to the number of additional attributes in the subclass. However, in this example, the edge lengths were defined according to experiences. Figure 5.4(b) shows the resulting class distances. These were used for all our experiments.

Note that, when using the proposed approach to derive class distances, the function d will satisfy the definition of a metric. To model appropriate distances for the examples that motivated our more general definition of d, further tuning of the values in the obtained matrix is needed.

5.3.2 Semantic accuracy reflected by shape

In order to take semantic accuracy into account, we should not only focus on classes of objects but also consider their shapes. An area may belong to the class forest, but does it also have the shape of a forest? This question is indeed difficult to answer. In the quality assessment for generalization this issue is often considered by measures that compare a shape before and after generalization (Frank & Ester, 2006). This approach requires correspondences of cardinality 1:1, which generally do not exist in aggregation.

Figure 5.5(a) shows an input map including two areas that are smaller than the required threshold for the target scale. There are multiple logically consistent generalization results; two of them are shown in Figures 5.5(b) and (c). If we just evaluate class changes, we need to attest a high semantic accuracy for the result in (b). However, we might feel cheated, since the hard constraint for contiguous shapes was only satisfied with an extremely long and narrow bridge between both forest areas (green) of the input. To formally express what is wrong with this result, we need to penalize untypical shapes. As a general rule these are non-compact shapes.



(a) a hypothetical input map (b) a logically consistent result (c) another logically consistent result

Fig. 5.5: A planar subdivision that is to be generalized (a). Areas marked with a red X are too small for the target scale. The results obtained by aggregation in (b) and (c) are both feasible. Though the semantic class distance to the input map is smaller for the result in (b), we would prefer the more compact result in (c).

Compactness is related to the principles of Gestalt theory, which was introduced in Sect. 2.7.1: as proximity is an important criterion for perceptual grouping, different parts of a composite object should be close to a common center. Consequently, the compactness of a shape is most commonly measured by comparison of its properties with those of a circle. Measures for compactness are discussed in detail in Sect. 7.2. Generally, we need to charge a cost for each region in the output map. A region is defined by a subset of V and a class in Γ. Therefore, we measure the non-compactness of a region with a function $c : 2^V \times \Gamma \to \mathbb{R}_0^+$. The generalized map consists of multiple regions, that is, it is a partition of V into subsets $\{V_1, V_2, \ldots, V_k\}$ with classes $\gamma'_1, \gamma'_2, \ldots, \gamma'_k$, where k is the number of regions in the generalized map. The total cost for non-compactness becomes:

$$f_{\text{comp}} = \sum_{i=1}^{k} c(V_i, \gamma'_i)\,. \tag{5.3}$$

We do not consider the aim for compactness as a new element of data quality, but as another aspect of semantic accuracy. In fact, referring to the "perceived reality", the definition of semantic accuracy by Salgé (1995) suggests that principles of human perception should be comprised.

5.3.3 Implications for generalization

As discussed in Sect. 5.1, another aim of generalization is positional accuracy. If we require that the boundaries in the generalized map are a subset of the boundaries in the input map, the positional accuracy in the result will be perfect. Under this restriction, hard size constraints can only be satisfied by aggregation. The set V will contain an element for each area in the original map. Both remaining aims, class similarity and compactness, can be conflicting. We need to find a trade-off, which we express as a weighted sum. For this we introduce the weight factor $s \in [0, 1]$, which defines the generalization objective as follows:

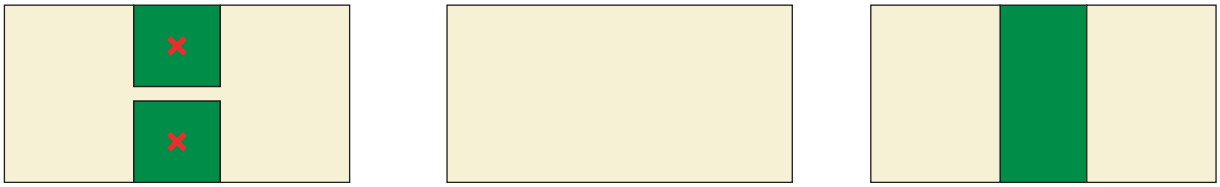
$$\text{minimize} \quad f = s \cdot f_{\text{class}} + (1 - s) \cdot f_{\text{comp}}\,. \tag{5.4}$$

With respect to this objective and with the restriction to the original boundaries, the aggregation method presented in Chapter 7 yields results of maximum semantic accuracy, subject to hard constraints that ensure logical consistency.

We can also apply the function f in Algorithm 1 (region-growing): given that a is the smallest area, we can define that the neighbor b is more compatible with a than the neighbor c, if merging a to b implies a lower increase of f than merging a to c. This means, the function f defines the order $\prec_a$ that we introduced in Sect. 4.1. Of course, the iterative approach does not produce an optimal solution. Nevertheless, this definition of compatibility is reasonable if we decide to apply Algorithm 1 instead of our new optimization approach.

Aggregation is the most important approach to satisfy size constraints. However, the assumption that generalization does not change region boundaries is too restrictive. Figure 5.6(a) shows an input map with two areas that fall below the required threshold. Figure 5.6(b) shows the semantically most accurate aggregation result, that is, a planar subdivision that is solely defined by a subset of the original boundaries.

We would prefer the map in Fig. 5.6(c), but this cannot be obtained by aggregation alone. The problem in this example is that the forest areas in the input are not adjacent. In order to combine them into a contiguous shape, the separating corridor needs to be collapsed first, such that the bone-shaped farmland area (light color)



(a) a hypothetical input map; areas marked with X are too small for the target scale

(b) a logically consistent result that can be obtained by aggregation

(c) a logically consistent result that cannot be obtained by aggregation

Fig. 5.6: A map that is to be generalized (a). The result (c) is semantically more accurate than (b).

is split into two parts. A similar problem appears in the example in Figures 1.1 and 1.2. New boundaries were introduced to allow a combination of the three settlement areas.

Generally, we could ask for the logically consistent planar subdivision that minimizes the objective (5.4). In this case we would need to allow new boundaries. However, to develop a comprehensive optimization approach for this general problem seems too ambitious. An approach to the problem with iterative meta-heuristics would imply the disadvantages discussed in Sect. 3.7. We cannot obtain optimal results and it would be difficult to generate near-optimal results subject to hard constraints. Therefore, we divide the generalization task into different parts, which we solve in succession.

5.4 Development of a generalization workflow

When applying the proposed generalization methods, the first step is to ensure our assumptions about the input data. Conflicts with the requirement for simplicity of the subdivision need to be resolved. Furthermore, objects need to be reclassified if classes in the input scale are subsumed by more general classes in the target scale. For example, objects of type deciduous forest and coniferous forest need to be reclassified into forest.

Before aggregating areas we need to remove disturbing details such as the narrow polygon part in Fig. 5.6(a). Otherwise, we cannot obtain the wanted result in Fig. 5.6(c). For this purpose Sect. 6.1 presents an approach based on skeletonization of areas. This method collapses narrow polygons and polygon parts and ensures that the subdivision remains simple. Thereafter, the aggregation of areas can be performed, assuming that all disturbing details were removed. Finally, we need to consider the simplification of lines, which is needed to adapt the map to the reduced level of detail. In contrast to collapse, line simplification does not change the topology of the subdivision (if performed appropriately). Thus the decision about its position in the generalization workflow is less critical. Most line simplification approaches for planar subdivisions keep vertices of degree higher than two. In other words, vertices shared by more than two regions remain in the generalized map. As area aggregation greatly reduces the number of such vertices, the simplification of lines should be the last operator in the generalization workflow. This implies that we need to ensure size constraints for the areas during line simplification. In Sect. 6.2 the method of de Berg *et al.* (1998) is extended to meet this requirement.

We illustrate the functionality of our generalization workflow, before presenting its modules and results in detail: Figure 5.7 shows a small example and gives references to detailed explanations of the processing steps.

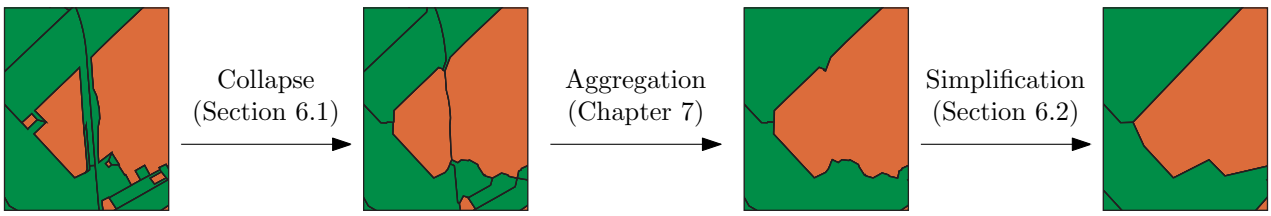


Fig. 5.7: The proposed generalization workflow applied on a sample of the ATKIS DLM 50 (left). The result (right) satisfies specifications for the DLM 250.

Chapter 6

Algorithms for Geometric Generalization

This chapter presents two new generalization methods: A method for collapsing areas and area parts to lines (Sect. 6.1) and a line simplification method that ensures the size constraints that are defined for areas (Sect. 6.2). Both methods can be classified as geometric generalization operators, that is, they mainly influence the shapes of objects.

6.1 Area collapse

As discussed in the previous section, the collapse of narrow polygons or polygon parts is needed to remove disturbing details before applying an aggregation procedure. In some cases the collapse of areas is necessary to comply with database specifications. Recall the definition of a river in Table 2.1: depending on its width the river either needs to be represented by an area (width $\geq$ 12 m) or by a line (width $<$ 12 m). Usually a greater width threshold is defined for the smaller target scale, thus map generalization requires some areas to be collapsed to lines. The collapse procedure that is presented in this section has been presented in detail, focusing on the derivation of road centerlines from a cadastral dataset of polygons (Haunert & Sester, 2004, 2008a; Haunert, 2005a). Collapse is not explicitly mentioned in the terminology of generalization operators by Hake *et al.* (1994) (recall Fig. 2.1). However, it can be seen as a certain form of symbolization, since a physical entity of a two-dimensional extent becomes represented by a line symbol.

Section 6.1.1 lists requirements for the collapse task. Different skeleton structures that can be applied are shown in Fig. 6.1 and explained in Sections 6.1.2 to 6.1.4. Raster-based methods are explicitly excluded from the discussion, since the existing input as well as the demanded output for this work is vector-based. Nevertheless, raster-based skeleton operators are often used for cartographic applications (Su *et al.*, 1998). A basic method based on straight skeletons is shown in Sect. 6.1.5, which handles single areas. Section 6.1.6 presents a global approach for areas in a planar subdivision.

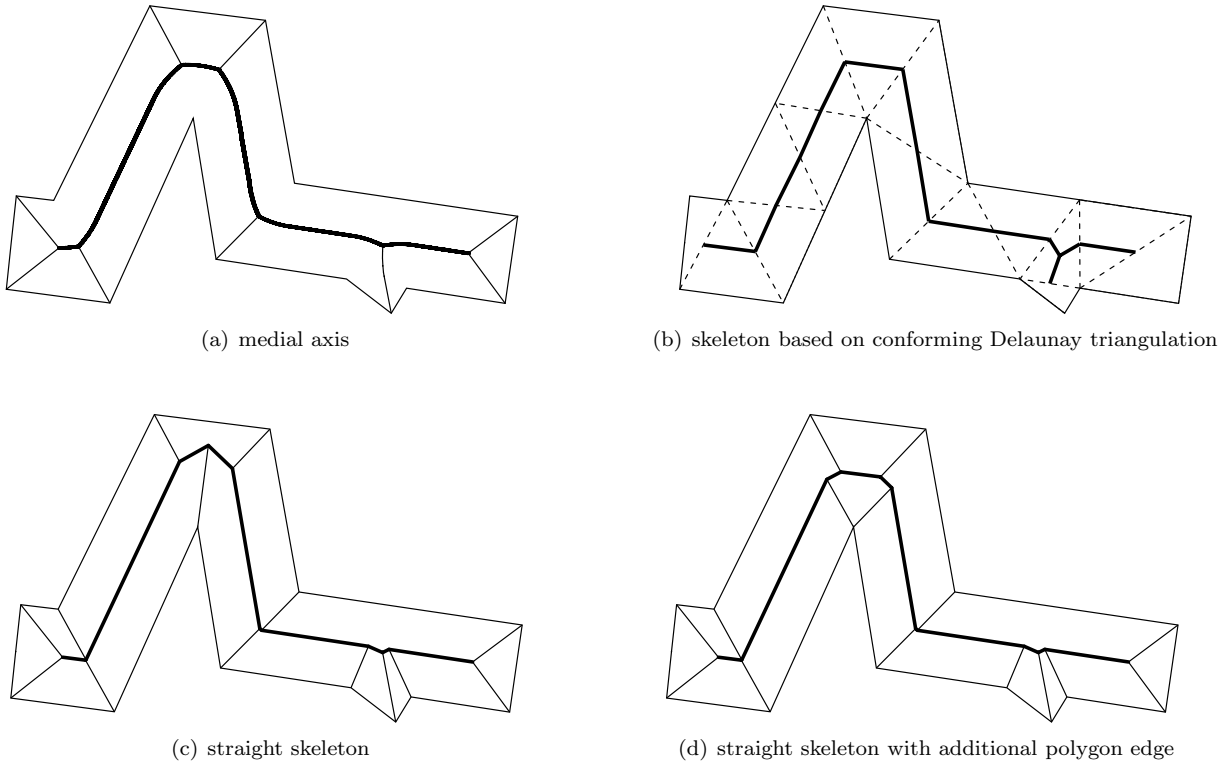
6.1.1 Requirements

We define the following requirements for the collapse of areas:

1. A collapsed area needs to be distributed to its neighbors, such that the subdivision remains simple.
2. The shapes of neighbors should be modified as little as possible.
3. Semantic distances should be considered, such that relatively large parts of the collapsed area change to similar classes.
4. Boundaries of high priority need to be preserved.
5. A partial collapse must be possible, such that a feature keeps its two dimensional representation in parts that exceed a certain width.

In the case that a line representation is to be created, the following requirement needs to be added:

6. The derived centerline must reflect the major elongations of the polygon and ignore small anomalies.



(a) medial axis

(b) skeleton based on conforming Delaunay triangulation

(c) straight skeleton

(d) straight skeleton with additional polygon edge

Fig. 6.1: Comparison of skeleton operators (centerlines bold).

The following additional requirement was not considered in our previous publication (Haunert & Sester, 2008a):

7. When processing multiple areas the result should not depend on the definition of an order.

For specific problems we need to make these requirements concrete and add further requirements. For example, in the case that a line representation needs to be derived for rivers, other geometric characteristics may be favorable than for road centerlines. In a previous work we presented an algorithm that eliminates skeleton edges corresponding to polygon parts that do not conform to a defined object model (Haunert & Sester, 2008a). We applied a model for roads to derive road centerlines and we defined rules to reshape the centerlines at road junctions. In the following sections, however, we focus on possibilities to satisfy the general requirements stated above.

6.1.2 Medial axis

Figure 6.1(a) shows the *medial axis*, which is defined as the locus of points that have more than one closest neighbor on the polygon boundary. The bold centerline in the figure is defined by those points of the medial axis whose closest neighbors do not belong to adjacent edges of the polygon. This skeleton is widely used in geographic information sciences for the analysis of shapes. It consists of straight lines and second order lines, which result from reflex angles of the polygon. An algorithm of linear time was found by Chin *et al.* (1995). For practical reasons often approximations of medial axes are used, which only consist of straight segments. Since the medial axis is equal to the line Voronoi diagram of the polygon edges, it can be approximated by the Voronoi diagram of a discrete set of boundary points. To generate a good approximation the vertices of the polygon need to be densified first. This technique is used by Roberts *et al.* (2005) to derive street centerlines. Since the Voronoi diagram for a set of points is the dual graph of the Delaunay triangulation, the approximated medial axis can be derived by connecting the circumcenters of adjacent triangles of the triangulation. Here a constrained Delaunay triangulation can be used to ensure that the triangulation covers the same area as the polygon. The medial axis has a very smooth appearance.

6.1.3 Triangulation-based skeleton

Another skeleton, which is also based on a constrained Delaunay triangulation of the polygon, was presented by Chithambaram *et al.* (1991). Alternatively a conforming Delaunay triangulation can be applied (Bader & Weibel, 1997). The basic procedure for the skeleton construction from the triangulation does not differ for both cases and is defined as follows: For each triangle that shares one edge with the polygon a skeleton edge is constructed by connecting the centers of both other triangle edges. Note that connecting the centers of triangle edges is not the same as connecting the circumcenters of triangles, which is the approach used to approximate the medial axis.

Triangles that do not share any edge with the polygon (0-triangles) need to be handled specially. A common approach is to introduce three skeleton edges that connect the triangle's centroid with each center of its edges. The result of this procedure is shown in Fig. 6.1(b) using a conforming Delaunay triangulation. Penninga *et al.* (2005) discuss its advantages compared to the constrained Delaunay triangulation. Since 0-triangles with very obtuse angles can be avoided, unwanted shifts of the skeleton can be reduced. Another problem that is presented by Penninga *et al.* is the possible generation of unwanted spikes in 0-triangles. Figure 6.1(b) shows that the small disturbance in the polygon boundary has a significant effect on the major axis of the skeleton; a spike is produced. For a better appearance of the skeleton, additional rules for the handling of 0-triangles can be defined. However, very difficult cases can appear like multiple adjacent 0-triangles. The occurrence of these spikes is the biggest disadvantage of the triangulation-based skeleton. The conforming Delaunay triangulation is not unique and so the triangulation-based skeleton is not well-defined. This is a disadvantage if results need to be reproduced.

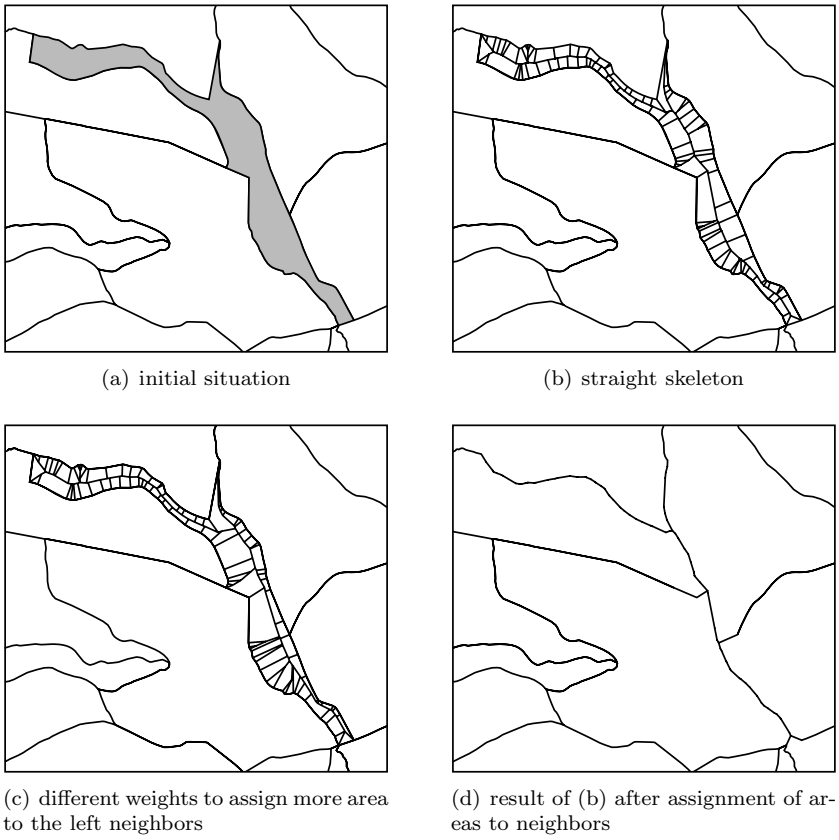
6.1.4 Straight skeleton

Figure 6.1(c) shows the *straight skeleton*, which was presented by Aichholzer *et al.* (1995). It can be imagined as a roof that covers the polygon ground plan and consists of planes that arise with constant slope from each polygon edge. Each skeleton edge is a bisector of two polygon edges. The construction of the straight skeleton is based on a stepwise shrinking process of the polygon, that can be performed by simultaneous parallel offsets of the polygon edges. During this offset two different types of events need to be handled: *edge events*, which mean that a polygon edge is omitted, and *split events*, which mean that a polygon edge is split in two.

An important geometric property is that the skeleton mostly consists of long straight lines that reflect the major axes of the polygon. The geometrical anomaly on the lower right side still has an effect on the shape of the skeleton in Fig. 6.1(c). Yet the generated spike is much less dominant than in the triangulation-based skeleton and is comparable to the medial axis. We can define the centerline of the skeleton as the union of all skeleton edges that are not incident to a vertex of the polygon. This centerline is depicted with a bold line.

We can alter the straight skeleton by assigning slopes to certain roof planes that differ from the default value. With this variation we can shift the centerline in certain directions. This has been applied in a different domain, namely for the three-dimensional reconstruction of buildings from ground plans and laser scanning data (Haala & Brenner, 1997). In Sect. 6.1.5 we use this technique for the collapse of areas in a topographic database. Eppstein & Erickson (1998) describe a sub-quadratic algorithm for the straight skeleton. They also consider the case of different roof slopes. Felkel & Obdržálek (1998) present an implementation of the straight skeleton and discuss special cases.

A disadvantage of the straight skeleton is its sensitivity to reflex angles close to 360°. In these cases the skeleton will not be situated close to the center of the polygon. The uppermost bend of the polygon in Fig. 6.1(c) is a typical example of this case. We can solve this problem by introducing additional polygon edges of zero length in reflex vertices. Consequently, an additional roof plane arises from each of these vertices. We define the orientation of such a plane as follows: the projection of the plane's normal vector to the horizontal plane must be collinear with the bisector of the original edges incident to the reflex vertex. Its slope is defined to be the same as the slope of the other roof planes. To select the vertices that are treated like this, we define a threshold for the maximum allowed angle between two polygon edges. Tănase & Veltkamp (2004) use this technique to approximate the medial axis with the straight skeleton. Figure 6.1(d) shows the straight skeleton after the insertion of an additional edge. This skeleton has the same advantages as the standard straight skeleton. In addition the effect of centerline shifts at reflex vertices is reduced.



(a) initial situation

(b) straight skeleton

(c) different weights to assign more area to the left neighbors

(d) result of (b) after assignment of areas to neighbors

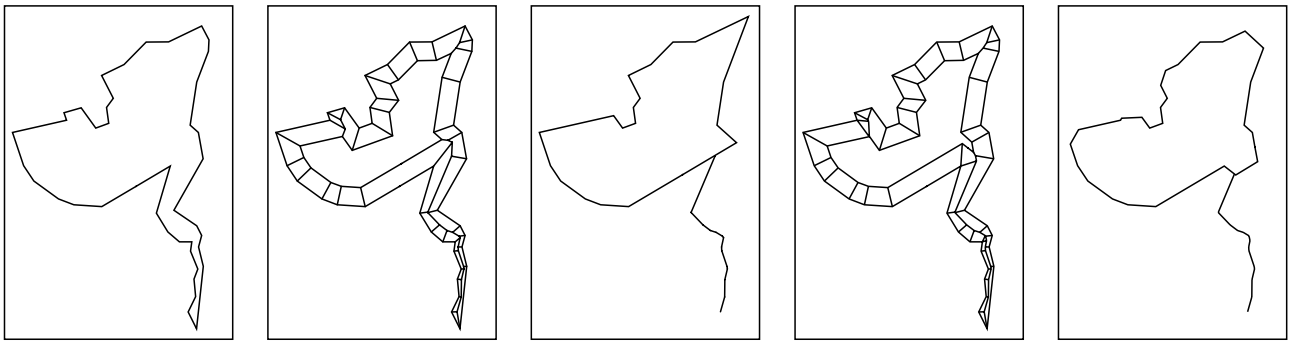
Fig. 6.2: Collapse operator based on straight skeleton.

6.1.5 Treatment of single areas in a topographic database

Ai & van Oosterom (2002) approach the collapse task with the triangulation-based skeleton. It allows different weights to be applied to different polygon edges, and so the centerline can be shifted such that a greater portion of the area is assigned to semantically similar neighbors. However, the risk of obtaining unwanted spikes or zigzagging boundaries is a disadvantage. The medial axis is less sensitive to these disturbances, but the exact algorithms are hard to adapt to special problems like the partial collapse or the collapse with different weights.

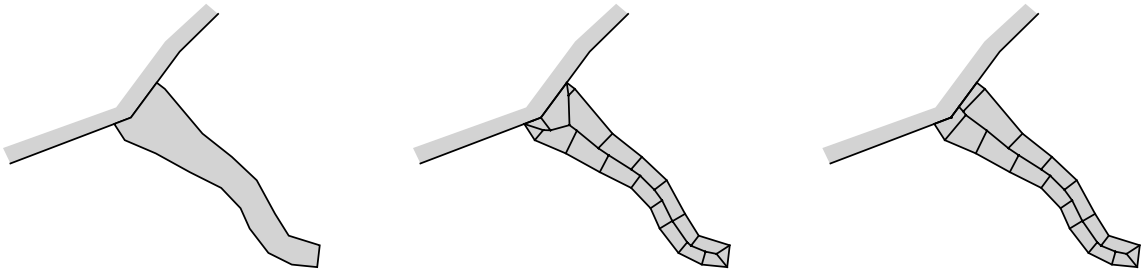
In contrast, we can apply the straight skeleton to all stated problems with some simple modifications to the original algorithm. With the straight skeleton extension presented by Tănase & Veltkamp (2004), that is, with multiple additional edges in the reflex vertices, no significant geometric difference exists in comparison with other approximations of the medial axis. However, the smooth skeleton that is produced by the medial axis sometimes does not correspond to the level of granularity of the original object, especially when straight man-made objects such as roads or buildings are involved. Here we need a less smooth skeleton that can be achieved adding maximally one edge to a reflex vertex as was shown in Fig. 6.1(d). Figure 6.2(a) shows a planar subdivision with a gray polygon that is to be collapsed. The edges of the straight skeleton define a decomposition of the original polygon (Fig. 6.2(b)). Each fragment of this decomposition is incident to only one polygon edge. Therefore it can be unambiguously assigned to one of the neighbors. Fragments that result from artificially added planes in reflex vertices can simply be separated by the original bisector into two parts, in order to assign the area to the original polygon edges. Figure 6.2(d) shows the resulting map after the application of the collapse operator based on the straight skeleton with equal inclinations of planes. The area was assigned to its neighbors leading to only small changes of their shapes. In Fig. 6.2(c) we assigned a larger weight and accordingly a larger part of the eliminated area to the neighbors on the left side.

Figure 6.3(a) shows a lake and an incoming river as one area. The partial collapse task requires that this area is split into lines and areas according to a given width threshold. A solution with the straight skeleton can be found that takes advantage of the shrinking process on which the skeleton construction is based. The proposed method is very similar to the morphologic opening operator, which is often applied to raster data (Gonzalez



(a) (b) (c) (d) (e)

Fig. 6.3: Partial collapse of a river and a lake. (a) Initial polygon. (b) Uncompleted straight skeleton and offset polygon at 20 m. (c) After application of the inverse parallel offset to the remaining polygon in (b). (d) As (b) but with additional edges in reflex vertices. (e) After application of the inverse parallel offset with additional edges in reflex vertices to the remaining polygon in (d).



(a) river polygon touching boundary of lake

(b) collapse without consideration of the boundary of high priority

(c) collapse with consideration of the boundary of high priority

Fig. 6.4: Straight skeleton that preserves a boundary of high priority.

& Woods, 2002). More precisely, the events described in Sect. 6.1.4 will be performed only if they happen within half of the width threshold. Events which happen later are not to be processed. Figure 6.3(b) shows the skeleton after its termination at an early stage. We applied a width threshold of 40 m. In some parts the original polygon has shrunk to lines. Here we need to derive the centerline of the constructed skeleton, which we can do in the same manner as in the case of a complete collapse. In other parts a shrunken polygon is left. Here we need to recover the extent of the original polygon boundary by offsetting the shrunken polygon as shown in Fig. 6.3(c). After constructing the offset, we need to clip edges of the derived centerline that intersect the resulting polygon. In order to avoid the unwanted effects of sharp reflex vertices, we can introduce additional edges as shown in Fig. 6.1(d). This approach should be used for both the offset of the polygon to its interior (Fig. 6.3(d)) and the inverse offset in the second step of the procedure (Fig. 6.3(e)). With this method, the spike in the upper right corner of Fig. 6.3(c) is avoided.

Often we need to perform an area collapse, but certain topological relationships must not be lost and boundaries of high priority must not be changed. An example is a river that flows into a lake (Fig. 6.4(a)). The derived linear representation for the river must still be connected to the lake, but a modification of the lake boundary is not allowed. Applying a naive collapse operator does not assure this. Figure 6.4(b) shows the straight skeleton of the river polygon using default settings. The topological relationship is lost. We can solve the task by modifying the inclinations of the planes that define the skeleton. Increasing the inclinations of planes in polygon edges that are shared by the river and the lake causes the skeleton to move in the direction of the lake. Defining vertical planes would mean that the skeleton touches these edges. In order to get along with the existing algorithms and implementations that rely on offsets in two dimensions, we can define a plane with a very high inclination, for example 1000%, instead of a vertical plane. In practice both definitions do not result in visual differences; small topological errors can easily be cleaned with a snapping routine. Figure 6.4(c) shows the result of this collapse operator.

6.1.6 A global approach for the collapse of areas in a planar subdivision

We can apply the proposed operator from Sect. 6.1.5 for the collapse or partial collapse of single areas. If we consider a planar subdivision that consists of multiple areas, we would need to define which areas are to be collapsed and in which order this is done. Generally, we will not obtain the same result, if we apply the collapse procedure on the same areas in a different order. To avoid a large set of required parameters, we can use the following procedure that handles all areas of a planar partition at the same time. Similarly to the proposed partial collapse method for a single area, the procedure only requires a single offset parameter. Figure 6.5 illustrates the approach.

(a) original dataset ATKIS DLM 50 including narrow polygons, for example the river in the upper right corner

(b) polygons after offset to their interior by 25 m and straight skeleton of free space (white polygon)

(c) resulting map after merging fragments of the white polygon in (b) to their neighbors

Fig. 6.5: Collapse of areas in a planar partition by offset and skeleton construction.

In a first step, we apply the offset of each polygon to its interior. A polygon or a polygon part whose width falls below the offset does not survive the procedure. In a second step, we use the straight skeleton to collapse the resulting free space. Again, to avoid that sharp reflex vertices have high influences on the result, we can use the straight skeleton with additional edges of zero length, which was shown Fig. 6.1(d).

The original map in Fig. 6.5(a) contains multiple small or narrow polygons, for example, the river in the upper right corner or the narrow connection between the large settlement and the small annex on its left side. In Fig. 6.5(b) we applied an offset of 25 m and we constructed the skeleton for the resulting free space (white). The straight skeleton partitions the free space into small fragments. For each of these fragments, there is only one adjacent offset polygon (classified areas). Merging each fragment into its classified neighbor, we obtain the map in Fig. 6.5(c). Small polygons and polygon parts of width less than 50 m are eliminated and the subdivision remains simple. Polygon parts whose widths exceed 50 m survive the procedure. This corresponds to 0.2 mm in scale 1:250 000. According to Hake *et al.* (1994, page 110), two thin lines should not be closer to each other in a colored map. We term this minimum distance of 0.2 mm as the *map resolution*.

In some cases, polygons are fragmented into several parts. For example, the big settlement is fragmented into five parts, four of them forming a contiguous region. The straight skeleton has been used before for the decomposition of shapes into natural parts (Tănase & Veltkamp, 2003; Tănase, 2005). However, the new approach allows the collapse, partial collapse, and decomposition of multiple areas to be performed comprehensively.

We will use the result after this global procedure as input for the aggregation method presented in Chapter 7. We therefore should be concerned about a possible increase of the number of polygons. The collapse of small areas results in a decrease of the number of polygons (some river polygons have vanished). The decomposition of polygons results in an increase of the number of polygons (the big settlement area has been split into five parts). With both effects, however, we only obtain a marginal increase of 1% for our dataset.

6.2 Line simplification

This section presents a line simplification method that can be used in a workflow for the generalization of a planar subdivision, together with the developed methods for area collapse and aggregation. The simplification of lines is a well-known problem that has found much attention before. For the simplification of lines that define a planar subdivision, we can apply an existing method after de Berg *et al.* (1998), which is reviewed in Sect. 6.2.1. However, we need to modify this method, in order to ensure size constraints for areas. That is, if we simplify the boundary of a region, its area must not fall below the required size threshold. Section 6.2.2 deals with this problem. Section 6.2.3 summarizes the method and presents results of a test.

6.2.1 An existing optimization approach to line simplification

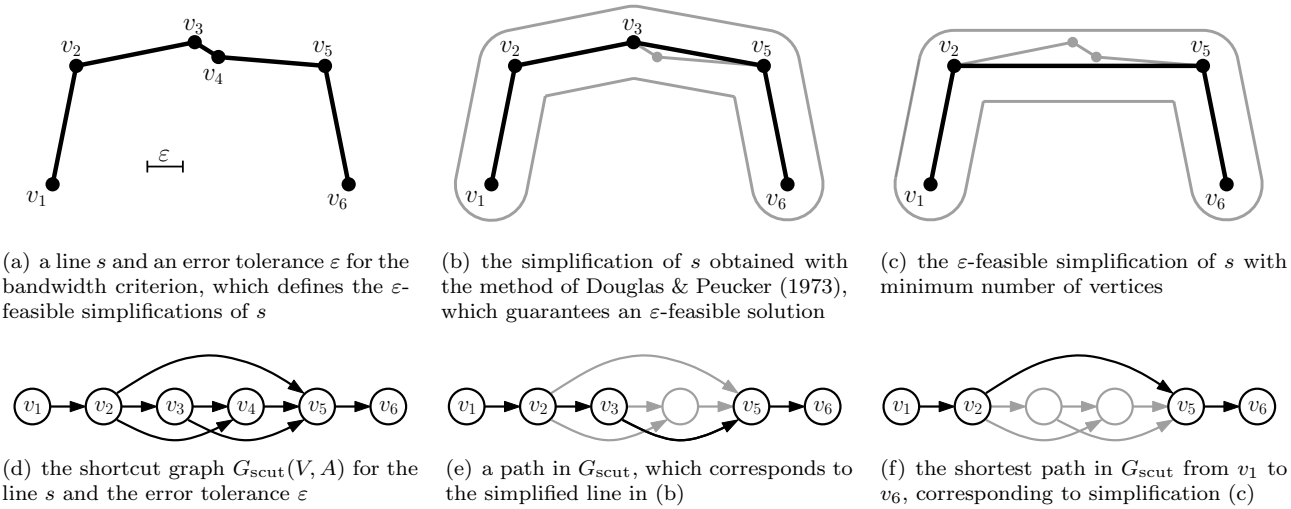
Usually, a cartographic line is given with a sequence of line vertices $s = (v_1, v_2, \ldots, v_n)$. For example, the line in Fig. 6.6(a) is defined as $s = (v_1, v_2, v_3, v_4, v_5, v_6)$; each pair of consecutive vertices is connected by a straight line. Most line simplification methods yield a simplified line s' that is a subsequence of s, that is, $s' = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ with $i_1 = 1$, $i_k = n$, $1 < k \leq n$, and $i_{j+1} > i_j$ for $j = 1, 2, \ldots, k-1$. An example of such a method is the algorithm of Douglas & Peucker (1973). Though it is not based on global optimization, we briefly review the method, as it is very often applied in practice.

In addition to the line s, the method of Douglas & Peucker (1973) requires a single parameter $\varepsilon \in \mathbb{R}^+$ as input, which defines the degree of simplification. More precisely, it defines an error tolerance. The requirement not to exceed this tolerance is often referred to as the *bandwidth criterion*:

- For each line segment $(v_{i_j}, v_{i_{j+1}})$ of the simplified line s', the vertices $v_{i_j}, v_{i_j+1}, \ldots, v_{i_{j+1}}$ of s must be within ε-distance.

We call any simplification that satisfies this criterion *ε-feasible*. In terms of quality, the bandwidth criterion ensures positional accuracy of the simplified line.

The algorithm of Douglas & Peucker (1973) starts with the simplified line $s'_0 = (v_1, v_n)$ and iteratively adds vertices from s until the bandwidth criterion is satisfied. Generally, for a line segment (v_a, v_c) that does not satisfy the bandwidth criterion, the vertex v_b with $a < b < c$ that has maximal distance from the line (v_a, v_c) is selected. The segment (v_a, v_c) is replaced by the two segments (v_a, v_b) and (v_b, v_c). For our example, the result of this procedure is shown in Fig. 6.6(b). Vertex v_3 was added first, then also v_2 and v_5 needed to be added.



(a) a line s and an error tolerance ε for the bandwidth criterion, which defines the ε-feasible simplifications of s

(b) the simplification of s obtained with the method of Douglas & Peucker (1973), which guarantees an ε-feasible solution

(c) the ε-feasible simplification of s with minimum number of vertices

(d) the shortcut graph $G_{\text{scut}}(V, A)$ for the line s and the error tolerance ε

(e) a path in G_{scut}, which corresponds to the simplified line in (b)

(f) the shortest path in G_{scut} from v_1 to v_6, corresponding to simplification (c)

Fig. 6.6: Two simplifications of a cartographic line and their corresponding paths in the shortcut graph G_{scut}. The simplification in (c) is optimal according to the objective for a minimum number of vertices, which was proposed by Deveau (1985). This solution is given with the shortest path in G_{scut} from v_1 to v_6, see (f).

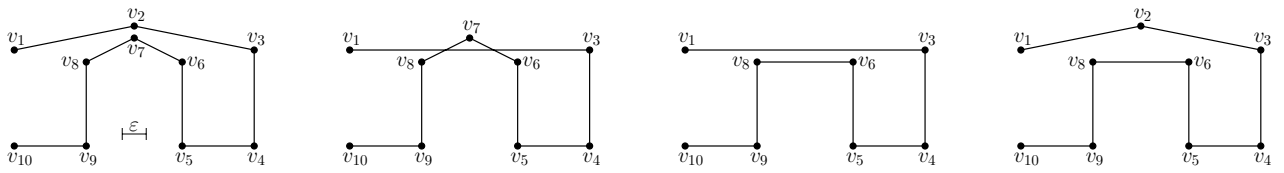
Deveau (1985) proposed another approach that also ensures the bandwidth criterion. In contrast to the algorithm of Douglas & Peucker (1973) it is based on global optimization: Among the ε-feasible simplifications of s, a search is performed for the solution with the minimum number of vertices. Figure 6.6(c) shows the solution that is optimal according to this criterion. It only contains four vertices instead of five vertices, which is the result of the algorithm of Douglas & Peucker (1973).

To solve this optimization problem we can define the *shortcut (di)graph* $G_{\text{scut}}(V, A)$ with V being the set of vertices of s and A including an arc for a pair of vertices of V if the connecting line segment satisfies the bandwidth criterion. Figure 6.6(d) shows the graph G_{scut} for our example. Each ε-feasible simplification of s corresponds to a path in G_{scut}; two paths are shown in Figures 6.6(e) and 6.6(f). Selecting an arc (v_i, v_j) implies that vertices $v_{i+1}, v_{i+2}, \ldots, v_{j-1}$ are ommited. Therefore, we call an arc in A a *shortcut*. Defining unit lengths for arcs, that is, unit costs for shortcuts, the ε-feasible simplification of s with the minimum number of vertices corresponds to the shortest path from v_1 to v_n in G_{scut}.

We can solve the line simplification problem in two steps: first we construct the shortcut graph G_{scut} and secondly we search for the shortest path from v_1 to v_n in G_{scut}. In the first step all feasible shortcuts need to be found, which requires $\mathcal{O}(|V|^2)$ time (Chan & Chin, 1996). The second problem can be solved with any algorithm for the general shortest path problem, for example, the method of Dijkstra (1959). However, the graph G_{scut} has a special property that we can exploit to define a more efficient solution: it does not contain any cycle. In this case, we can solve the problem with a simple dynamic programming method (Papadimitriou & Steiglitz, 1998). Generally, this approach allows the shortest path in an acyclic digraph to be found in $\mathcal{O}(|V| + |A|)$ time. In conclusion, constructing the shortcut graph G_{scut} is more complex than finding the shortest path in G_{scut}. Solving the whole simplification problem requires $\mathcal{O}(|V|^2)$ time.

Certainly, the aim for a minimum number of vertices is reasonable if the general objective is data compression. Furthermore, minimizing the number of vertices will reduce the clutter on the map and improve its legibility. However, in the context of spatial data quality, the approach of Deveau (1985) is of particular interest as it can be extended to consider other performance measures and additional constraints. Campbell & Cromley (1991) define a general model that allows different optimization criteria to be incorporated. With some measures and constraints the problem becomes more involved. However, solutions have been found for the preservation of angles (Chen *et al.*, 2005) and distances (Gudmundsson *et al.*, 2007), which both can be seen as criteria of geometrical accuracy.

The problem of how to avoid self-intersecting lines in the generalized map has been investigated in detail. Estkowski & Mitchell (2001) proved that it is NP-hard to reduce a planar subdivision to a minimum number of vertices while ensuring the bandwidth criterion and keeping the subdivision simple. de Berg *et al.* (1998) developed an efficient heuristic that guarantees results without self-intersections. We do not review the proof of Estkowski & Mitchell (2001) and the method of de Berg *et al.* (1998) in detail, but discuss a simple example to illustrate why the simplicity requirement renders the problem combinatorially hard and how we can tackle the problem with a simple heuristic.



(a) a line s with tolerance ε for the bandwidth criterion

(b) a simplification of s with self-intersections

(c) the optimal feasible simplification

(d) a feasible simplification that is not optimal

Fig. 6.7: A line with three simplifications.

Figure 6.7(a) defines an instance of the simplification problem: The aim is to find the ε-feasible, not self-intersecting simplification s' of the polygonal line s that has a minimum number of vertices. In order to apply our approach based on a shortest path formulation, we would like to decide independently for each pair of vertices whether it defines a feasible simplification. Applying shortcut (v_1, v_3) results in a self-intersecting polygon (see Fig. 6.7(b)). Hence, we might conclude that the shortcut is not allowed. This conclusion, however, is wrong:

Applying $(v_1\,, v_3)$ together with the shortcut $(v_6\,, v_8)$ results in a feasible simplification, which actually is the optimal solution (see Fig. 6.7(c)). Such dependencies between shortcuts render the problem NP-hard (Estkowski & Mitchell, 2001). If we forbid the shortcut $(v_1\,, v_3)$ and apply the method based on shortest path formulations, we will find the solution in Fig. 6.7(d). Though this is not optimal, it is at least a feasible solution. In fact, de Berg *et al.* (1998) propose a heuristic method for the simplification of a planar subdivision that is based on this approach: shortcuts that potentially violate the simplicity constraint are excluded in advance from the arc set A. Then, the shortest path in G_{scut} offers a feasible, but not the optimal simplification of a line. To simplify all lines of the planar subdivision, the boundaries between each two adjacent areas are processed in succession.

6.2.2 Line simplification with size constraints

Let's assume that we wish to simplify the boundary s between areas ℓ (the area to the left of s) and r (the area to the right of s). We can do this by searching for the shortest path in a shortcut graph. However, with the basic approach, we run the risk that one of the areas ℓ or r becomes too small, that is, the area ℓ falls below a required size threshold $\theta_\ell \in \mathbb{R}^+$ or the area r falls below its threshold $\theta_r \in \mathbb{R}^+$. To avoid these cases, we can introduce hard constraints into the shortest path formulation.

For this, we define the function $\Delta w : A \to \mathbb{R}$ such that applying shortcut $(v_i, v_j) \in A$ implies that the size of ℓ increases by $\Delta w(v_i, v_j)$ as it is shown in Fig. 6.8. If applying shortcut (v_i, v_j) implies that the size of ℓ decreases, then $\Delta w(v_i, v_j)$ is negative. We define our problem as follows:

Problem (Line simplification with size constraints)**.** Given a line $s = (v_1\,, v_2\,, \ldots\,, v_n)$, the original area sizes $w_\ell\,, w_r \in \mathbb{R}^+$, the required sizes $\theta_\ell\,, \theta_r \in \mathbb{R}^+$, and the tolerance $\varepsilon \in \mathbb{R}^+$, find an ε-feasible simplification $s' = (v_{i_1}\,, v_{i_2}\,, \ldots\,, v_{i_k})$ with $i_1 = 1$, $i_k = n$, $1 < k \leq n$, and $i_{j+1} > i_j$ for $j = 1\,, 2\,, \ldots\,, k-1$, such that the line has a minimum number of vertices and the following constraints are satisfied:

- Area ℓ must not become too small, i.e.,

$$w_\ell + \sum_{j=1}^{k-1} \Delta w(v_{i_j}\,, v_{i_{j+1}}) \geq \theta_\ell\,.$$

- Area r must not become too small, i.e.,

$$w_r - \sum_{j=1}^{k-1} \Delta w(v_{i_j}\,, v_{i_{j+1}}) \geq \theta_r\,.$$

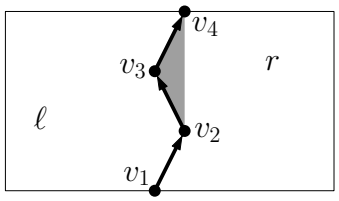


Fig. 6.8: Applying shortcut $(v_2\,, v_4)$, the shaded region is added to ℓ. The size of ℓ increases by $\Delta w(v_2, v_4)$.

Bose *et al.* (2006) investigated a similar problem, which is to answer the following question:

> Given a line $s = (v_1\,, v_2\,, \ldots\,, v_n)$ does there exist a simplification $s' = (v_{i_1}\,, v_{i_2}\,, \ldots\,, v_{i_k})$ with $k < n$ such that $\sum_{j=1}^{k-1} \Delta w(v_{i_j}\,, v_{i_{j+1}}) = 0$?

Obviously, this is a special decision version of line simplification with size constraints: It is the special version where $w_\ell = \theta_\ell$, $w_r = \theta_r$, and ε is set so high that any simplification is ε-feasible. Bose *et al.* (2006, Theorem 7) proved that their problem is NP-hard; therefore, we can conclude that the more general problem line simplification with size constraints is NP-hard, too. We cannot hope for an efficient and exact solution to the problem.

Bose *et al.* (2006) also discussed an optimization version of their problem where the aim is to minimize $\left|\sum_{j=1}^{k-1} \Delta w(v_{i_j}\,, v_{i_{j+1}})\right|$ and the number of edges in the simplified line must not exceed a given value. The authors presented an efficient algorithm for this problem that guarantees a solution with a bounded additive error, that is, the cost of the obtained solution never exceeds the cost of the optimal solution by more than a

user-defined value $\delta H > 0$. However, this does not allow our general problem to be solved, that is, to cope with hard size constraints.

To solve line simplification with size constraints, we consider the problem in the general context of weight-constrained shortest path problems. In addition to edge lengths l these require an additional weight for each edge. The problem is to find the shortest path whose weight is within given bounds. Many non-efficient exact algorithms and heuristics exist to solve this problem, which we can adopt. A review of methods is given by Dumitrescu & Boland (2001), which includes an approach by integer programming. Dumitrescu & Boland (2001) also discuss a simple approach that uses algorithms for the K shortest path problem. This approach is interesting, as it allows the problem to be solved mainly by reusing code for the unconstrained shortest path problem. We now discuss the basic idea of this approach.

If we remove the size constraints, we can solve our line simplification problem with an efficient algorithm for the shortest path problem. We can do this in a first step. If we obtain a solution that satisfies the size constraints, we have the optimal solution to the constrained problem. If we obtain a solution that does not satisfy the size constraints, we can search for the second shortest path in the graph and again check whether the size constraints are satisfied. We can simply continue with this routine, until we find the first feasible path, that is, we search the third, fourth, and fifth shortest path in the graph and so on. After a finite number of iterations we gain the optimal solution of the line simplification problem with size constraints. Hopefully, there is a feasible solution that requires the addition of only a few additional edges, compared with the optimal solution of the relaxed problem. In this case, the method will find the optimal solution in relatively short time.

Normally, the K shortest path problem requires an instance of the shortest path problem $(G\,, l\,, u\,, v)$ and the parameter K as input. The problem is to find a set of K paths, each being at least as short as any other path that is not included in the set. However, we do not know how many paths we need to investigate, in order to find a feasible one. This means, we cannot specify the parameter K in advance. Nevertheless, we can apply the existing algorithm of Hoffman & Pavley (1959) for our problem. It defines a simple procedure that allows a given path to be deviated, such that each Nth shortest path p_N with $N > 1$ can be obtained by deviating one of the paths $p_1\,, p_2\,, \ldots\,, p_{N-1}$. The algorithm starts with the shortest path p_1 and adds all deviations of p_1 to a list, ordered by their lengths. Consequently, the first path in this list is the second shortest path p_2. Its deviations are added to the list in the correct order, which implies that the next path in the list is p_3. This procedure can be continued without specifying the number of paths in advance. The list will simply grow until we find a feasible path.

If we are unlucky, we need to process many iterations until we find a feasible path. As the list can grow extremely fast, we might wish to abort the procedure after a certain number of iterations. In this case, we could still apply the approach by integer programming, using the formulation of Dumitrescu & Boland (2001). However, this approach was not tested. The next section presents the results that were obtained using the algorithm of Hoffman & Pavley (1959).

6.2.3 Summary of the developed line simplification method

The line simplification method based on the shortcut graph $G_{\text{scut}}(V, A)$ was implemented in a C++ program. The developed method was implemented to simplify a boundary that separates two areas. To simplify the whole planar subdivision, such boundaries are simplified in succession. For this purpose it is practical to maintain the polygons in a topological data structure that explicitly represents incidence relationships between nodes and arcs as well as arcs and faces. The doubly-connected edge list, which is explained by de Berg *et al.* (2008), was applied for this task.

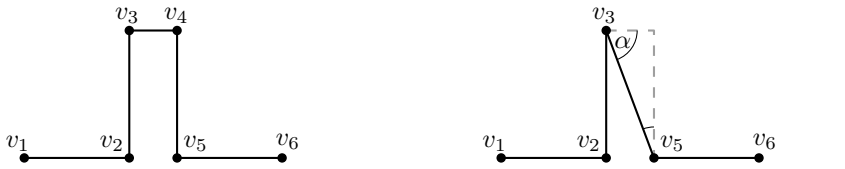
The basic optimization criterion was applied, that is, the length of each arc was defined by 1. However, several hard constraints were considered to ensure quality. These are summarized by the following list:

- The simplified lines must satisfy the bandwidth criterion. The parameter ε is defined according to the typical resolution of a paper map, which is 0.2 mm. Hence, for the target scale 1:250 000, we need to define $\varepsilon = 50$ m.
- The simplified lines must not intersect. In order to ensure this requirement, the heuristic approach of de Berg *et al.* (1998) is applied, i.e., shortcuts that potentially violate the simplicity requirement are excluded in advance from the arc set A.

- Size thresholds for areas are ensured applying the newly developed method.

Initial tests with these requirements revealed that the method often produces unwanted sharp corners. Therefore, we add the following requirement for shortcuts, which is illustrated in Fig. 6.9:

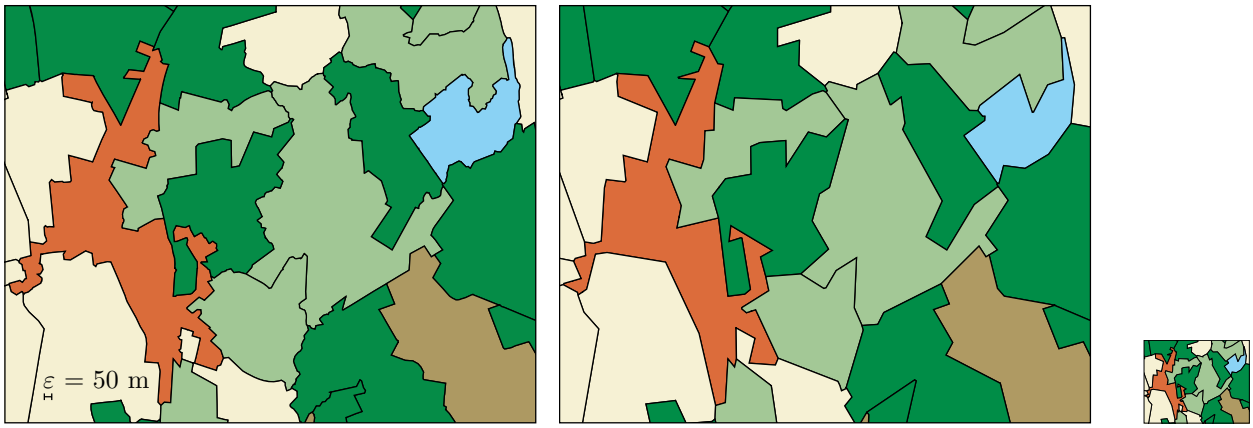
- A shortcut (v_i, v_j) is only allowed if the angle α between the corresponding line and the line segment (v_i, v_{i+1}) does not exceed a user-defined threshold $\alpha_{\max}$. The same requirement must hold for the angle between (v_i, v_j) and (v_{j-1}, v_j). By experiments, a setting of $\alpha_{\max} = 30°$ was found to give satisfactory results.



(a) an input line (b) the result of applying shortcut $(v_3\, v_5)$

Fig. 6.9: A shortcut that does not satisfy the angle constraint with $\alpha_{\max} = 30°$: The angle α between lines $(v_3\, v_4)$ and $(v_3\, v_5)$ is too wide.

Figure 6.10 shows a result that was obtained with this setting. The sample is a small part of a planar subdivision that contains 733 areas and 52832 vertices. The accuracy of the lines corresponds to scale 1:50 000. The simplification of the whole subdivision took 6 minutes on a computer having 1 GB RAM and a 2.8 GHz Intel Pentium 4-CPU. This also includes the time that was needed to read the set of input polygons, to build the topological data structure from the geometries, and to write the polygons to a file. The simplified subdivision has 10413 vertices, which means that the number of vertices was reduced by 80%.



(a) input dataset with the error tolerance ε (b) the dataset after line simplification (c) result in (b) at scale 1:250 000

Fig. 6.10: A result of the developed line simplification method. The sample in (a) was obtained by aggregating areas of the ATKIS DLM 50 to satisfy size thresholds for the ATKIS DLM 250, using the method that is presented in the next chapter. It is part of the sample in Fig. 8.8.

Chapter 7

A new Approach to Area Aggregation by Optimization

We now concentrate on the aggregation of areas in a planar subdivision. Section 7.1 gives a general, graph-theoretical problem statement that builds on our definitions of logical consistency and semantic accuracy. Section 7.2 specifies the problem with respect to different compactness measures. Section 7.3 proves that the area aggregation problem is NP-hard. As we cannot hope for an efficient and exact algorithm, we turn to mixed-integer programming and heuristics.

We introduce different MIPs for the aggregation problem, which were inspired by formulations of Zoltners & Sinha (1983), Williams (2002), and Shirabe (2005b) for spatial allocation problems. We have presented this new approach to generalization by mixed-integer programming in two publications (Haunert & Wolff, 2006; Haunert, 2007b). Similarities and differences of the developed MIPs and the existing MIPs for spatial allocation will be explained. We classify the developed formulations according to two different models:

- The MIP in Sect. 7.4 defines each composite region by selecting a node as a center. The region's components are explicitly assigned to the center of the region. There are different possibilities to ensure contiguous regions in this model; we will discuss two of them in detail. The number of variables and constraints in this MIP is $\mathcal{O}(|V|^2)$ with V being the set of areas in the input map.
- A different approach is presented in Sect. 7.5. We will introduce a MIP that also selects nodes as centers of regions. However, the assignments of components to centers will not explicitly be expressed by variables. Instead, we define a flow on the adjacency graph of the planar subdivision that implies to which center a node belongs. As a consequence we get by with a linear number of variables and constraints.

Both approaches only allow subsets of the compactness measures that we discuss. We will discuss in detail which combinations of measures can be applied.

Section 7.6 presents heuristics that can be used in conjunction with the presented MIPs to speed up the processing. This includes a technique that decomposes a problem instance into smaller instances and so allows large datasets to be processed and updates to be handled efficiently. For the first presentation of this method refer to Haunert (2007a). Section 7.7 defines a neighborhood function for the problem and presents an alternative heuristic approach by simulated annealing that uses this neighborhood definition. For the original presentation of the simulated annealing method, refer to Haunert (2007c). Sect. 7.8 discusses possible combinations of the proposed methods.

7.1 Definition of the problem AreaAggregation

We define $G(V, E)$ to be the adjacency graph (the dual graph) of the planar subdivision, which contains a node for each region of the input map and an edge between two nodes, if the corresponding regions share a common boundary. A region in the target scale is defined by a pair (V', γ') with $V' \subseteq V$ and $\gamma' \in \Gamma$. Its class is defined by γ' and its shape can be obtained by union of shapes that correspond to nodes in V'. To aggregate the nodes into multiple regions we search a partition of V referred to as $P = \{V_1, V_2, \ldots, V_k\}$ with $k \in \{1, 2, \ldots, |V|\}$. The term partition implies that different parts do not intersect and the union of all parts equals V. Note that we do not assume that the number of output regions k is given in advance. We now give the formal problem statement. Figure 7.1 shows an example.

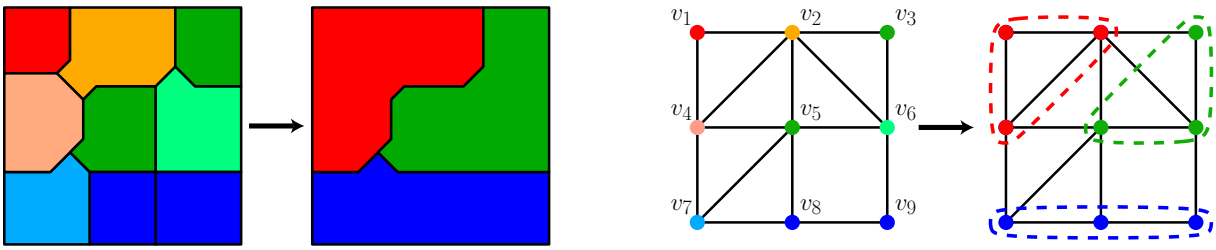
Problem (AreaAggregation)**.** Given a tuple $(G, w, \gamma, \theta, d, c, s)$ where:

- $G(V, E)$ is a planar graph,
- $w : V \to \mathbb{R}^+$ refers to node weights, i.e., area sizes,
- $\gamma : V \to \Gamma$ represents classes of nodes and Γ refers to the set of all classes of the map,
- $\theta : \Gamma \to \mathbb{R}^+$ refers to weight thresholds corresponding to minimal allowed sizes for classes,
- $d : \Gamma^2 \to \mathbb{R}_0^+$ expresses the semantic distance between classes,
- $c : 2^V \times \Gamma \to \mathbb{R}_0^+$ expresses the non-compactness of a composite region, and
- $s \in (0, 1)$ is a weight factor that expresses the trade-off between the objectives for minimum class change and compactness,

find a partition $P = \{V_1, V_2, \ldots, V_k\}$ of V and classes $\gamma'_1, \gamma'_2, \ldots, \gamma'_k \in \Gamma$ with $k \in \{1, 2, \ldots, |V|\}$, such that

- for each $i = 1, 2, \ldots, k$ (i.e., for each composite region) the following three requirements hold:
  (R1) the graph induced by V_i is connected (i.e., the composite region is contiguous),
  (R2) there is a node $v \in V_i$ with unchanged class, i.e., $\gamma'_i = \gamma(v)$, which we call the *center*, and
  (R3) V_i has total weight at least $\theta(\gamma'_i)$,
- and the cost $f = s \cdot f_{\text{class}} + (1 - s) \cdot f_{\text{comp}}$ is minimized, where:

  $f_{\text{class}} = \sum_{i=1}^{k} \sum_{v \in V_i} w(v) \cdot d(\gamma(v), \gamma'_i)$ is the cost for class change and
  $f_{\text{comp}} = \sum_{i=1}^{k} c(V_i, \gamma'_i)$ is the cost for non-compact shapes.



(a) map representation (b) graph representation

Fig. 7.1: An instance of AreaAggregation and a solution $P = \{\{v_1, v_2, v_4\}, \{v_3, v_5, v_6\}, \{v_7, v_8, v_9\}\}$, both displayed in map and graph representation. Assuming that all areas in the input have unit size, all aggregates satisfy the weight threshold $\theta = 3$.

In the following we call any region that fulfills the requirements (R1)–(R3), *feasible*. We call any region that fulfills requirement (R3), *weight-feasible*. Note that, according to the problem definition, each node is free to become a center of a region or not. The term connectivity is used instead of contiguity when discussing the problem in terms of graphs: a region with node set $V' \subseteq V$ is contiguous if and only if the subgraph of G induced by V' is connected. The cost function f corresponds to the objective (5.4), which was defined in Sect. 5.3.3 to express semantic accuracy. This combines the cost for class changes f_{class}, which is based on the function d, and the cost for non-compact shapes f_{comp}, which is based on the function c.

7.2 Compactness of shape

In Sect. 7.1 we expressed the objective for compact shapes in a very general way by the function c. We specify this definition here, in order to model the preferences of cartographers with respect to compactness. MacEachren (1985) gives a detailed discussion of different compactness measures. These could be expressed as objectives to be optimized in the area aggregation problem. However, we need to be careful in selecting a measure, as our choice will affect the solvability of the problem: with most measures, certain approaches, in particular mixed-integer programming, will become impossible. This is because most measures contain non-linear terms.

Due to this reason, only one of MacEachren's measures, which is based on the perimeter and the area of a shape, will be reviewed in Sect. 7.2.1. We will simplify this measure by neglecting the influence of the area (Sect. 7.2.2). In Sect. 7.2.3 we discuss another approach, which is based on distances to a center. This has been used to model compactness in several mixed-integer programs for districting problems. However, we define a new version of this measure that is based on shortest path lengths, instead of direct distances.

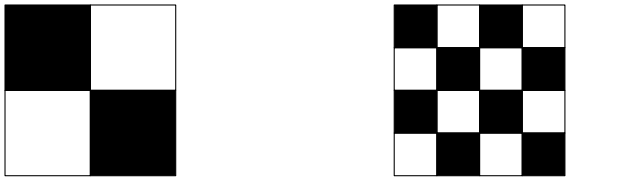
7.2.1 A classical perimeter-area-measure

The first measure investigated by MacEachren (1985) is:

$$c_1 = \frac{\text{perimeter}}{2 \cdot \sqrt{\pi \cdot \text{area}}}\,. \tag{7.1}$$

This results in $c_1 = 1$ for circles and in higher values for more complex shapes. Another important feature of this measure is its size invariance, which means that the value of c_1 does not change if a shape is scaled. These characteristics remain when applying an exponent to c_1, which is often done to charge very high costs for non-compact shapes.

Let's now discuss the setting $c := c_1$ for our global optimization problem: for each composite region we charge an individual cost equal to c_1. We consider two different solutions for the same instance, both containing regions that only differ in size. For example, one solution contains 4 square-shaped regions of size 2 m × 2 m, and the other solution contains 16 squares of size 1 m × 1 m (Fig. 7.2). If we now charge the same cost for the non-compactness of each composite region, the first solution with few large regions will be preferred, simply because a smaller number of equal penalties is charged.



(a) a partition with 4 shapes (b) a partition with 16 shapes

Fig. 7.2: Two partitions with shapes that only differ in size.

To define a measure that results in equal costs for the discussed partitions, we can charge an individual cost proportional to the size of a region, that is:

$$c_2 = \text{area} \cdot c_1 = \frac{1}{2\sqrt{\pi}} \cdot \sqrt{\text{area}} \cdot \text{perimeter}\,. \tag{7.2}$$

Indeed, setting $c := c_2$ seems to be a good choice, as the shapes in both solutions do not differ in terms of compactness. However, the equation contains a square root of a variable measure (the area of a region in the output) in a product with another variable (the perimeter). Unfortunately, we cannot express this using linear terms only. At this point, our problem clearly exceeds the possibilities of mixed-integer linear programming. However, in the next section we will simplify the measure, in order to get along with this technique.

Another possible approach would be to define the cost for a region by:

$$c_3 = \text{perimeter}^2. \tag{7.3}$$

As c_2, the measure c_3 is neutral with respect to the size of the shapes in the resulting partition. This measure could be expressed in a mixed-integer *quadratic* program, which consists of a quadratic objective function and linear constraints. General solutions techniques also exist for this class of problems (Lazimy, 1982). Their complexity, however, is much higher than in the linear case. Therefore, we do not follow this approach.

7.2.2 The perimeter as a measure of compactness

In order to introduce a linear approximation of the measure c_2, let's assume that the resulting regions in the output have a reasonable and constant size. In this case we can simply define:

$$c_{\text{peri}} = \text{perimeter}\,, \tag{7.4}$$

and set $c := c_{\text{peri}}$. Certainly our assumption is not very realistic, because, when minimizing the perimeter of areas, the method will prefer results with few large regions. In the extreme example that the objective for class change is neglected, the optimal result would be one big region containing all nodes. It is clear that this result is not wanted.

To cope with this, we could introduce additional requirements like an upper bound for the size of regions or a lower bound for the number of elements in the partition P. In Sect. 7.6.2 we propose an alternative heuristic approach that is based on a set of nodes that are predefined as centers. With this approach, we can avoid regions that are too large by defining that a region must not contain more than one center. Our assumption becomes much more realistic with this requirement, since a region cannot become too small due to the hard size constraints, and not too large, if the centers are appropriately defined.

To formally express c_{peri} as a function of a node set $V' \subseteq V$, we define the function $\lambda : E \to \mathbb{R}^+$, such that $\lambda(e)$ with $e = \{u, v\} \in E$ is equal to the length of the common boundary between the areas corresponding to u and v. The measure c_{peri} becomes:

$$c_{\text{peri}}(V') = \sum_{e \in E'} \lambda(e)\,, \tag{7.5}$$

with $V' \subseteq V$ and E' being the set of edges with one endpoint in V', that is:

$$E' = \big\{\{u, v\} \in E \;\big|\; u \in V' \wedge v \notin V'\big\}. \tag{7.6}$$

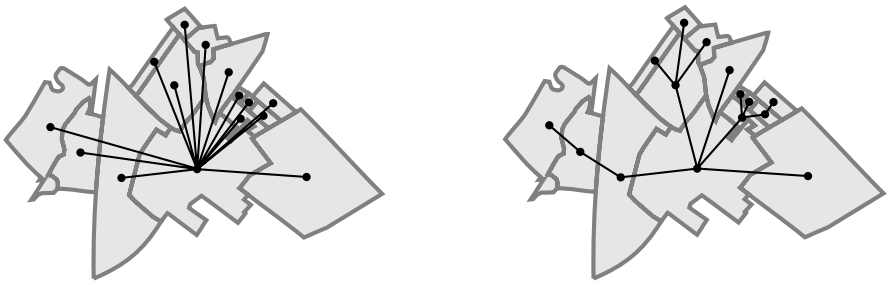
We neglect boundaries to the exterior of the dataset in this definition, simply because these boundaries are constant and cannot be optimized.

Though measures of compactness are often applied as rather abstract driving forces towards simpler shapes, the measure c_{peri} has a very concrete meaning: in case boundaries are drawn with dark color, minimizing the boundary length of the partition can be regarded as reducing the clutter on the map. This is a common aim of map generalization (Jansen & van Kreveld, 1998). Also the previously discussed iterative approaches to area aggregation in map generalization consider the length of boundaries as criterion when choosing a neighbor for merging (van Oosterom, 1995).

We can define a reasonable variant of the measure c_{peri} by setting $\lambda(e)$ equal to the number of line segments of the boundary. With this we can minimize the number of line segments in the partition, which is relevant for data compression. However, to produce well-generalized maps, we consider the length of the boundary to be more important.

7.2.3 Measures based on distances to a center

A measure of compactness that is commonly applied to districting problems is based on Euclidean distances between centroids (Cloonan (1972), Zoltners & Sinha (1983), Schröder (2001)). The region defined by the set $V' \subseteq V$ contains a node $u \in V'$ whose corresponding shape defines the *reference point* of the region by its centroid. The region is considered to be compact, if the centroids of all other nodes in V' are close to this.



(a) based on direct distances (c_{dist}) (b) based on shortest path lengths (c_{path})

Fig. 7.3: Compactness of a typical region based on distances to a center.

To respect different weights of nodes, a penalty is charged for each node, which is equal to the product of the node's weight and the distance of its centroid from the reference point. Figure 7.3(a) illustrates this measure for a region with several components. For each component, a line connecting the centroid with the reference point is displayed.

The measure is often applied to the problem of minimizing average travel distances to certain facilities like stores. For this problem, the measure has a very concrete meaning. Though the issue of travel distances is not relevant for the area aggregation problem, we can apply the measure here as well, yet with a more abstract meaning. Hess & Samuels (1971) use a similar compactness measure based on squared Euclidean distances. With this measure, higher penalties are charged for long distances.

To use the distance-based measures for the area aggregation problem, we define the reference point by the centroid of an area that corresponds to a node of unchanged class. In other words, we enforce that the center, according to the problem definition and the reference point, are defined by the same node. This definition will slightly ease the problem: we will need less variables to express the model. Nevertheless, the approach is reasonable, since it is undesirable for nodes with an unchanged class to only appear on the margin of a region. In contrast, it is preferable for nodes to 'gather around' the center of an unchanged class. If there are several nodes with this class, then among these the center is defined by the node for which the overall penalty is minimal.

It is clear that, according to this definition, the compactness for the same subset of nodes $V' \subseteq V$ is measured differently, if the corresponding region receives different classes. Because of this, we define c not only as a function of subsets of nodes, but also of classes.

To formalize this measure in a general way, we define the distance function $\delta : V^2 \to \mathbb{R}_0^+$ between pairs of nodes. With this, we define the measure $c_{\text{dist}} : 2^V \times \Gamma \to \mathbb{R}_0^+$ as:

$$c_{\text{dist}}(V', \gamma') = \min \left\{ \sum_{v \in V'} w(v) \cdot \delta(v, u) \quad \middle| \quad u \in V' \wedge \gamma(u) = \gamma' \right\}. \tag{7.7}$$

Defining $\delta(v, u)$ to be the Euclidean distance between the centroids, c_{dist} models the commonly applied measure to minimize travel distances. We will always use this setting when applying c_{dist}. To formalize the measure of Hess & Samuels (1971), we can define $\delta(v, u)$ as the squared Euclidean distance. Generally, any other function, for example, polynomials of higher degree, can be applied.

We can define another variant of c_{dist} with lengths of shortest paths instead of direct distances. For this, we embed the graph G in the plane such that each node is located at the centroid of the corresponding area and each edge is drawn with a straight line segment. The distance $\delta_{V'}$ between two nodes u and v of the same region $V' \in P$ is measured on this geometric graph. It is defined as the length of the shortest path connecting u and v without leaving the region V'. With this definition, the distance between two nodes depends on the set of nodes that is assigned to the same region, that is, it is different for different sets V'. However, we can formalize the measure in a straightforward way, replacing δ in Eq. (7.7) with $\delta_{V'}$. This yields:

$$c_{\text{path}}(V', \gamma') = \min \left\{ \sum_{v \in V'} w(v) \cdot \delta_{V'}(v, u) \quad \middle| \quad u \in V' \wedge \gamma(u) = \gamma' \right\}. \tag{7.8}$$

(a) based on direct distances (c_{dist}) (b) based on shortest path lengths (c_{path})

Fig. 7.4: Compactness of a hypothetical region based on distances to a center; squares depict optimal centers.

Figure 7.3(b) illustrates this measure. A black line between two nodes is drawn if the corresponding edge belongs to a shortest path between a node of the region and the center. Together these edges define the shortest path tree. The measure c_{path} is not defined if the center cannot be reached from a node of the same region via such a path. We can neglect this case, since connectivity is a hard constraint in the area aggregation problem.

Figure 7.4 illustrates the measures c_{dist} with direct distances (a) and c_{path} with shortest path lengths (b) for two hypothetical, spiral regions. Since average travel distances to the center are small when measured on direct lines, the regions will be considered relatively compact with c_{dist}. Conversely, the average length of the shortest path to the center is very high when allowing only for connections within regions. In the presented example, the measure c_{path} certainly better reflects the aim for compactness in map generalization. We will present a possibility to model this measure by means of linear expressions in Sect. 7.5.

Complex cases like the example in Fig. 7.4 will seldom appear in practice. The example that was presented in Fig. 7.3 certainly presents a much more realistic case. Because of this, we consider the measures c_{dist} and c_{path} similarly relevant.

Like c_{peri}, the measures c_{dist} and c_{path} are not size invariant, since distances to centers are shorter for smaller shapes. Nevertheless, the functions c_{dist} and c_{path} will attain high values for complex shapes. Presumably, both measures will only coarsely reflect the geometrical characteristics of a shape. However, we can combine them with c_{peri}. We suggest a combination of c_{dist} with c_{peri}, by introducing another weight factor $s' \in [0\,,1]$. With this we define a trade-off by:

$$c(V', \gamma') = s' \cdot c_{\text{dist}}(V', \gamma') + (1 - s') \cdot c_{\text{peri}}(V')\,. \tag{7.9}$$

7.3 Complexity of AreaAggregation

We now assess the complexity of AreaAggregation. We show that the problem is NP-hard even if we define certain simplifications. A simpler proof for a less restricted problem is given by Haunert & Wolff (2006).

First we define the decision version of AreaAggregation: Given an instance of AreaAggregation $(G, w, \gamma, \theta, d, c, s)$ and an integer C, does there exist a solution with cost C or less? If we can prove that this problem is NP-hard we also know that the optimization version of the problem is NP-hard.

Theorem 1. *Given an instance of* AreaAggregation $(G, w, \gamma, \theta, d, c, s)$ *and an integer C, it is NP-hard to decide whether a solution with cost at most C exists. This is still true if the number of classes is restricted to two, the threshold θ is defined to be the same for all classes, all nodes have equal weights, the distance between each two different classes is the same, and the compactness of shapes is not considered, that is, $s = 1$.*

As usual we prove NP-hardness by reduction from a problem that is known to be NP-hard. In our proof we reduce from PlanarVertexCover, which is known to be NP-complete (Garey *et al.*, 1974). We need to show that, for each instance of PlanarVertexCover $(G\,, C)$, we can construct an instance of the decision version of AreaAggregation, that is, $(G^*\,, w\,, \gamma\,, \theta\,, d\,, c\,, s)$ and the maximally allowed cost C^*, such that the answer to the second problem is 'yes' if and only if the answer to the first problem is 'yes'. This means that, if there was an efficient algorithm for AreaAggregation, we could efficiently solve PlanarVertexCover

as well. In other words, solving AreaAggregation is at least as hard as solving PlanarVertexCover. As PlanarVertexCover is NP-hard, AreaAggregation is NP-hard, too.

Certainly, there are other possibilities to prove NP-hardness of the area aggregation problem. To understand the proof it is not essential to comprehend how this particular reduction was found. In fact, several attempts were needed to develop the presented proof. For inexperienced readers it is advisable to first accept the way the instance of AreaAggregation is constructed and then to follow the reasoning why this approach works.

Proof. Given an instance of PlanarVertexCover, that is, a planar graph $G(V, E)$ and an integer C, we construct the input graph $G^*(V^*, E^*)$ for the aggregation problem as shown in Fig. 7.5. We define the set of classes $\Gamma = \{\text{black}, \text{white}\}$. For each node $v \in V$, we add three black nodes to V^*. We refer to an arbitrary one of them as the node corresponding to v. The three nodes are connected by edges forming a triangle. For each edge $\{u, v\} \in E$, we add $|V|$ white nodes to V^*. We connect each of these nodes by edges with the nodes corresponding to u and v. We define a unit weight $w = 1$ for each node and the threshold $\theta = 2$ for each class. For changing the class of a node, we charge one unit of cost, that is, we define $d(\text{white}, \text{black}) = d(\text{black}, \text{white}) = 1$ and $d(\text{white}, \text{white}) = d(\text{black}, \text{black}) = 0$. The maximally allowed cost is defined by $C^* = C$.

We now prove that there is a solution of the instance of AreaAggregation with cost C^* if and only if there is a vertex cover of G with cardinality C.

For each vertex cover of G with cardinality $\bar{C}$, there is a corresponding solution of the aggregation problem with cost $\bar{C}$: in G^*, we simply need to change the black nodes that correspond to the nodes in the vertex cover of G into white nodes. As the vertex cover includes an endpoint for each edge in E, all white nodes will get a white neighbor. For each triangle of black nodes, at least two adjacent nodes will remain black. Thus the subgraphs induced by the white and by the black nodes are all weight-feasible.

It remains to show that, for any solution of the aggregation problem with cost $\bar{C}$, there is a vertex cover of G with cardinality $\bar{C}$. Let's assume that no such vertex cover exists, that is, the cardinality of a minimum vertex cover of G is larger than $\bar{C}$. This implies that there is at least one edge $\{u, v\}$ in E such that the black nodes in V^* corresponding to u and v keep their colors (classes). In this case, we can only satisfy the weight thresholds by changing the color of all white nodes in V^* that were added for the edge $\{u, v\}$. This, however, is very expensive, that is, we need to spend a cost of $|V|$. As $|V|$ is the cardinality of a trivial vertex cover of G, that is, the one including all nodes in V, we have found a contradiction to our assumption.

Note that our reduction can be done in polynomial time. Therefore, we can say that PlanarVertexCover *polynomially reduces to* AreaAggregation. □

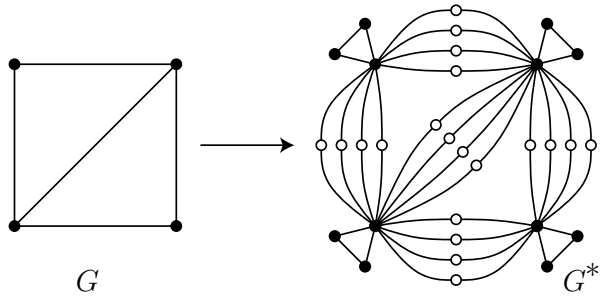


Fig. 7.5: Proof of hardness by reduction: An instance of PlanarVertexCover (left) and the corresponding instance of AreaAggregation.

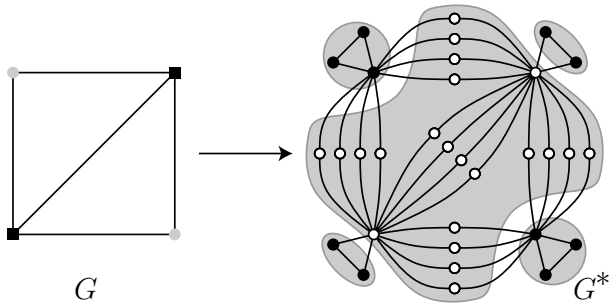


Fig. 7.6: A vertex cover of the graph G (square vertices, left) and the graph G^* with a corresponding solution of the AreaAggregation instance (right).

Again, Figures 7.5 and 7.6 show how the reduction works: let's assume that we want to know whether there is a vertex cover of G (Fig. 7.5, left) that has cardinality $C = 2$ or less. In order to solve this original problem we could construct the corresponding instance of AreaAggregation (Fig. 7.5, right) and ask whether it can be solved with cost $C^* = 2$. Indeed, there is such a solution (Fig. 7.6, right): exactly two nodes needed to change their class from black to white. Consequently, a hypothetical algorithm for AreaAggregation would

return the answer 'yes' and a list of re-colored nodes. This implies that there is also a vertex cover of G with cardinality 2 (Fig. 7.6, left), which is equal to the set of re-colored nodes in V^*. Thus, we can answer our original question with 'yes'. The solution of AreaAggregation implies a solution of PlanarVertexCover.

Since NP-hardness is proven for equal distances between classes, it is also assured for symmetric distances, that is, if $d(\gamma_1, \gamma_2) = d(\gamma_2, \gamma_1)$ for all γ_1 and γ_2 in Γ. In Sect. 5.3 we argued that the distance function d is indeed symmetric if we define its values based on lengths of shortest paths in a graph. The proof shows that the problem does not become easier in this case.

To show that the decision version of AreaAggregation is NP-complete, it remains to show that it is in $\mathcal{NP}$. Recall from Sect. 3.2.2 that problems in $\mathcal{NP}$ are in a way easier than other problems: if we had a procedure for guessing a solution, we could apply an efficient algorithm to verify this solution. Verifying a solution of AreaAggregation implies to check its feasibility and to calculate its costs. If this can be done in polynomial time, the problem is in $\mathcal{NP}$. The complexity of calculating the costs depends on the definition of the compactness function c. However, it is reasonable to assume that this can be assessed efficiently. This is certainly true for all compactness measures that were introduced in Sect. 7.2.

Proposition 1. *Given that, for any feasible region, the value of the function c can be calculated in polynomial time, the decision version of* AreaAggregation *is NP-complete.*

Proof. Assuming that we guessed a partition and new classes for the obtained regions, we can efficiently check whether this solution is feasible and we can efficiently calculate its cost. Consequently, the decision version of AreaAggregation is in $\mathcal{NP}$. Since it is NP-hard and in $\mathcal{NP}$, it is NP-complete. □

Since there is no hope of finding an exact polynomial time algorithm for AreaAggregation, we focus on heuristics and mixed-integer programming.

7.4 A MIP with assignments of nodes to centers

In order to introduce a MIP for AreaAggregation, we first consider a simplified problem. We neglect the hard constraint for contiguity of regions (requirement (R1)) and the objective for compact shapes, that is, we define $s := 1$. The remaining problem demands a partition of V into weight-feasible parts of uniform class, such that each part contains at least one node with its class unchanged. A solution of this problem can be defined by assigning nodes to centers. In this model the meaning of a center is twofold: it defines the new class for all its assigned nodes and, by this, also fixes the class-dependent weight threshold. Initially, all nodes are assumed to be potential centers, resulting in variables:

$$x_{uv} \in \{0, 1\}, \qquad \text{with } x_{uv} = 1 \text{ if and only if node } v \in V \text{ belongs to center } u \in V.$$

With these variables, we define the *basic IP* as follows:

$$\text{minimize} \quad \sum_{u \in V} \sum_{v \in V} w(v) \cdot d(\gamma(v), \gamma(u)) \cdot x_{uv} \tag{7.10}$$

subject to

$$\sum_{u \in V} x_{uv} = 1 \qquad \text{for all} \quad v \in V, \tag{7.11}$$

$$\sum_{v \in V} w(v) \cdot x_{uv} \geq \theta(u) \cdot x_{uu} \qquad \text{for all} \quad u \in V, \tag{7.12}$$

$$x_{uv} \leq x_{uu} \qquad \text{for all} \quad u, v \in V, u \neq v. \tag{7.13}$$

Constraint (7.11) ensures that each node is assigned to exactly one center. Constraint (7.12) enforces that all nodes that are assigned to the same center define a weight-feasible region. The last constraint is needed to allow only assignments to nodes that are assigned to themselves, in other words, to those that are selected as centers.

The basic IP has $\mathcal{O}(|V|^2)$ variables and constraints. Note that, by assigning nodes to classes and not to centers, a more compact formulation of the simplified problem would be possible. However, the extensibility of the model would be limited.

7.4.1 A heuristic approach: connectivity based on precedence relationship

In this section we discuss an extension of the basic IP that ensures connectivity, that is, contiguity of regions. The method is adopted from Zoltners & Sinha (1983) who investigated the problem of defining contiguous sales territories by assigning geographic units to given centers. Caro *et al.* (2004) use this method for school redistricting. In contrast to these approaches, the method will be applied here to solve the problem for an unknown set of centers.

Evidently, in order to end up with contiguous regions, a node can only be assigned to a distinct center, if at least one of its neighbors is assigned to the same center. This requirement is emphasized by Wright *et al.* (1983). However, it is important to note that this does not suffice. Consider two adjacent nodes being assigned to the same center: both nodes will mutually satisfy their requirements, without ensuring the connectivity to the center. To cope with this, we introduce an acyclic *precedence relationship*. With this, it will be possible to ensure connectivity.

We first define a directed graph $G'(V, A)$ with two converse arcs for each edge in E, that is,

$$A = \big\{uv \;\big|\; \{u, v\} \in E\big\}\,. \tag{7.14}$$

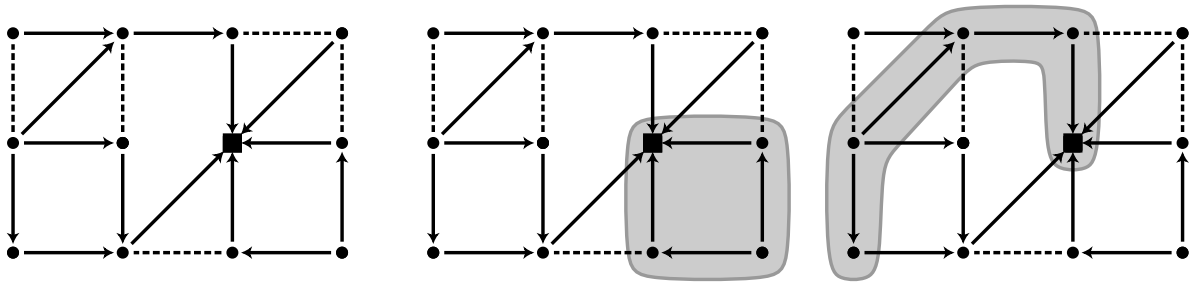
Additionally, we introduce arc lengths $l : A \to \mathbb{R}_0^+$. Given a center $u \in V$ and a node $v \in V$, such that $u \neq v$, we define the set of predecessors of v with respect to center u as:

$$\mathrm{Pred}_u(v) := \big\{w \in V \;\big|\; L(u\,, w) < L(u\,, v) \wedge \{v, w\} \in E\big\}\,,$$

with $L : V^2 \to \mathbb{R}_0^+$ being the length of the shortest path in G' between two nodes using arc lengths l.

Note that this definition of L is different from the definition of $\delta_{V'}$ in Sect. 7.2.3, because shortest paths are not constrained to be within regions of the resulting map. In contrast, $L(u, v)$ is constant for given nodes u and v and can be computed in advance for all pairs of nodes. This problem is commonly referred to as the "All Pairs Shortest Path Problem". A simple solution of this problem is to apply $|V|$ times the algorithm of Dijkstra (1959). With this approach, we can solve the problem in $\mathcal{O}(|V|^2 \log |V|)$ time.

Different possibilities exist for defining the edge lengths. The simplest is to define $l(a) := 1$ for all $a \in A$. An example of the precedence relationship with this setting is illustrated in Fig. 7.7(a). Arcs are drawn from each node to its predecessors. The resulting directed graph is acyclic and the center is the only sink. For some edges of the adjacency graph, both endpoints have the same distance to the center. These edges are displayed as dashed lines. However, when defining non-uniform arc lengths, these cases are rare exceptions.



(a) precedence relationship with respect to a center (displayed as square)

(b) a feasible region

(c) a non-feasible region

Fig. 7.7: Precedence relationship and the feasibility of regions with constraint (7.15).

Before defining our setting for l, we introduce the constraint for connectivity. The basic IP from Sect. 7.4 can be modified by replacing constraint (7.13) with the constraint:

$$x_{uv} \leq \sum_{w \in \mathrm{Pred}_u(v)} x_{uw} \qquad \text{for all} \quad u\,, v \in V\,, u \neq v\,. \tag{7.15}$$

This constraint means that a node v can only be assigned to a center u, if at least one of its predecessors with respect to u is also assigned to this center. The constraint clearly forbids disconnected regions, since the center can be reached from any assigned node via predecessors without leaving the region. Figure 7.7(b) shows an example that satisfies constraint (7.15).

In addition, by applying constraint (7.15), several contiguous regions will be excluded. An example of this is displayed in Fig. 7.7(c). The region does not contain any predecessor of the node located in the lower left corner. By defining another center, the same set of nodes might constitute a feasible region, since the precedence relationship is defined individually for each potential center. Nevertheless, we cannot guarantee the feasibility of contiguous regions. Zoltners & Sinha (1983) legitimate the restriction of their model with their preference for compact sales districts: the non-feasible contiguous regions are likely to be non-compact. So, they can probably be excluded without loosing good solutions. As compactness is also an objective of map generalization, the application of this strong connectivity definition is a promising approach. Since the optimal solution might be missed using this, we will refer to it as a heuristic approach.

The definition of the arc lengths certainly affects the validity of this method. A reasonable definition is:

$$l(uv) = w(v)\,. \tag{7.16}$$

With this definition, the size of the smallest contiguous region containing two nodes can be expressed by a function $w_{\min} : V^2 \to \mathbb{R}_0^+$, with:

$$w_{\min}(i,j) = w(i) + L(i,j)\,. \tag{7.17}$$

Since the first summand $w(i)$ is constant for a fixed center i, Eq. (7.16) defines a node to be close to the center, if its minimum region with i is small.

Estimating the consequences of the requirement for strong connectivity is very difficult. Because of this, we will introduce a model in the next section that guarantees connectivity in a general sense. Comparisons of the results obtained with the proposed methods will show whether the restrictions are acceptable, when applying the defined setting. These tests will be presented in Chapter 8.

In the following sections, the MIP made up by the objective function in (7.10) and the constraints in (7.11), (7.12), and (7.15) will be called the *P-R MIP*. It only comprises binary variables, but we will introduce continuous variables in Sect. 7.4.4 to factor in compactness. Appendix A shows the P-R MIP for a small problem instance.

7.4.2 An exact approach: connectivity based on a spanning tree

This section introduces an extension to the basic IP that models connectivity in a general sense. The method is based on an idea of Williams (2002) for the problem of optimally selecting a set of nodes from a planar graph such that the selected node set induces a connected subgraph. In contrast to this problem, area aggregation aims to group all nodes into several parts, each defining a contiguous region. We show that an extension of the method by Williams is possible, but the size of the IP will be quadratic and not linear in the number of nodes.

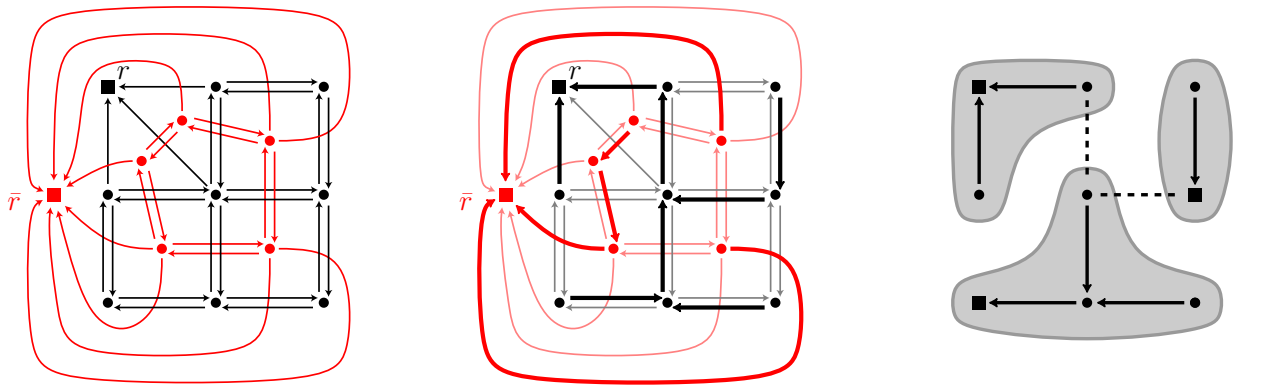
The method of Williams can be outlined as follows: the selected nodes define the node set of a digraph. In order to ensure connectivity, this digraph is enforced to be a subtree of a spanning tree T of G. Cycles in this spanning tree are forbidden by introducing a spanning tree $\bar{T}$ on the dual graph $\bar{G}$ of G. By claiming that either an edge is part of T, or its dual edge is part of $\bar{T}$, cycles in T and $\bar{T}$ become impossible.

To explain the approach for the area aggregation problem in detail, we apply the definition of arcs A according to Eq. (7.14). Evidently, a region with nodes $V' \subseteq V$ is contiguous, if and only if there is a subset $A' \subseteq A$, such that A' defines a spanning tree of V'. To express the selection of an arc a for the spanning tree of the region with center u, variables $t_{ua} \in \{0,1\}$ are introduced. Constraints will be imposed on these variables to ensure that arcs with $t_{ua} = 1$ form a tree with root u and that all nodes of the tree are assigned to u.

Constraint (7.18) ensures that, if and only if a node v is assigned to center u with $v \neq u$, then there is one outgoing arc of v that is selected for the center's tree. Constraint (7.19) defines that, if node v does not belong to center u, there is no incoming arc that is selected for the tree with root u.

$$\sum_{a=vw \in A} t_{ua} = x_{uv} \qquad \text{for all} \quad u\,, v \in V\,, u \neq v \tag{7.18}$$

$$t_{ua} \leq x_{uv} \qquad \text{for all} \quad u\,, v \in V, a = wv \in A \tag{7.19}$$



(a) initial setting: displayed arcs can be selected for spanning trees on G (black) and its dual $\bar{G}$ (red)

(b) a spanning tree T (arcs with $T_a = 1$ bold black) and its corresponding tree $\bar{T}$ on the dual graph (arcs with $\bar{T}_{\bar{a}} = 1$ bold red)

(c) a partition of V with selected arcs, i.e., arcs with $t_{ua} = 1$ for centers u (squares); constraint (7.24) restricts selectable arcs to T (dashed)

Fig. 7.8: Reduction of G to a spanning tree T according to Williams (2002) and a partition of T into subtrees.

It remains to forbid cycles in the digraphs of arcs that are selected for the same center. We can exclude cycles by defining a spanning tree T on G and a spanning tree $\bar{T}$ on the dual graph $\bar{G}(\bar{V}, \bar{E})$ of G. Two arbitrary nodes $r \in V$ and $\bar{r} \in \bar{V}$ are defined as roots of T and $\bar{T}$, respectively. Variables $T_a \in \{0,1\}$ with $a = uv \in A\,, u \neq r$, are introduced to express whether arc a is selected for T or not. The same is done for the dual tree, that is, variables $\bar{T}_{\bar{a}} \in \{0,1\}$ with $\bar{a} = \bar{u}\bar{v} \in \bar{A}\,, \bar{u} \neq \bar{r}$ and $\bar{A} = \{\bar{u}\bar{v} \mid \{\bar{u}, \bar{v}\} \in \bar{E}\}$ are introduced. Similar to constraint (7.18), constraints (7.20) and (7.21) claim that, for each node in $V \setminus \{r\}$ there is one outgoing arc in T and for each node in $\bar{V} \setminus \{\bar{r}\}$ there is one outgoing arc in $\bar{T}$.

$$\sum_{a=uv \in A} T_a = 1 \qquad \text{for all} \quad u \in V \setminus \{r\} \tag{7.20}$$

$$\sum_{\bar{a}=\bar{u}\bar{v} \in \bar{A}} \bar{T}_{\bar{a}} = 1 \qquad \text{for all} \quad \bar{u} \in \bar{V} \setminus \{\bar{r}\} \tag{7.21}$$

Constraint (7.22) expresses that, for each edge $e = \{u, v\} \in E$, one of the arcs $a = uv$ and $b = vu$ belongs to T, or one of the arcs $\bar{a} = \bar{u}\bar{v}$ and $\bar{b} = \bar{v}\bar{u}$ belongs to $\bar{T}$, with $\bar{e} = \{\bar{u}, \bar{v}\}$ being the dual edge of e.

$$T_{a=uv} + T_{b=vu} + \bar{T}_{\bar{a}=\bar{u}\bar{v}} + \bar{T}_{\bar{b}=\bar{v}\bar{u}} = 1 \qquad \text{for all} \quad e = \{u, v\} \in E, \quad \bar{e} = \{\bar{u}, \bar{v}\} = \text{Dual}(e) \tag{7.22}$$

This ensures that neither T nor $\bar{T}$ contain any cycle. The effect of this constraint is illustrated in Figures 7.8(a) and 7.8(b). We define the expressions T_a and $\bar{T}_{\bar{a}}$ with $a = ru \in A$ and $\bar{a} = \bar{r}\bar{u} \in \bar{A}$, which appear in this constraint, as constants with value zero. This means that we cannot select arcs leaving the roots for the trees.

Finally, constraints (7.23) and (7.24) exclude cycles in the digraphs that were defined by variables t_{ua} in the beginning. The first expression explicitly forbids cycles that only consist of two converse arcs. The second expression forbids bigger cycles by claiming that an arc $a = uv$ can only be selected, if itself or its opponent $b = vu$ belongs to the tree T. By this, we assure that the selected arcs form subtrees of T as being shown in Fig. 7.8(c).

$$t_{ua} + t_{ub} \leq 1 \qquad \text{for all} \quad u \in V, a = vw \in A, b = wv \tag{7.23}$$

$$t_{ua} \leq T_a + T_b \qquad \text{for all} \quad u \in V, a = vw \in A, b = wv \tag{7.24}$$

Expressions (7.10)–(7.12) and (7.18)–(7.24) constitute an IP for the area aggregation problem that allows for connectivity in a general sense. We refer to this IP as the *Tree MIP*. Just as the P-R MIP it only contains binary variables, but again we will introduce continuous variables to model compactness. The Tree MIP required the introduction of many auxiliary binary variables, but still it has a quadratic size. In order to reduce the computation time, we will introduce a MIP of linear size in Sect. 7.5. First, however, we discuss how to express compactness in the P-R MIP and the Tree MIP.

7.4.3 Expressing compactness according to the distance measure c_{dist}

To minimize direct distances to centers, we get along with the variables of the basic IP. With the variables x_{uv} we can express the measure c_{dist} from Eq. (7.7) simply as follows:

$$\text{minimize} \quad \sum_{u \in V} \sum_{v \in V} w(v) \cdot \delta(v,u) \cdot x_{uv} \,. \tag{7.25}$$

For a single node v, a cost equal to $w(v) \cdot \delta(v,u)$ is charged, with u being the center to which v is assigned, that is, $x_{uv} = 1$. Given a region $V' \in P$ and a class $\gamma' \in \Gamma$, one is free to choose any node $u \in V'$ with $\gamma(u) = \gamma'$ as center. By minimizing the expression in Eq. (7.25), the center will be chosen among them, such that the overall penalty is minimal. Therefore, this formulation correctly reflects c_{dist}. Since $w(v) \cdot \delta(v,u)$ is constant for given nodes u and v, the requirement for linear terms in the objective function is clearly satisfied. This is also true for direct distances with exponents different to one. However, to define the length of the shortest connecting path within the region, we need to know which other nodes belong to the same region. Therefore, we cannot simply adapt the model to express the measure c_{path}.

7.4.4 Expressing compactness according to the perimeter measure c_{peri}

To express the cost for perimeters of regions, we need additional auxiliary variables:

$$y_{ue} \in [0,1] \,, \quad \text{with } y_{ue} = 0, \text{ if at least one endpoint of } e \in E \text{ does not belong to center } u \in V \,.$$

We define constraint (7.26) to ensure this property of the variables y_{ue}, that is, y_{ue} with $e = \{v,w\}$ is set to 0, if $x_{uv} = 0$ or $x_{uw} = 0$:

$$\left.\begin{array}{rcl} y_{ue} & \leq & x_{uw} \\ y_{ue} & \leq & x_{uv} \end{array}\right\} \quad \text{for all } u \in V, e = \{v,w\} \in E \tag{7.26}$$

Charging a cost for the perimeters of regions can also be regarded as giving a benefit to edges contained in regions. With the variables y_{ue} we can choose this approach. Let's first express this alternative view for the total cost for perimeters. The sum of perimeters in the result is equal to the sum of perimeters in the input minus twice the length of boundaries that are eliminated by the aggregation, that is,

$$\sum_{i=1}^{k} c_{\text{peri}}(V_i) = \sum_{v \in V} c_{\text{peri}}(\{v\}) - 2 \cdot \sum_{e \in E^*} \lambda(e) \,, \tag{7.27}$$

with $E^* \subseteq E$ being the set of edges that are contained in a region and the boundary lengths $\lambda : E \to \mathbb{R}_0^+$ according to the definition in Sect. 7.2.2. As the first term on the right-hand side of Eq. (7.27) is constant, we only need to minimize the second term. This can be expressed by:

$$\text{minimize} \;\; -2 \cdot \sum_{u \in V} \sum_{e \in E} \lambda(e) \cdot y_{ue} \,. \tag{7.28}$$

Since a benefit (a negative cost) is given proportional to y_{ue}, the variable y_{ue} always attains the value of its upper bound. Hence y_{ue} is one for edges $e \in E$ included in the region with center u. We keep the factor 2 in objective (7.28). This allows the combination of c_{dist} and c_{peri} (Eq. (7.9)) to be expressed using a combination of the objectives (7.25) and (7.28), simply by applying the weight factor s'.

7.4.5 Optimizing multiple objectives

In order to find a good solution, we need to apply a combination of multiple objectives. We choose the combination of c_{dist} and c_{peri} in Eq. (7.9) to measure compactness. The resulting cost for non-compact shapes

f_{comp} is combined with the cost for class change f_{class} as defined in Eq. (5.4). We can express this combined objective in the P-R MIP and the Tree MIP as follows:

$$\begin{aligned} \text{minimize} \quad & s \cdot \sum_{u \in V} \sum_{v \in V} w(v) \cdot d(\gamma(v), \gamma(u)) \cdot x_{uv} \\ & + (1-s) \cdot s' \cdot \sum_{u \in V} \sum_{v \in V} w(v) \cdot \delta(v,u) \cdot x_{uv} \\ & - 2 \cdot (1-s) \cdot (1-s') \cdot \sum_{u \in V} \sum_{e \in E} \lambda(e) \cdot y_{ue}. \end{aligned} \tag{7.29}$$

Obviously, the first line and the second line in objective (7.29) can be summarized, since both contain the same variables. However, we leave the formulation in the presented form to better distinguish different objectives. Since the formulation includes binary and fractional variables, it defines a mixed-integer program. The objective (7.29) is a fundamental part of the proposed aggregation method. It has been tested using different settings for s and s'. The results of these tests will be presented and discussed in Chapter 8. First, however, we discuss another MIP, which has a linear size.

7.5 An exact approach: a MIP of linear size based on a flow model

An approach to the problem of selecting an optimal, connected subset of nodes from a graph is given by Shirabe (2005b). After a brief explanation of this method, a new approach is presented that takes up the general idea.

7.5.1 The flow model of Shirabe (2005b)

The method of Shirabe (2005b) is based on the definition of arcs in Eq. (7.14). Instead of binary variables that express whether an arc is selected, nonnegative fractional variables are introduced that define a flow on an arc. This flow can be regarded as the amount of a commodity being transported between two nodes. Connectivity can be ensured with the following constraints that control the incoming and outgoing flow for each node:

- The center of the selected region is defined to be the only sink in the network, meaning that there is more incoming than outgoing flow.
- All other nodes that are selected for the region are sources that contribute a positive amount of flow.
- Other nodes do not receive any flow.

Figure 7.9 shows an example of a resulting network. The network of arcs with positive flow can contain cycles and does not necessarily define a tree. However, connectivity is ensured since all inserted flow will reach the center without leaving the selected region.

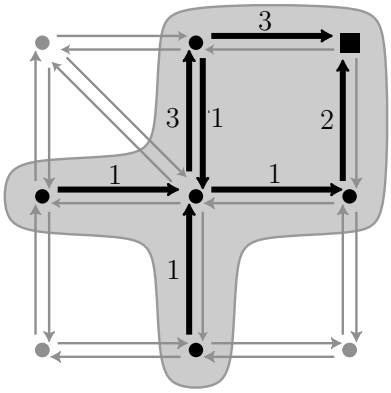


Fig. 7.9: Connectivity based on the flow model of Shirabe (2005b). Each source (circles) contributes a flow of one unit. There is a single sink (square), that is, a node that consumes a positive amount of flow.

The method of Shirabe (2005b) can be adopted for the area aggregation problem, that is, to produce multiple connected regions: multiple variables could be defined for each arc, here, to express a separate flow for each potential center. With this multi-commodity flow model, the basic IP from Sect. 7.4 could be extended, again resulting in an IP of quadratic size. Since this extension of Shirabe's model is rather straightforward, it will not be presented here in detail. However, there is a more interesting extension of this model, which manages with a single-commodity flow. It is very likely that this will result in a significant decrease of computation time, since the resulting MIP has a linear size.

7.5.2 A new flow model for AreaAggregation

The newly developed MIP presented in this section uses the definition of flow on the graph G. However, in contrast to the approach of Shirabe (2005b), this is not only used to ensure connectivity, but also to ensure weight-feasible regions and to express the objective of compact shapes. In order to achieve a MIP of linear size, nodes will be assigned to classes and not to centers. This does not allow the compactness measure c_{dist} from Sect. 7.4.4 with direct distances, but it is possible to express the similar measure c_{path} with shortest path lengths. Before discussing this possibility, let's investigate the special case when the sole objective is to minimize changes of classes. After a brief explanation of the idea, the formulation is presented in detail. Figure 7.10 illustrates this flow model.

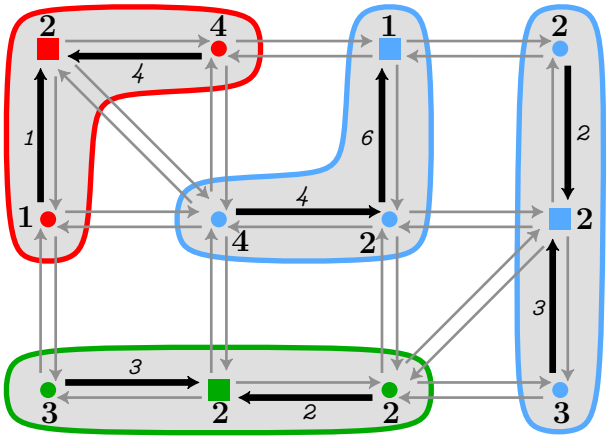


Fig. 7.10: Connectivity based on our flow model. Arcs with positive flow ($f_a > 0$ and $F_a = 1$) are displayed in bold black. Node weights are displayed by bold numbers, values of the flow variables f_a by italic numbers. Sinks ($s_v = 1$) are displayed by squares, sources ($s_v = 0$) by circles. All aggregates exactly satisfy the weight threshold $\theta = 7$.

Flow variables $f_a \in \mathbb{R}^+$ define the amount of a commodity being transported on arcs $a \in A$, with $A = \{uv \mid \{u, v\} \in E\}$. The arcs with positive flow define a directed graph. The weakly connected components of this graph yield the regions in our model. To indicate whether an arc carries a positive flow, a binary variable F_a is introduced for each arc $a \in A$. The variable F_a is set to 1, if $f_a > 0$. Introducing binary variables b_{vc}, we express whether a node $v \in V$ receives class $c \in \Gamma$. To ensure that all nodes in a region receive the same class, we enforce that two adjacent nodes should receive the same class, if a connecting arc carries positive flow. For each node $v \in V$, a binary variable s_v is introduced, to express whether it is a sink or a source of the network. Sources contribute a positive amount of flow to the network that is equal to their weight. Thus, it is not possible to create a connected component that only contains sources, and so, fixing the class of a sink, we are sure to have at least one node of unchanged class in each region. Regions of sufficient size are obtained by claiming that a sink receives a certain amount of flow: the sum of this flow and the sink's own weight must not fall below the weight threshold defined for the class of the sink.

Let's investigate the defined requirements for the flow in the example of Fig. 7.10: the node in the upper left corner is a sink, whose own weight is two. In order to satisfy the weight threshold of seven, the incoming flow needs to exceed the outgoing flow by at least five units. In this example, the requirement is exactly fulfilled. The node in the second row and third column of the grid-like graph is a source. It receives a flow of four and

emits a flow of six, thus its net outflow is two, which equals its weight. Since these constraints are fulfilled for all nodes, the resulting partition is feasible.

The following list summarizes the introduced variables:

$f_a \in [0, M]$	flow on arc $a \in A$, with $M = \sum_{v \in V} w(v)$ being the total weight of the graph.
$F_a \in \{0, 1\}$	$F_a = 1$ if arc $a \in A$ carries positive flow.
$b_{vc} \in \{0, 1\}$	$b_{vc} = 1$ if and only if node $v \in V$ receives class $c \in \Gamma$.
$s_v \in \{0, 1\}$	$s_v = 1$ if and only if node $v \in V$ is a sink.

With the variables b_{vc}, the objective for minimum class change is defined as follows:

$$\text{minimize} \quad \sum_{v \in V} \sum_{c \in \Gamma} w(v) \cdot d(\gamma(v), c) \cdot b_{vc}. \tag{7.30}$$

To link the variables F_a and f_a, such that $F_a = 1$ if $f_a > 0$ we add the following constraint:

$$M \cdot F_a \geq f_a \qquad \text{for all} \quad a \in A\,. \tag{7.31}$$

The next constraint is similar to constraint (7.11) in the basic IP. Instead of claiming that each node is assigned to one center, each node is assigned to one class:

$$\sum_{c \in \Gamma} b_{vc} = 1 \qquad \text{for all} \quad v \in V \tag{7.32}$$

Constraint (7.33) ensures that two adjacent nodes receive the same class if a connecting arc carries positive flow.

$$b_{uc} \geq b_{vc} + (F_{uv} - 1) \qquad \text{for all} \quad uv \in A, c \in \Gamma \tag{7.33}$$

To ensure that a sink keeps its original class, the following constraint is added:

$$b_{v,\gamma(v)} \geq s_v \qquad \text{for all} \quad v \in V \tag{7.34}$$

The next two constraints ensure the connectivity and weight-feasibility of regions with equal class. We will discuss their effect in detail.

$$\sum_{a=vu \in A} f_a - \sum_{a=uv \in A} f_a \geq w(v) - s_v \cdot M \qquad \text{for all} \quad v \in V \tag{7.35}$$

$$\sum_{a=vu \in A} f_a - \sum_{a=uv \in A} f_a \leq w(v) - s_v \cdot \theta(\gamma(v)) \qquad \text{for all} \quad v \in V \tag{7.36}$$

We consider first the case that node v is a source, meaning $s_v = 0$. Then, the two inequalities can be summarized by one equation:

$$\sum_{a=vu \in A} f_a - \sum_{a=uv \in A} f_a = w(v) \qquad \text{for all} \quad v \in V$$

This means just that the net outflow from a source equals the node's weight. In the case that $s_v = 1$, meaning that v is a sink, constraint (7.35) is relaxed and constraint (7.36) becomes:

$$\sum_{a=uv \in A} f_a - \sum_{a=vu \in A} f_a + w(v) \geq \theta(\gamma(v)) \qquad \text{for all} \quad v \in V,$$

meaning that the sum of the net inflow to a sink and its own weight, is not smaller than the threshold. So, the requirement for the weight of a contiguous region with equal class is fulfilled.

Figure 7.11 shows a second, more complex feasible solution of the MIP with constraints (7.31)–(7.36), revealing two facts: the arcs with positive flow can define cycles and a region can contain more than one sink, if it includes more than one node of unchanged class. Regions with multiple centers which have different classes, are not feasible. A region with K sinks is at least K times larger than the weight threshold for its class. However, such

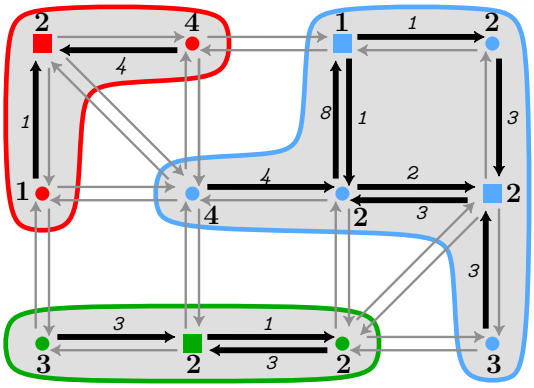


Fig. 7.11: Connectivity based on our flow model without constraint (7.37). The graph induced by arcs with positive flow can contain cycles; a region can contain more than one center. Labels have the same meaning as in Fig. 7.10.

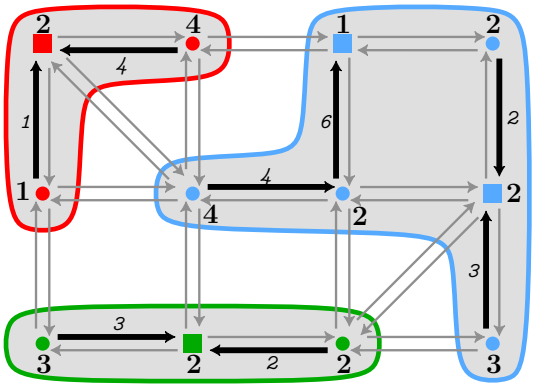


Fig. 7.12: Definition of the aggregation result from a MIP solution, when expressing compactness by the boundary length. Though there is no exchange of flow between two blue components, both components define a single region.

a region cannot necessarily be split into K feasible regions, which is simply because the flow coming from a source can be split and run into different sinks.

Cycles and multiple sinks in a single region are unproblematic, because, in any case, the resulting partition defines a feasible solution to the area aggregation problem. However, we define one more constraint, ensuring that, for each region, the arcs with positive flow define a tree whose root is the only sink of the region. A simple possibility to ensure this is to require that, for each source, there is at most one outgoing arc with positive flow and, for each sink, there is none:

$$\sum_{a=uv \in A} F_a \leq 1 \qquad \text{for all} \quad u \in V. \tag{7.37}$$

The solution presented in Fig. 7.10 satisfies this constraint. Adding constraint (7.37), we observed a small improvement in terms of performance. The feasibility of potential aggregation results is not affected.

7.5.3 Considering compactness according to c_{path} and c_{peri}

The flow variables f_a can be used to express the compactness measure c_{path}, which is based on lengths $\delta_{V'}$ of shortest paths that are constrained to a region V'. To explain this, let's observe a flow from its source v to the sink: for each arc a of the taken path, the variable f_a increases by $w(v)$. With constraint (7.37) there is only one sink per region. Because of this, we can express the aim for compactness as follows:

$$\text{minimize} \qquad \sum_{a=uv \in A} \delta(u,v) \cdot f_a. \tag{7.38}$$

Optimally solving the MIP with this objective, the selected tree of arcs with positive flow will automatically become equal to the region's shortest path tree – otherwise the solution will not be optimal. The resulting cost for a region is just the same as c_{path} as defined in Eq. (7.8).

To also measure the boundary length c_{peri} in Eq. (7.5), we need to detect whether or not two adjacent nodes belong to the same region. Generally, the variables b_{vc} do not allow this question to be answered: in the optimal solution two adjacent nodes can belong to different regions, though they receive the same class. In other words, the case $b_{vc} = b_{uc} = 1$ for some $c \in \Gamma$ does not tell us whether the boundary corresponding to the edge $\{u, v\}$ remains in the generalized map. We could use the variables x_{uv} from Sect. 7.4 instead, but with these variables the resulting MIP would again have a quadratic size, meaning that the advantage of the presented formulation would be lost.

However, for the special case when c_{peri} is applied as the only compactness measure, that is, it is not combined with c_{path}, a simpler extension is possible. In this case, we can assume without loss of generality that, in the

optimal aggregation result, no two adjacent regions have the same class. When merging such regions, the cost for the boundary length will always decrease, while the cost for class change will not change. This approach requires a slightly different translation of the optimal MIP-variables into the optimal aggregation result. The optimal partition is defined by the connected components of the graphs that are induced by nodes of equal class, as shown in Fig. 7.12.

The formulation of this objective is very similar to the formulation in Sect. 7.4.4. Instead of variables y_{ue}, we introduce variables b'_{ce}:

$$b'_{ce} \in [0,1]\,, \quad \text{with } b'_{ce} = 0, \text{ if at least one endpoint of } e \in E \text{ does not receive class } c \in \Gamma.$$

Note that for a constant number of classes, the number of variables b'_{ce} is linear, while the number of variables y_{ue} was quadratic in the input size. The variables b'_{ce} can be linked to the variables b_{uc}, as it was done in (7.26).

$$\left.\begin{array}{rcl} b'_{ce} & \leq & b_{vc} \\ b'_{ce} & \leq & b_{wc} \end{array}\right\} \quad \text{for all} \quad c \in \Gamma, e = \{v,w\} \in E. \tag{7.39}$$

We can formalize the objective of shortest boundaries as follows:

$$\text{minimize} \qquad -2 \cdot \sum_{c\in\Gamma} \sum_{e\in E} \lambda(e) \cdot b'_{ce}. \tag{7.40}$$

7.5.4 Summary and discussion of applicability

The presented formulation can be applied to optimize class change in combination with one of the two compactness measures c_{path} or c_{peri}.

In the case that $c = c_{\text{path}}$ is applied, the objective is defined by:

$$\begin{aligned} \text{minimize} \qquad & s \cdot \sum_{v\in V} \sum_{c\in\Gamma} w(v) \cdot d(\gamma(v), c) \cdot b_{vc} \\ & +(1-s) \cdot \sum_{a=uv\in A} \delta(u,v) \cdot f_a. \end{aligned} \tag{7.41}$$

In this case, the regions of the solution are defined by connected components of the graph that is comprised of arcs with $F_a = 1$. We refer to this objective, together with constraints (7.31)–(7.37), as the *Flow MIP*.

The following expression defines the objective for the case that c_{peri} is applied:

$$\begin{aligned} \text{minimize} \qquad & s \cdot \sum_{v\in V} \sum_{c\in\Gamma} w(v) \cdot d(\gamma(v), c) \cdot b_{vc} \\ & -2 \cdot (1-s) \cdot \sum_{c\in\Gamma} \sum_{e\in E} \lambda(e) \cdot b'_{ce}. \end{aligned} \tag{7.42}$$

Here, the regions of the solution are defined by connected components of the graphs induced by nodes of the same class.

In the case that both compactness measures are to be combined, the presented formulation is not sufficient to model the problem. In this case, we need to model explicitly whether or not a node belongs to a region. This can be achieved with the variables x_{uv} from Sect. 7.4, but will result in a MIP of quadratic size.

However, the Flow MIP is an important achievement, since it can be used to test the effects of heuristics for certain settings of the objective function. Both the P-R MIP and the Flow MIP can be used to solve the aggregation problem with the aim of minimum class change. The results will give insight about the effects of the requirement for strong connectivity and other heuristics.

Additionally, the Flow MIP presents a novel solution for other problems, for example, the minimization of travel distances that are constrained to the interior of districts.

7.6 Specialized heuristics

Due to the NP-hardness of the area aggregation problem, we cannot hope for an efficient and exact algorithm. This section presents heuristics that have been developed to solve problem instances of considerable size, and to update the generalized dataset without processing the entire dataset again. A heuristic approach has already been introduced in Sect. 7.4.1, with the definition of a strong connectivity requirement based on a precedence relationship. We define additional heuristics in this section to eliminate certain variables in the developed mixed-integer programs and to decompose a problem instance into smaller instances. The independent problem instances can be solved in parallel, for example, in order to speed up the processing of large datasets. Furthermore, if the source dataset is updated, only the affected problem instances need to be solved again.

The first heuristic is to assume that two areas will not become part of the same region if their distance is large (Sect. 7.6.1). The second heuristic is to assume that centers of regions and consequently their classes are defined by dominant, large areas (Sect. 7.6.2). This heuristic allows a dataset to be decomposed into independent problem instances. The definition of centers also leads to an idea for an iterative algorithm, which is explained in Sect. 7.6.3.

7.6.1 Distance heuristic

To eliminate variables x_{uv} in the Tree IP and the P-R MIP, we can use the fact that it is very unlikely that two nodes are merged in the same region, if their distance is large.

For this we use the minimum weight $w_{\min}$ of a region containing two nodes; this was introduced in Eq. (7.17) to define the distance for the precedence relationship. For each node $i \in V$, we define the set of nodes

$$S_i = \left\{ j \in V \;\middle|\; w_{\min}(i,j) < \theta(\gamma(i)) \right\}. \tag{7.43}$$

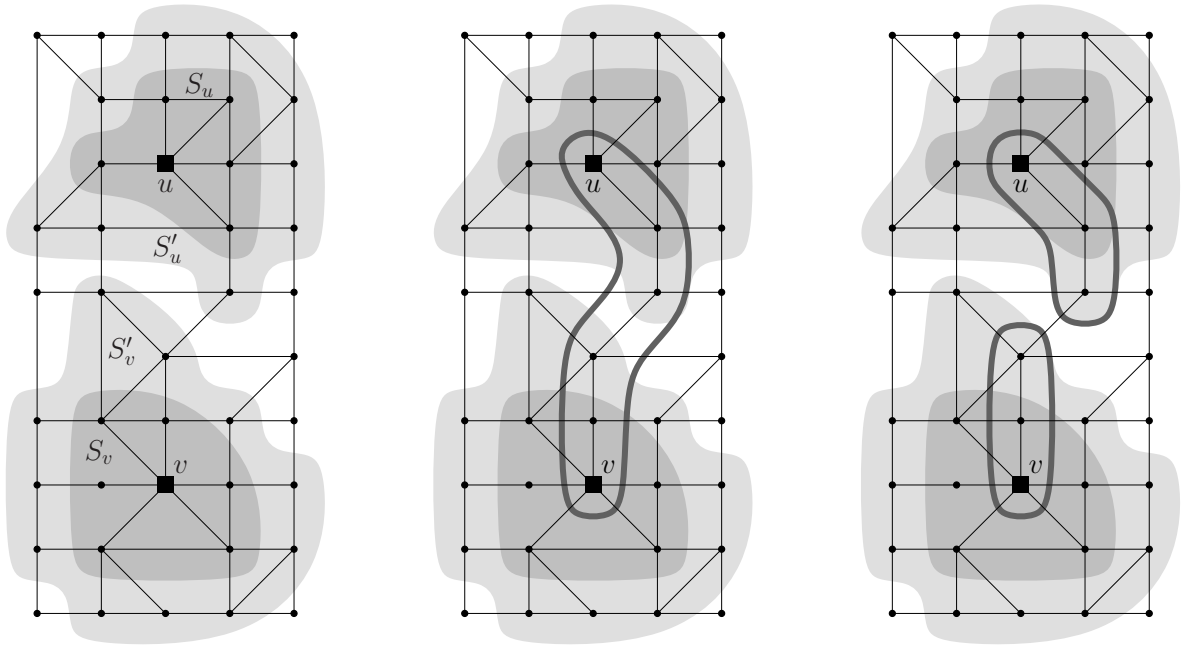
This definition implies the following: if a node v is not contained in S_i, then each contiguous region that contains i and v satisfies the threshold for the class of i. We define S'_i to be the set of nodes in S_i and their neighbors. If $S_i = \emptyset$, we define $S'_i = \{i\}$. In Fig. 7.13(a), these sets are displayed for two nodes, u and v, of the same graph. We observe:

Proposition 2. *If the sets S'_u and S'_v do not intersect, then each contiguous region that contains the nodes u and v can be separated into two feasible regions with centers u and v.*

Proof. Let $V' \subseteq V$ be a contiguous region such that $u, v \in V'$ and $S'_u \cap S'_v = \emptyset$. Since V' is contiguous, it contains a path p from u to v. Let i be the first node of p that is not contained in S_u. Let j be the last node of p that is not contained in S_v. Let p_1 be the subpath of p from u to i. Let p_2 be the subpath of p from j to v. Since $S'_u \cap S'_v = \emptyset$, the two paths do not intersect and each of them contains at least one node. Let w_1 be the total weight of p_1. Since $w_{\min}(u,i)$ is the minimum weight of a path from u to i, we have $w_1 \geq w_{\min}(u,i)$. Since $i \notin S_u$, Eq. (7.43) implies $w_{\min}(u,i) \geq \theta(\gamma(u))$. This implies $w_1 \geq \theta(\gamma(u))$, which means that the region with center u that contains the nodes of p_1 is weight-feasible. Similarly, the region with center v that contains the nodes of p_2 is weight-feasible, too. It remains to assign each node of V' that is not contained in p_1 or p_2 to one of the two regions such that both regions remain contiguous. In order to do this, we can iteratively extend the region with center u to its neighbors in V' that have not yet been assigned to one of both region. If we cannot extend this region any further, we apply the same procedure to the region with center v. We end up with a partition of V' into two contiguous weight-feasible regions. □

Figure 7.13(b) and (c) show an example of this separation. Each region containing u and v has a weight of at least $w_{\min}(u,v) = 6$. Since S'_u and S'_v do not intersect, such a region can be split into two parts, each having a weight of at least three.

Note that for a variant of the aggregation problem, we can achieve this separation without risking to loose optimality. Without the requirement of one node with unchanged class per region and with equal thresholds for different classes, both regions are feasible without changing the classes of the result. So, in the case that the aim for compactness is neglected, the cost of the solution is not affected by this separation. However, for the area aggregation problem defined in Sect. 7.1, the separation may imply additional cost: one part can be infeasible



(a) nodes u and v with sets S_u, S_v, S'_u, and S'_v according to Eq. (7.43)

(b) the smallest contiguous region containing u and v

(c) the region from (b), after separation into two feasible regions

Fig. 7.13: An example of the distance heuristic with $w(v) = 1$ for all $v \in V$ and $\theta(\gamma) = 3$ for all $\gamma \in \Gamma$: each contiguous region containing the two nodes u and v can be split into two feasible regions, since $S'_u \cap S'_v = \emptyset$. Therefore, we set x_{uv} and x_{vu} to 0.

when fixing its class. Also, the sum of costs of both parts can be higher than the cost of their aggregate, as the aggregate may be more compact. Nevertheless, we apply the reduction discussed above, knowing that it is a heuristic that can theoretically affect the quality of our solution. We summarize.

Distance Heuristic:

- A node u is not assigned to a center v if $S'_u \cap S'_v = \emptyset$.

This heuristic is reasonable, since $S'_u \cap S'_v = \emptyset$ implies that each contiguous region containing u and v can be split into two weight-feasible parts. Assuming that for a given center u the number of nodes v with $S'_u \cup S'_v \neq \emptyset$ is constant, the number of remaining variables in the MIP becomes linear. This is an improvement, as the P-R MIP and the Tree MIP originally had a quadratic number of variables.

7.6.2 Center heuristic

A very common approach to speed up the solution of districting problems is to predefine the set of centers (Hess & Samuels (1971), Hojati (1999), Schröder (2001)). We choose this approach to reduce the number of variables in our problem instances. For the aggregation problem, the concept of centers was introduced in our problem definition (Sect. 7.1) and used for the definition of the compactness measures c_{dist} and c_{path} (Sect. 7.2.3). We recapitulate:

- Each region contains exactly one *center* that defines both the class of the region and the reference point for measuring its compactness.

Our idea is to predefine heavy nodes as centers, as the weight of a node is a multiplier for costs that are charged for its class change, and its distance to the center. So, it is likely that solutions with large centers are good according to the objective function. Consequently, we exclude nodes with low weights from the set of potential centers.

When defining a set of nodes as centers, we need to ensure that the problem does not become over-constrained. We can guarantee the feasibility of the problem if we maximally fix one node with unchanged class, for each

region in a feasible start solution. To find a start solution we apply Algorithm 1, that is, region-growing. As the quality of this solution is relatively low, we choose a rather conservative approach, that is, the majority of nodes will not be constrained. We define the center heuristic as follows.

Center Heuristic:

(H1) For each region obtained with Algorithm 1, the largest area with unchanged class is a center and
(H2) each other area of size less than 10% of the threshold is excluded from the set of potential centers.

The first part of the center heuristic (H1) implies that, among other areas, all areas that are sufficiently large for the target scale are predefined as centers. About 7% of all areas in our input dataset at scale 1:50 000 fall into this category when applying the specifications for the target scale 1:250 000. Predefining these nodes as centers does not necessarily need to be regarded as a deficit. Recall our notion of completeness from Sect. 5.1: an area that is large enough for the target scale must not be lost, that is, its class must not change. This is ensured by fixing such nodes as centers. However, other nodes will be additionally constrained with the defined heuristic, which is necessary to obtain an acceptable performance. Applying the second part of the center heuristic (H2) to our dataset, 61% of all areas are excluded from the set of potential centers. Despite this high percentage, these areas make up only 9% of the total area of the dataset.

The center heuristic allows certain variables to be eliminated. However, it also allows a problem instance to be decomposed into smaller instances. Figure 7.14 illustrates an example of such a decomposition.

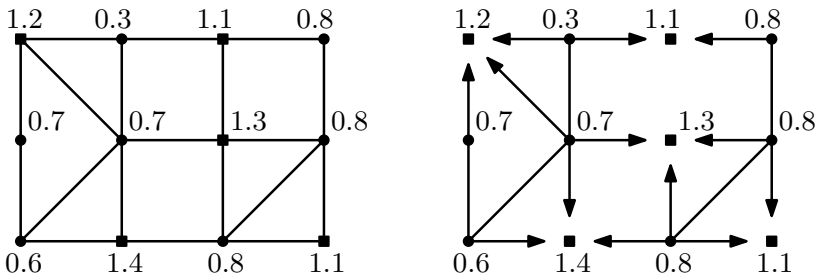


Fig. 7.14: An instance of the aggregation problem (left) that can be decomposed into two independent problem instances (right). Node weights are displayed as numbers, centers with $w(v) \geq \theta(\gamma(v))$ as squares. The threshold is defined by $\theta(\gamma) = 1$ for all $\gamma \in \Gamma$. Arrows illustrate that predefined centers cannot be assigned to other nodes, but unconstrained nodes can be assigned to predefined centers.

We specify the decomposition as follows:

Proposition 3. *Let $V_\theta = \{v \in V \mid w(v) \geq \theta(\gamma(v))\}$. Under the condition that each node in V_θ is fixed as a center and the subgraph of G induced by $V \setminus V_\theta$ is not connected, the* AreaAggregation *instance with any of the compactness measures c_{dist}, c_{path}, and c_{peri} decomposes into several (that is, two or more) independent problem instances. Each such problem instance comprises a connected component of the subgraph of G that is induced by nodes $V \setminus V_\theta$ and the centers surrounding this component.*

Proof. Let $G(V, E)$ be a connected input graph for the aggregation problem. We consider the case that the subgraph of G that is induced by nodes $u \in V$ with $w(u) < \theta(\gamma(u))$ has two connected components. However, the proof can easily be generalized to the case that there are more of such components.

Let V_1 and V_2 be the node sets of the connected components after adding their surrounding nodes. Furthermore, let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be the subgraphs of G that are induced by V_1 and V_2, respectively. For each node u in the intersection $I = V_1 \cap V_2$ we have $w(u) \geq \theta(\gamma(u))$, that is, u is a predefined center. Let V^* be a subset of V that contains exactly one fixed center $v \in I$. Let $V_1^* = V^* \cup V_1$ and $V_2^* = V^* \cup V_2$. We will prove that V^* defines a feasible region for the aggregation problem if and only if V_1^* and V_2^* define feasible regions. Furthermore, we will prove that the cost of the region defined by V^* is equal to the sum of costs of the regions defined by V_1^* and V_2^*. This allows the aggregation problem to be independently solved for G_1 and G_2. The final region for the fixed center v is the union of regions that, in the solutions of the independent sub-instances, have the same center v.

Since $v \in V^*$, $v \in V_1^*$, $v \in V_2^*$, and $w(v) \geq \theta(\gamma(v))$, each of the regions defined by V^*, V_1^*, and V_2^* is weight-feasible and contains a node of unchanged class (that is, the center v). Since $V_1^* \cap V_2^* = \{v\}$ and $V_1^* \cup V_2^* = V^*$, the set V^* defines a contiguous region if V_1^* and V_2^* define contiguous regions. Since there is no edge $\{v_1, v_2\} \in E$ with $v_1 \in V_1^* \setminus \{v\}$ and $v_2 \in V_2^* \setminus \{v\}$, the set V^* does not define a contiguous region if one of the sets V_1^* or V_2^* does not define a contiguous region. Therefore, V^* defines a feasible region if and only if V_1^* and V_2^* define feasible regions.

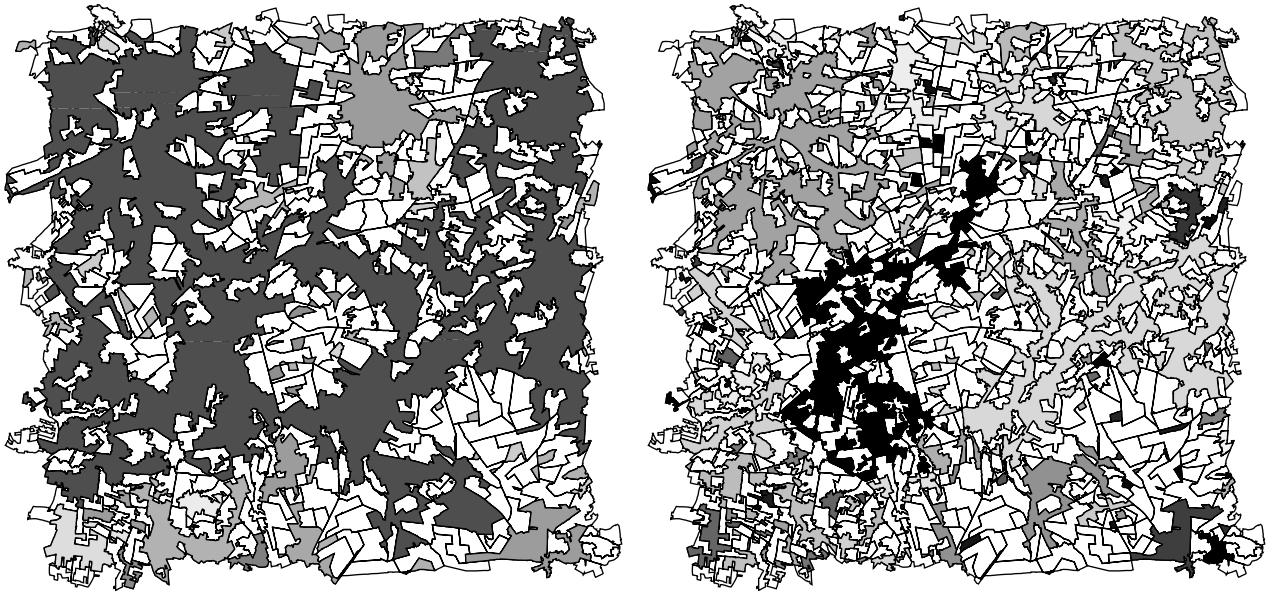
We now show for different measures that the cost of the region defined by V^* is equal to the sum of costs of the regions defined by V_1^* and V_2^*.

Both the cost for the class change f_{class} and the compactness measure c_{dist} are defined as the sum of terms that are individually charged for each node; these terms only depend on properties of the node and the center to which the node is assigned. The same center v is defined for the regions V_1^*, V_2^*, and V^*. Furthermore, each other node of V^* is contained either in V_1^* or V_2^*, but not in both. Therefore, according to f_{class} and c_{dist}, the regions defined by V_1^* and V_2^* together imply the same cost as the region defined by V^*.

The measure c_{path} additionally depends on the length of the shortest path that is contained in a region and connects a node with the center. For a region V^* that contains a fixed center, such a path will always be contained in one of the regions V_1^* or V_2^*. Again, this is because there is no edge $\{v_1, v_2\} \in E$ with $v_1 \in V_1^* \setminus \{v\}$ and $v_2 \in V_2^* \setminus \{v\}$. Therefore, also the measure c_{path} for V^* is equal to the sum of c_{path} for V_1^* and V_2^*.

Finally, we discuss the measure c_{peri}. In Sect. 7.4.4 we argued that a cost for the perimeter of a region can be alternatively expressed as a benefit (i.e., a negative cost) that is given for edges contained in a region. Again, we choose this perspective. Each edge contained in the region defined by V^* is either contained in the region defined by V_a^* or in the region defined by V_b^*, but not in both. Consequently, the benefit given for V^* is equal to the sum of benefits for V_a^* and V_b^*. In conclusion, the cost of the region V^* is equal to the sum of costs of the regions V_a^* and V_b^*. □

The decomposition of a problem instance into smaller, independent problem instances can be used for the following two tasks. (1) If we wish to process large datasets, we can solve the independent problem instances either in succession or in parallel. Both approaches reduce the processing time. (2) If we wish to propagate an update that was inserted into the source dataset, we only need to resolve those problem instances that contain updated areas. This also reduces the processing time since we do not need to process the whole map again.



(a) applying thresholds for target scale 1:250 000 (b) applying half thresholds corresponding to intermediate scale

Fig. 7.15: Independent problem instances resulting from center heuristic (different gray shades). The largest problem instance is displayed with the darkest shade. White areas have sufficient size.

7.6.3 Introducing intermediate scales

Figure 7.15 shows the independent problem instances that result from the center heuristic for our dataset at scale 1:50 000 (recall Fig. 4.5). After applying the collapse procedure the dataset contains 5537 areas. In Fig. 7.15(a) we applied the thresholds according to the existing specifications for the target scale 1:250 000. Though the original problem instance decomposes into 145 smaller instances, one huge connected component remains that contains 4226 areas; this is far too large to be processed at once. We therefore suggest the definition of intermediate scales, that is, to satisfy smaller thresholds first. In Fig. 7.15(b) we applied thresholds of half value, yielding much smaller independent problem instances. If we know that we can solve instances with a number of K nodes, we can further decrease the thresholds until no instance is larger than this. Hence, we can translate the capability to solve problem instances of limited size into the capability to handle limited differences in scale.

Certainly, the number of scale steps that are needed to reach the target scale is different for different parts of the dataset. Our dataset contains large forest areas that have a separating effect, however, in other parts there are many small areas. We therefore suggest using Algorithm 5 to locally introduce intermediate thresholds, whenever they are needed to restrict the size of the resulting instances. Figure 7.16 illustrates the algorithm using a one-dimensional example.

Throughout the algorithm, we maintain a set of problem instances, that is, connected components (defined in line 1). We let them grow by iteratively adding the smallest area that falls below the required threshold (lines 2–18). If we obtain a connected component containing K areas, we define and solve a problem instance as in Fig. 7.14, that is, we allow the contained areas to be aggregated with each other or to assign them to one of the adjacent areas (line 9). We need to define the intermediate thresholds θ' for this step such that all areas in the neighborhood of the component (that is the centers) have sufficient weight; thus we define $\theta'(\gamma) = \min\{\theta(\gamma), W_{\min}\}$, with $W_{\min}$ being the smallest area in the neighborhood. To 'solve' the problem also means that we update the map by replacing areas with the composite regions we found. In lines 14–15 we handle the case that different components may unite when adding a single area, thus we might end up with instances of more than K areas. To avoid this, we solve the largest involved component. Finally, we solve the remaining instances in succession applying the thresholds for the target scale (line 19).

We now argue that Algorithm 5 generalizes both: the original iterative approach as well as the optimization approach with the center heuristic. Let's first discuss the special setting of $K := 1$. In this case the algorithm iteratively selects the smallest area and assigns it to one of its neighbors, which is identical to the approach of Algorithm 1. If we set $K := |V|$, we solve the problem in a single step without intermediate thresholds. Our assumption is that we obtain better results if we choose K to be as high as possible, thus we allow as many merges to be planned ahead as possible.

We can apply Algorithm 5 to the complete dataset in order to create datasets at smaller scales. For example, we can choose this approach for the set-up of a multiple representation database (MRDB). If the source data set is updated, we also need to update the derived small-scale datasets. Recall from Sect. 2.8 that we should avoid a complete re-generalization of the most detailed layer of an MRDB. Instead, we should apply incremental generalization methods (Kilpeläinen & Sarjakoski, 1995). Our problem decomposition based on the center heuristic indeed allows such an approach: we only need to apply Algorithm 5 to those problem instances again that were affected by the update. However, if a change affects a large problem instance, for example, the component of 4226 areas in Fig. 7.15(a), we would still need to apply Algorithm 5 to the whole component.

As processing the affected problem instances can take a lot of time, it is reasonable to define and to permanently maintain datasets of intermediate scales in the MRDB. For example, we can define an intermediate scale with half thresholds, yielding the problem instances in Fig. 7.15(b). If wish to create a new small-scale dataset, we create the intermediate scale first and then derive the small-scale map from this intermediate representation. We can update the intermediate scale by considering only a relatively small subset of areas, since the potentially affected problem instances are relatively small. In the same way we can propagate the update from the intermediate scale to the small scale. In any case we can apply Algorithm 5 to ensure that the occurring problem instances are small enough to be solved.

Algorithm 5 Iterative aggregation of areas in big scale steps

1: $\Pi \leftarrow$ a set of connected components, initially empty
2: $a \leftarrow$ smallest area below threshold for the target scale
3: **while** a is not Null **do**
4: $\Pi' \leftarrow$ the set of connected components in Π containing a neighbor of a
5: **if** total number of areas contained in $\Pi' < K$ **then**
6: Remove all components in Π' from Π.
7: Create a new component p comprising all areas in Π' as well as a.
8: **if** p contains K areas **then**
9: Solve p.
10: **else**
11: Insert p to Π.
12: **end if**
13: **else**
14: Solve the instance that corresponds to the component in Π' having most areas.
15: Remove this component from Π.
16: **end if**
17: $a \leftarrow$ smallest area below threshold for the target scale, not contained in Π
18: **end while**
19: Solve all problem instances corresponding to the remaining components in Π.

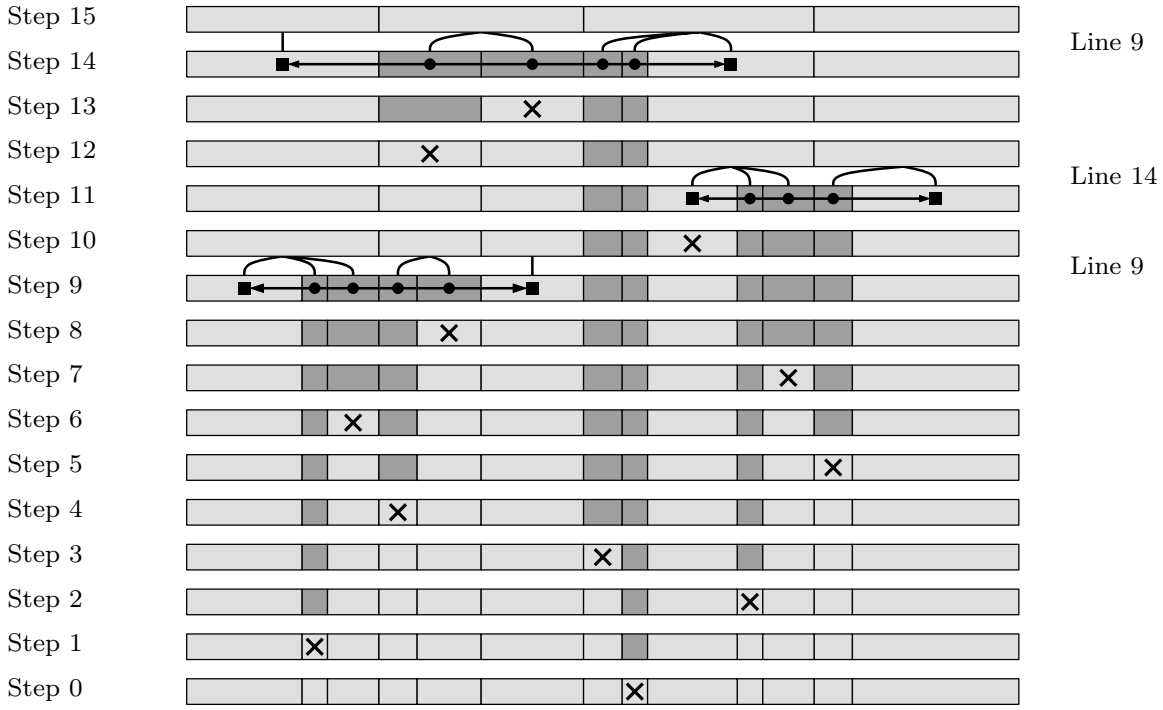


Fig. 7.16: Several steps of Algorithm 5 with $K = 4$ (from bottom to top). The optimization method was applied three times; both cases that are defined in lines 9 and 14 of the algorithm occurred. Areas that were added to components in Π are displayed in dark gray. Among the other areas (light gray), the smallest area is selected in each iteration (marked with X). The connected components in Π are updated by adding this area, if this does not imply a connected component of more than four areas. If a connected component of exactly four areas is created, the corresponding problem instance is solved. In the example, this happens in steps 9 and 14. If adding an area would imply a connected component with more than four areas, the problem instance corresponding to the largest adjacent component is solved. In the example, this happens in step 11: adding the smallest area in step 10 to Π would imply a connected component of six areas.

7.7 An alternative heuristic approach by simulated annealing

This section introduces an alternative method for AreaAggregation, which instantiates the general simulated annealing approach defined in Algorithm 3 on page 41. In order to find the initial feasible solution we simply apply Algorithm 1, that is, region-growing. The 'least important area' is defined according to size; the 'most compatible neighbor' is defined to be the neighbor that results in minimal costs when merging the selected area. In the remainder of this section we define a neighborhood function and detail our additional problem-specific design decisions. Results of this simulated annealing approach will be presented in Chapter 8, together with results that were obtained by mixed-integer programming.

7.7.1 A neighborhood based on swaps of nodes

The neighborhood structure is the most important issue in the development of a simulated annealing method. In particular, we need to ensure that the solution space is connected (see requirement (S2) on page 42). The neighborhood structure defined in this section is similar to the one used by Tavares-Pereira *et al.* (2007) in a hybrid evolutionary optimization approach for districting problems. Basically we define the neighborhood $N(s)$ of a feasible solution $s \in S$ as the set of solutions that can be obtained by removing a single node from a region and assigning this node to an adjacent region. Additionally, we allow that the removed node defines a new region. We restrict the set of feasible classifications to those that are optimal for a partition. This means that each change of the partition triggers an update of the centers (and thereby the classes) of the involved regions, such that the cost for the resulting partition is minimized. We refer to this procedure as a *swap*, which we now define formally.

We define a swap for each ordered node pair ab where a and b belong to two different regions of the partition and $\{a, b\} \in E$. We denote the region containing a by V^a and the region containing b by V^b. Let $W^a = V^a \setminus \{a\}$ and $W^b = V^b \cup \{a\}$. Furthermore, let $W_1^a, W_2^a, \ldots, W_q^a$ be the node sets of the connected components of the subgraph of G induced by W^a. Applying the swap ab implies that the region defined by V^a is replaced by regions $W_1^a, W_2^a, \ldots, W_q^a$. The region defined by V^b is replaced by W^b. For examples, see Figures 7.17(a), (b), and (d). In Fig. 7.17(b) the set W^a is empty. The region $V^a = \{a\}$ completely merges into the region V^b.

Additionally, we define a swap for each single node a in a contiguous region. Again we denote the node set of this region by V^a and we apply the definitions of W^a and $W_1^a, W_2^a, \ldots, W_q^a$ as before. Applying the swap a implies a new region with node set $\{a\}$. The region with node set V^a is replaced by new regions with node sets $W_1^a, W_2^a, \ldots, W_q^a$. Figure 7.17(c) shows a swap that is defined by a single node.

We call a swap *simple* if the subgraph of G induced by W^a does not have more than one connected component. This definition applies to both a swap defined by a pair ab and a swap defined by a single node a. The swaps in Figures 7.17(a)–(c) are simple. In contrast, the swap in Fig. 7.17(d) is non-simple, since it implies that the region of four nodes is split into two contiguous regions.

If we apply a swap, a solution can become infeasible. This is because a region can become too small if we remove a node. However, it is critical to restrict the set of swaps to those that keep regions weight-feasible. Consider an initial solution containing only regions that exactly satisfy their weight thresholds. Removing any node from its region creates an infeasible solution. The initial solution represents an isolated point in the solution space. Therefore, the requirement for connectivity of the solution space would be violated. In order to ensure the connectivity of all solutions via the defined neighborhood, we relax the constraint for weight-feasibility during the simulation, which is illustrated in Fig. 7.18.

Our relaxation is based on a penalty $p' : 2^V \times \Gamma \to \mathbb{R}_0^+$, which we define as follows:

$$p'(V', \gamma') = \begin{cases} \theta(\gamma') - \sum_{v \in V'} w(v)\,, & \text{if } \sum_{v \in V'} w(v) < \theta(\gamma'), \\ 0\,, & \text{else}\,. \end{cases} \tag{7.44}$$

We define the cost for the relaxed problem by:

$$f' = r \cdot \sum_{i=1}^{k} p'(V_i, \gamma_i') + (r-1) \cdot f\,, \tag{7.45}$$

with $f = s \cdot f_{\text{class}} + (1 - s) \cdot f_{\text{comp}}$ being the cost for the original problem in Eq. (5.4) and $r \in [0, 1]$ being the weight of the penalty for regions that are too small. We refer to the problem of minimizing this cost function as *relaxed aggregation problem*.

A drawback of the relaxation is that the simulated annealing algorithm may terminate with regions that do not satisfy the originally hard size constraint. We need to define an algorithm that repairs infeasible solutions. Again we apply the iterative algorithm for this. We select the smallest infeasible region and merge it with the best neighbor until the result is feasible. However, we apply this procedure only if the final solution is infeasible.

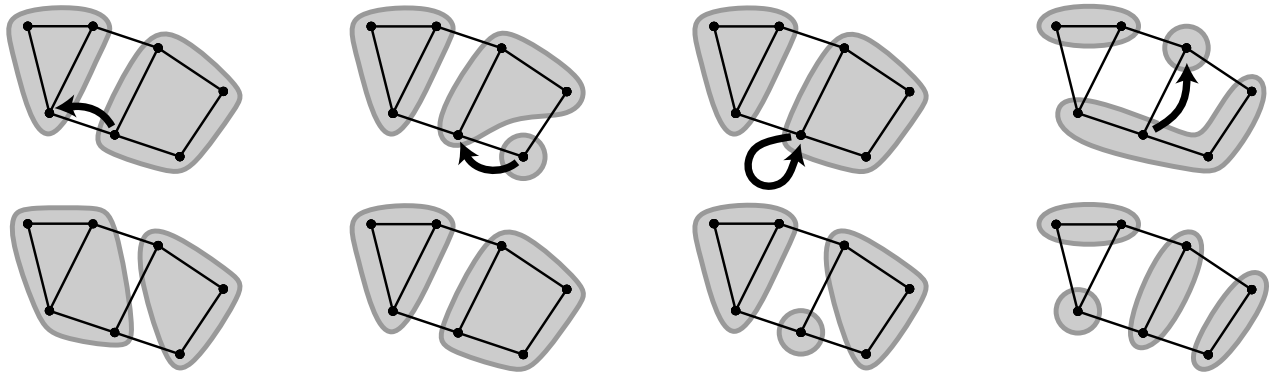


Fig. 7.17: The swap neighborhood. Solutions before (top) and after application of a swap (bottom).

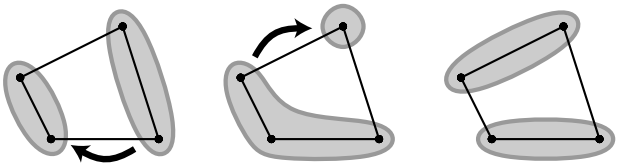


Fig. 7.18: If we define the size threshold $\theta = 2$ and a unit weight to each node, there is no feasible path from the left solution to the right solution. Relaxing the size constraint, the graph defined by the neighborhood function becomes connected. The right solution can be reached via the solution in the center.

7.7.2 Asymptotic convergence

We now investigate the theoretical aspects of our simulated annealing method. In particular, we show that with a certain setting it converges to optimality.

Theorem 2. *The simulated annealing algorithm converges with probability* 1.0 *to the set of globally optimal solutions of the relaxed aggregation problem if the annealing is performed slowly enough and if we only allow simple swaps.*

Proof. We now prove that the requirements (S1)–(S3) on page 42 are satisfied. According to Michiels *et al.* (2007), this implies that the simulated annealing algorithm convergences to the set of globally optimal solutions.

Requirement (S1) states that the neighborhood graph must be finite. This is the case since the set of partitions is finite.

Requirement (S2) states that the neighborhood graph must be strongly connected. This is ensured since the trivial partition $P_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ can be reached from any other partition P_1 by applying maximally

$n = |V|$ swaps that are defined by a single node (see Fig. 7.17(c)). Similarly, any other partition P_2 can be reached from P_0 by applying a sequence of swaps, each of them merging a region of a single node into an adjacent region (see Fig. 7.17(b)). Consequently there is a path from P_1 to P_2, and furthermore the diameter of the neighborhood graph is at most $2|V|$.

Requirement (S3) states that the neighborhood graph must be symmetric. This is the case if we only allow simple swaps, that is, if we do not allow a region to be split as shown in Fig. 7.17(d). For each simple swap that does not change the number of regions (see Fig. 7.17(a)), there is an inverse swap of the same type. For each simple swap that reduces the number of regions by one (see Fig. 7.17(b)) there is an inverse swap that increases the number of regions by one (see Fig. 7.17(c)) and vice versa. □

Note that the neighborhood graph is not symmetric if we allow non-simple swaps, that is, if we allow a region to be split into multiple regions, as shown in Fig. 7.17(d). There is no inverse operation for this example.

Our proof shows that the algorithm converges to the global optimum of the *relaxed* area aggregation problem. This can be different from the optimum of the original problem. For this reason, let's discuss a relaxation of the area aggregation problem that is different from the relaxation defined in Equations (7.44) and (7.45). Again we define a penalty for regions that are too small:

$$p''(V', \gamma') = \begin{cases} f_0 + 1\,, & \text{if } \sum_{v \in V'} w(v) < \theta(\gamma') \\ 0\,, & \text{else}\,, \end{cases} \tag{7.46}$$

with f_0 being the total cost of the initial feasible solution according to the original cost function f. As an alternative to f' we could apply the cost function f'' in the relaxed area aggregation problem, which we define as follows:

$$f'' = \sum_{i=1}^{k} p''(V_i, \gamma_i') + f. \tag{7.47}$$

Applying the cost function f'' implies that a solution with regions that are too small is always more expensive than the initial feasible solution. Consequently, the best feasible solution will always be cheaper than the best infeasible solution. The simulated annealing method will converge to an optimal feasible solution.

7.7.3 Practical considerations

In practice we need to define a finite and preferably small number of iterations. Occasionally we need to accept regions that are too small, in order to escape local optima. However, with the large constant penalty p'', this would require an extremely high initial temperature and a slow cooling process. Therefore we apply the cost function f' with the penalty p' in Eq. (7.44), which better reflects the degree of how much the constraint for weight-feasibility is violated. For similar practical reasons, we allow non-simple swaps. The neighborhood becomes asymmetric with this definition, but we obtain more possibilities of variation without any additional effort.

An important requirement for the success of local search is that candidates can efficiently be generated and tested. To efficiently generate solutions we define a vector of arcs $\Omega = (\omega_0, \omega_1, \ldots, \omega_p)$. We define the vector Ω to contain all arcs of the set $A = \{uv \mid \{u, v\} \in E\}$. These arcs correspond to swaps defined by two nodes. Furthermore, we define Ω to contain a self arc uu for each node $u \in V$. These arcs correspond to swaps defined by a single node. To generate candidate solutions at random, we define $\omega : [0, 1) \to \Omega$ as the mapping of random numbers to the swaps. This is specified by:

$$\omega(x) = \omega_{i(x)} \quad \text{with} \quad i(x) = \lfloor x \cdot (p + 1) \rfloor\,. \tag{7.48}$$

In each iteration of the simulated annealing algorithm we randomly generate a number x uniformly distributed in $[0, 1)$. This number yields the arc $uv = \omega(x)$. If the node u and v belong to different regions then the arc uv indeed defines a swap. This is also true if $u = v$ and u is not the only node in its region. In both cases, the swap is applied: the new partition is generated and tested. Otherwise, we select the next arc by generating a random number.

Constructing and testing the new solution for a selected arc $ab \in \Omega$ includes several steps. Let $V^a \in P$ and $V^b \in P$ be the node sets of the original regions that contain a and b, respectively. We first need to find the connected components $W_1^a, W_2^a, \ldots, W_q^a$ of the graph induced by $W^a = V^a \setminus \{a\}$ to consider the case in Fig. 7.17(d), that is, the case that one region is split into multiple regions. This is achieved with a breadth first expansion from an arbitrary node, which generally allows the connected components to be found in a planar graph in $\mathcal{O}(|V|)$ time. The connected components $W_1^a, W_2^a, \ldots, W_q^a$ and the set $W^b = V^b \cup \{a\}$ define new regions. For each new region we need to find the optimal center, which defines the cost for the region. This can be done in $\mathcal{O}(|V|^2)$ time. The cost of other regions does not need to be computed, as it remains unchanged. The difference in cost between the new and the old solution becomes:

$$\sum_{i=1}^{q} \varphi(W_i^a) + \varphi(W^b) - \varphi(V^a) - \varphi(V^b)\,, \tag{7.49}$$

with $\varphi : 2^V \to \mathbb{R}_0^+$ being the cost f' in Eq. (7.45) that is charged for a single region. Note that $\varphi(V^a)$ and $\varphi(V^b)$ in Eq. (7.49) have been computed before. To avoid these costs being computed again, we store the cost for each region.

The proposed method often selects an arc uv from the set Ω that does not correspond to a swap. For instance, in the current partition, both nodes u and v belong to the same region. We also tested a version of the proposed method where Ω only contains those arcs that indeed correspond to swaps. However, this approach requires the vector Ω to be updated after each iteration. The approach proposed above turned out to be much more efficient: we need to reject more candidate arcs, but we do not need to update the vector Ω.

Note that the uniform distribution of arcs Ω does not exactly imply a uniform distribution of solutions in the defined neighborhood. This is because multiple arcs can correspond to the same solution. For instance, the new solution in Fig. 7.17(b) can be obtained from the original solution by selecting two different arcs. We could insist that Ω must not contain two arcs that correspond to the same solution. This however would again imply that we need to update the vector Ω.

7.7.4 Annealing schedule

A problem of simulated annealing is that several tuning parameters need to be defined. For our application we define the stop criterion in Algorithm 3 by the number of iterations κ, which we increase linearly according to the number of nodes of the problem instance. This definition respects the requirement to perform longer runs of simulated annealing to obtain high-quality results, for larger problem instances. In contrast, the initial temperature T_0 and the final temperature T_E are defined to be fixed for all problem instances. With this we can find the annealing parameter α for Eq. (3.20) by:

$$\alpha = \sqrt[\kappa]{\frac{T_E}{T_0}}\,. \tag{7.50}$$

As κ is defined depending on the number of nodes, also α depends on the input size. This means that the temperature is cooled down more slowly for larger problem instances. Figure 7.19 illustrates this approach.

Finally we need to set the parameter r, that is, the weight of the penalty for regions that are too small. In contrast to the weight factor s in Eq. (5.4), this does not reflect the preferences of cartographers with respect to different solutions. Instead it needs to be defined such that the complete method (including the repair procedure) yields feasible results with low cost f. In other words, r is a tuning parameter that is not inherent to the problem. If we set r too high, we run the risk of getting trapped in local optima. Contrarily, if we set r too low, we often obtain infeasible results, which need to be repaired with the simple iterative aggregation algorithm. As a consequence we may end up with costly solutions as well.

Normally, the tuning parameters for simulated annealing are found by experiments. The parameters in Table 7.1 were found using this approach. These were used to produce the results in Chapter 8, if not explicitly stated differently.

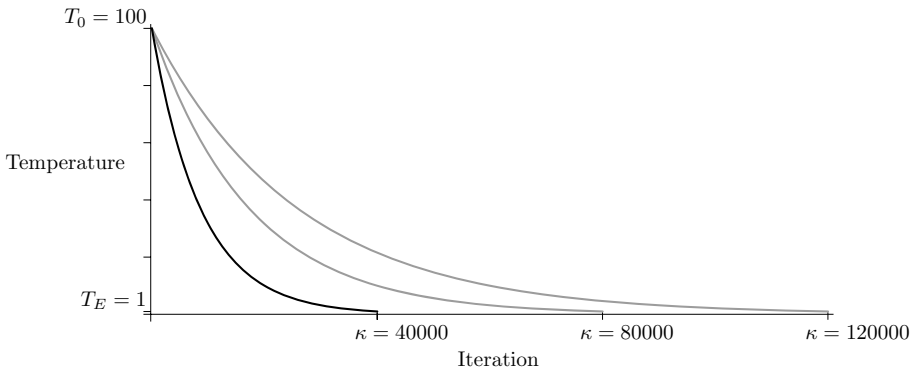


Fig. 7.19: Principle of the definition of an annealing schedules. Three schedules are shown, which all have the same start and end temperatures but different numbers of iterations.

name of parameter	symbol	value
initial temperature	T_0	10^9
final temperature	T_E	10^4
weight of penalties for areas that are too small	r	0.9999
number of iterations	κ	$1000 \cdot \|V\|$

Table 7.1: Experimentally found parameters for simulated annealing.

7.8 Possible combinations of the proposed methods

We conclude this chapter by discussing possible combinations of the proposed aggregation methods.

We can use the distance heuristic to eliminate variables in the P-R MIP or the Tree MIP. The center heuristic can be used to eliminate variables in all the MIP formulations that were presented. Moreover, it implies a decomposition of a large problem instance into smaller, independent instances – there is no assumption about the optimization method that is used to solve these instances. For example, we can also combine this method with the simulated annealing approach. The same is true for the method with intermediate scale steps. We can solve the occurring problem instances by mixed-integer programming or alternatively by simulated annealing.

The center heuristic and the idea of defining intermediate scales are especially interesting. Together, both approaches allow very large datasets to be partitioned into manageable subsets. We can generalize these subsets in parallel and, in the update scenario, we only need to handle those parts that include a changed object.

Probably, the best decision is to combine different heuristics. For example, if we apply the distance heuristic to eliminate variables in the MIPs, we can solve larger problem instances in the same time. This means that we can consider more areas in each step of Algorithm 5, that is, we can choose a higher value for the parameter K. The next chapter presents experimental results with our methods, which includes tests with different combinations of the proposed heuristics.

Chapter 8

Experimental Results

This chapter presents experimental results with the developed generalization workflow of collapse, aggregation, and line simplification. This workflow was applied to the dataset "Buchholz in der Nordheide" in the ATKIS DLM 50, which was introduced in Fig. 4.5 on page 50. The method was employed to satisfy specifications of the ATKIS DLM 250, that is, the target scale of generalization is 1:250 000.

With respect to the target scale, the offset for the collapse procedure was defined to be 25 m: only polygon parts of width greater than 50 m survive this collapse, which corresponds to 0.2 mm in the target scale, that is, the resolution of a printed map. Originally, the dataset contained 5461 areas. After the collapse procedure the number of areas increased to 5537 because some polygons were split into several parts. The size threshold θ for the aggregation problem was taken from the database specifications and the class distance d was applied according to the table in Fig. 5.4(b) on page 58. The tolerance ε for the line simplification problem was set to 50 m, which again corresponds to the map resolution.

When discussing the tests we mainly focus on the aggregation methods, which constitute the major contribution of this thesis. We distinguish the following problems:

- What is the performance of the presented aggregation methods, that is, how much time do the methods need to solve the problem and do the heuristics offer near-optimal solutions? We answer these questions in Sections 8.1 and 8.2.
- How well does the defined objective function reflect the aims of map generalization? We deal with this issue in Sect. 8.3.
- How can we assess the quality of the result? We discuss this problem in Sect. 8.4.

We summarize the most important results in Sect. 8.5.

The aggregation methods were first tested without the objective of compact shapes, that is, the sole objective was to minimize the cost f_{class} that expresses class changes. In Sect. 8.1 we discuss results of different MIP formulations with this setting. This simplification of the problem is due to two reasons: firstly, the developed MIP formulations do not allow the same compactness measures to be applied – we should only compare their performance when applying the same objective. Secondly, when applying the presented methods, we first need to find an appropriate setting of weights for different objectives. Presumably, there is no perfect setting that serves all purposes, since the definition of weights depends on personal preferences and the purpose of the generalized dataset. The objective for minimum class change can be considered a basic setting. Starting with this, we can increase the weight of the cost for non-compact regions f_{comp} until we visually perceive that the shapes are compact enough. We also discuss results of our heuristics in a setting where we did consider compactness, see Sections 8.2 and 8.3.

8.1 Performance tests of different MIPs

The P-R MIP, the Tree MIP, and the Flow MIP were tested for problem instances of different sizes that were taken from the ATKIS DLM 50. For these tests we used the Interactive Optimizer of the software ILOG CPLEX 9.100 on a Linux server with 4 GB RAM and a 2.2 GHz AMD-CPU. The optimizer reads a MIP from a text file (in the so-called CPLEX LP format), solves the MIP, and writes another text file with the result (in CPLEX MST format). A C++ program was implemented to create the input files for CPLEX. It requires a file with input polygons, their classes, sizes, and the values of the threshold function θ (in ESRI SHP format). The class distance matrix, that is, the values of d, need to be provided in a text file. The user needs to specify the weights of different objectives and select the method to apply, together with the heuristics that are to be used. The program analyzes the dataset, for example, it derives the adjacency graph G from the given geometries. It also reduces the set of variables according to the selected heuristics before writing the MIP to the file. Appendix A shows a small problem instance with the P-R MIP, which was output by the program. A second program is used in a post-process to generate a shape file from the solution found by the optimizer.

The time to analyze the dataset and to generate the MIP formulations is negligible compared to the time that is needed to solve the MIPs; therefore, we only assess the time that CPLEX needed for the MIP solutions. Basically, the solver applied the branch-and-cut method from Sect. 3.6.1: if the solver finds a new feasible solution that is better than the current incumbent solution, it still needs to prove the optimality of this solution by raising the lower bound for the objective function.

We also implemented Algorithm 1 (region-growing) in order to compare its results with those of the newly developed optimization methods. For all experiments with this method, the most compatible neighbor was defined according to the cost function f. This means that, in each step, the smallest area is merged with a neighbor such that the cost after this merge is minimal. In the case that there are several neighbors that imply a minimal cost, we select the largest among them. Obviously, this approach generally does not yield the globally optimal solution. If we compare the performance of different methods in terms of running time or cost values we always apply the same cost function f. As we neglect compactness in this section, the function f equals f_{class}. In this case, the cost only depends on the classes of areas and the semantic distances d.

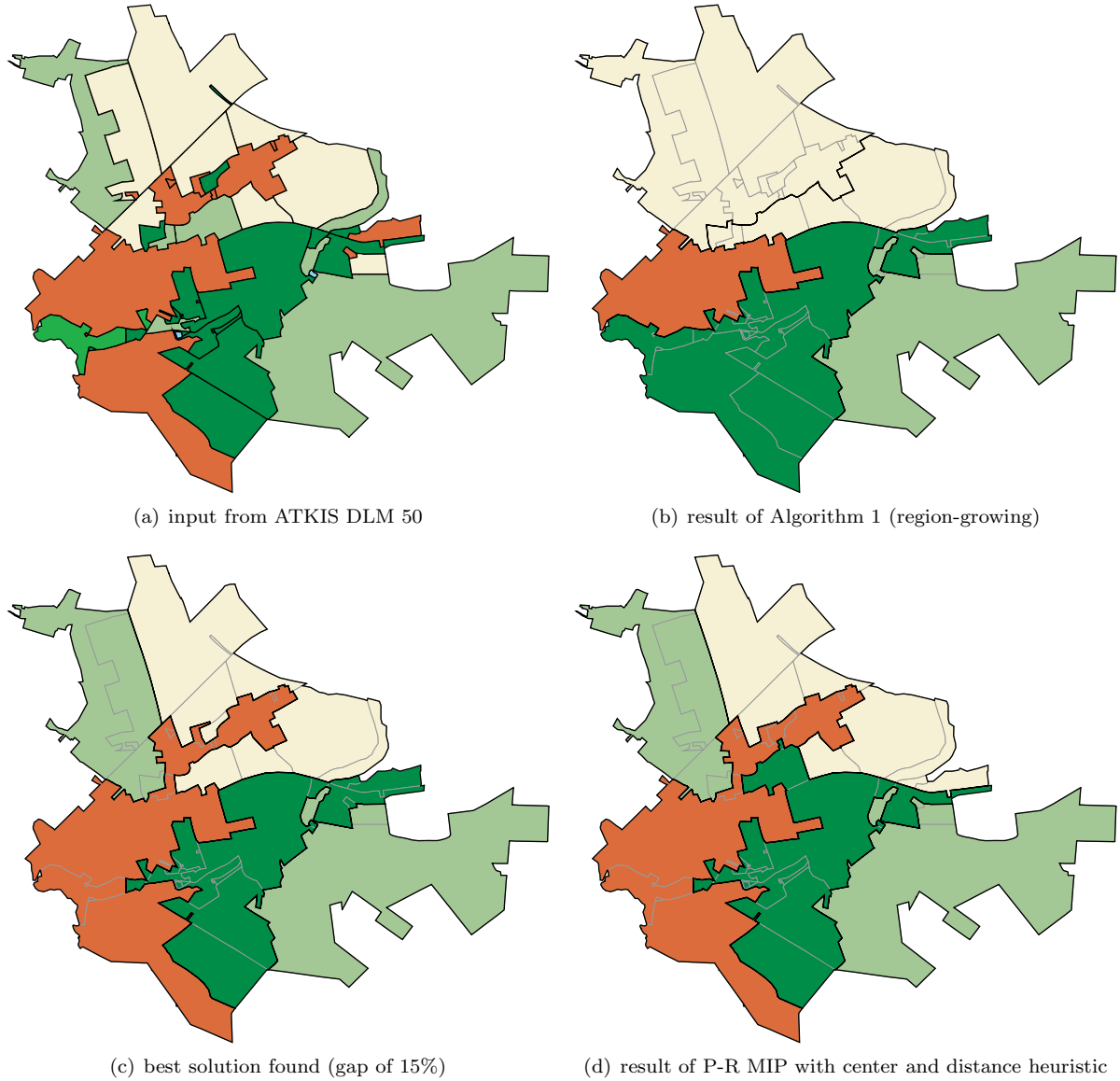
Using the Tree MIP from Sect. 7.4.2 the solver could only handle very small problem instances. The largest instance that was solved with proof of optimality contained 15 nodes; these were aggregated into two regions of different classes. The solver needed 654 seconds for this processing. Usually, finding good solutions did not require so much time, for example, the optimal solution to the instance with 15 nodes came out after 5 seconds. However, the remaining time (649 s) was needed to certify optimality, that is, to raise the lower bound until it was equal to the cost of the integer solution. Apparently, the bounds that are found by solving the LP-relaxation of the Tree MIP are not tight enough to keep the branching tree small. The solver performed better with the P-R MIP from Sect. 7.4.1 and the Flow MIP from Sect. 7.5. We focus on tests with these MIP formulations.

Table 8.1 summarizes the results for problem instances with different sizes that were solved with different methods. Figure 8.1(a) displays the input dataset of 50 areas that was processed in these tests. Figures 8.1(b)–(d) show results that were obtained with different methods. Figure 8.1(b) shows the result of Algorithm 1. Applying Equation (5.2) we obtain an average class distance $\bar{d} = 15.30$ for this result. Results for different instances that were obtained with this method are summarized in the first column of Table 8.1. All computations took less than a second on a standard desktop PC. The best solution found is shown in Fig 8.1(c), which has an average class distance $\bar{d} = 5.18$. This was obtained by applying the exact Flow MIP. However, even after 20 hours of processing, there remained a gap of 15% between the cost of the solution and the certified lower bound. We did not continue to prove the optimality of this solution, but assume that the solution is optimal. The running time for solving this instance is marked with an asterisk in Table 8.1. All other instances were solved with proof of optimality. Figure 8.2 illustrates the resulting class distances for the solutions in Fig. 8.1.

Though the Flow MIP allows larger instances to be solved than the Tree MIP, the required time is still very high and increases rapidly. Certainly, a method that requires 20 hours to solve a small instance is not appropriate for cartographic production. However the obtained results allow the results of other methods to be assessed. For example, we can say that the region-growing method fails to find near-optimal solutions: applying the Flow MIP, the average class distance $\bar{d}$ was reduced by two-thirds. In the example, this difference is due to some settlement areas in the input whose size is not sufficient for the target scale. These areas are lost with the region-growing approach. In contrast, the optimization method changes small forest and grassland areas into settlement, yielding one large, contiguous settlement for the target scale: the settlement areas are saved.

	Alg. 1	Flow MIP				P-R MIP							
		pure		center heuristic		pure		dist. heuristic		center heuristic		center + dist. heur.	
nodes	$\bar{d}$	time	$\bar{d}$	time	$\bar{d}$	time	$\bar{d}$	time	$\bar{d}$	time	$\bar{d}$	time	$\bar{d}$
30	29.30	90.2	9.20	10.3	9.20	4.6	9.20	8.5	9.20	0.01	12.81	0.01	12.81
40	22.31	12.7**h**	6.97	620.0	6.97	26.6	7.34	32.7	7.34	0.03	7.59	0.03	7.59
50	15.30	*20.0**h**	5.18	2.7**h**	5.18	62.2	5.18	95.2	5.18	0.45	5.64	0.24	5.64
60	15.49					570	5.85	416.5	5.85	0.79	6.66	0.76	6.66
100	12.48									21.6	5.93	21.5	5.93
200	9.85									100.4	4.68	202.9	4.68
300	6.92									714.7	4.50	444.8	4.50
400	7.04									1366.9	4.64	1032.2	4.64

Table 8.1: Experimental results with our MIPs, neglecting compactness. Computation times are in seconds of CPU time unless marked with **h**, which stands for hours. All instances were solved to optimality except *. The average class distance $\bar{d}$ is equal to the cost for class change per area as defined in Eq. (5.2). Class distances are defined according to Fig. 5.4(b).



(a) input from ATKIS DLM 50

(b) result of Algorithm 1 (region-growing)

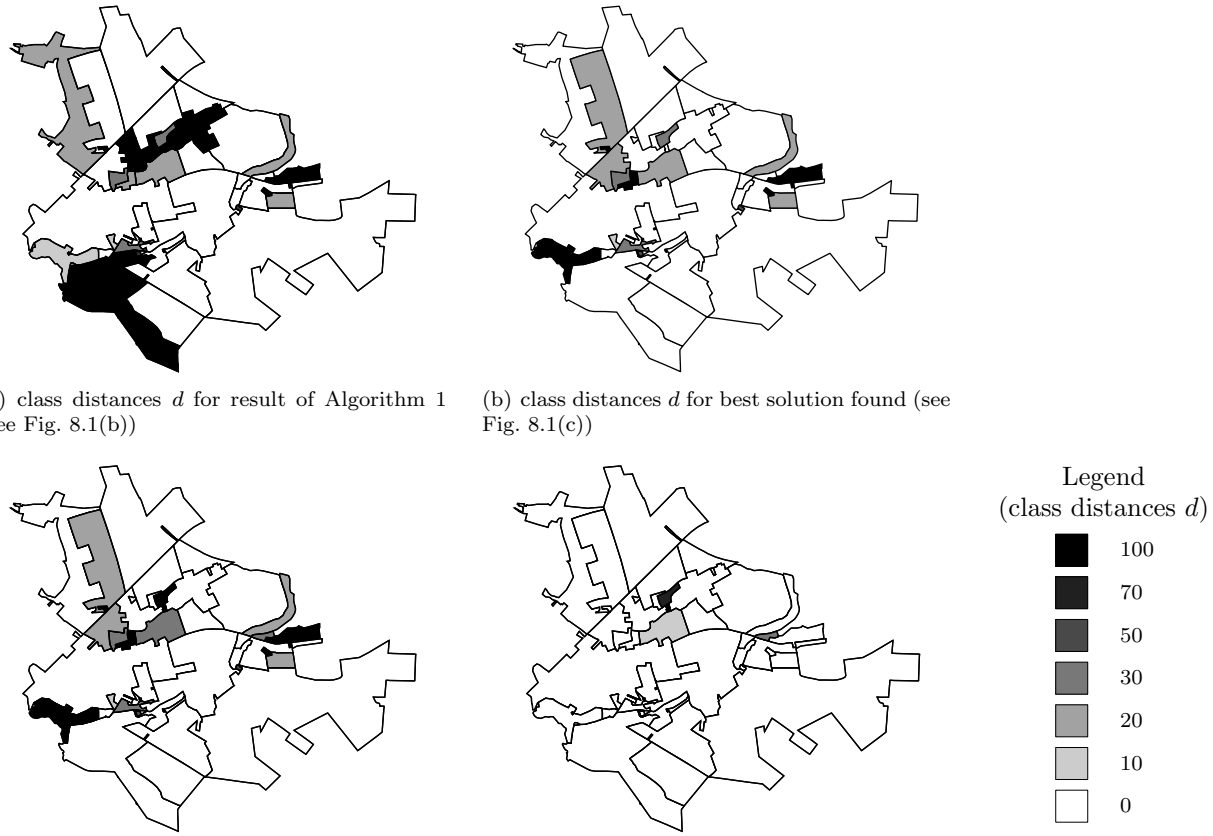
(c) best solution found (gap of 15%)

(d) result of P-R MIP with center and distance heuristic

Fig. 8.1: An instance with 50 nodes. The average class distances and the computation times for this instance are listed in line 3 of Table 8.1.

The computation time can be reduced with the center heuristic from Sect. 7.6.2. In our examples no differences to the optimal solution arise. Also by definition of the precedence relationship in Sect. 7.4.1 the computation time is reduced while the solution is only marginally affected. The computation time is greatly reduced by applying the center heuristic to the P-R MIP. Still the effects on the obtained results are acceptable. The distance heuristic is effective only for relatively large instances. This is plausible since long distances that allow variables to be eliminated, hardly exist in small datasets. Figure 8.1(d) shows the result of applying the P-R MIP with both heuristics. The average class distance is $\bar{d} = 5.64$, which is an increase of 9% compared to the exact solution to the Flow MIP. In order to identify the reason for the difference in cost, we can visualize the differences of semantic distances for single areas. Figure 8.2 displays distances of class changes as gray shades of areas, dark gray corresponds to expensive changes. In Fig. 8.2(d) we see the difference of both solutions: there are only four small areas whose classes differ from the classes of the optimal result. Comparing Fig. 8.1(c) and Fig. 8.1(d) we observe that an increase of cost by 9% only marginally affects the visual appearance of the map; we consider the result of the P-R MIP with the center and distance heuristic near-optimal.

With applicability on datasets of 400 areas the approach by mixed-integer programming becomes relevant for cartographic production. However, without additional heuristics, the developed MIP formulations do not allow a whole map sheet or a national database to be processed.



(a) class distances d for result of Algorithm 1 (see Fig. 8.1(b))

(b) class distances d for best solution found (see Fig. 8.1(c))

(c) class distances d for result with P-R MIP, center and distance heuristic (see Fig. 8.1(d))

(d) differences of class distances d for solution of P-R MIP and optimal result ((c) minus (b))

Fig. 8.2: Obtained class distances for the result in Fig. 8.1.

8.2 Performance tests of different heuristic approaches

As we cannot solve the developed MIPs for large problem instances, that is, thousands of polygons, we now discuss results of simulated annealing and results of the approach in several scale steps (Algorithm 5). Table 8.2 summarizes the results of simulated annealing for the instances that we discussed in the last section. These tests were performed on another computer with 1 GB RAM and a 2.8 GHz Intel Pentium 4-CPU, thus the results are not directly comparable to those in Sect. 8.1. However, the tests show that the quality of the results is similar to those obtained with the P-R MIP in combination with the center and distance heuristic. For small instances, we observe that the solution time is much lower with the mixed-integer programming approach. For the instance with 400 nodes, the simulated annealing method results in a solution with 5% higher cost, but it requires less time than is needed to solve the MIP.

We now discuss the results of our method with intermediate scale steps. Our basic assumption in Sect. 7.6.3 was that we obtain results of higher quality if we increase the number of areas that are evaluated in each step of an iterative merge procedure. This assumption motivated to extend Algorithm 1 by introducing an additional parameter K, that is, the number of areas that we evaluate in each iteration. This approach led to Algorithm 5.

To verify our basic assumption, we implemented Algorithm 5. For this we used the application programming interface of CPLEX (the ILOG CPLEX Callable Library). This allows the CPLEX optimization software to be embedded in Java applications. We tested this implementation with different settings of the parameter K. In a first test we solved the instance of 5537 nodes (the whole map sheet) with the objective of minimal class change. Setting $K = 1$ we obtained a solution with $\bar{d} = 6.14$. With this setting the method produces the same result as the iterative algorithm that selects the most compatible neighbor according to the defined cost function. The average class distance was reduced to $\bar{d} = 4.46$ when setting $K = 200$, which is a reduction by 37.5%. In this case the processing took 106 minutes. We also processed the same instance with simulated annealing. We set κ, the number of iterations, to $\kappa = 25\ 000\ 000$; with this we expected a similar running time. The actual running time was 92 minutes; the average class distance of the solution was $\bar{d} = 4.63$. This test shows that both methods perform similarly for large datasets, however, the deterministic method (Algorithm 5) still produces better results than simulated annealing.

We also tested a cost function that combines both objectives: class change and compactness. This means that we applied the combined cost function $f = s \cdot f_{\text{class}} + (1-s) \cdot f_{\text{comp}}$. We introduced this cost function in Eq. (5.4) to measure the semantic accuracy. Furthermore, we expressed the objective for compact shapes by combining the measures c_{dist} and c_{peri} with the additional weight factor s', that is, we set $c = s' \cdot c_{\text{dist}} + (1 - s') \cdot c_{\text{peri}}$ as defined in Eq. (7.9). For our test we applied the setting $s = 0.000715$ and $s' = 0.000015$ (given that areas are measured in m^2 and distances in m); these values resulted from tests that we discuss in the next section.

We applied Algorithm 5 with the combined cost function and five different values for K. Table 8.3 lists the results. It shows the average class distance $\bar{d}$ as well as the cost for class change, non-compact shapes, and their sum (normalized to 100). The resulting costs for non-compactness are very similar for different values of K. With $K = 200$ we only have 1.8% less costs than with $K = 1$. It seems that, concerning this objective,

nodes	iterations	time (s)	$\bar{d}_{SA}$	$\bar{d}_{SA}/\bar{d}_{\text{MIP}}$
30	30000	4.7	9.36	0.73
40	40000	5.4	7.26	0.96
50	50000	7.0	7.11	1.26
60	60000	8.1	6.21	0.93
100	100000	14.7	7.41	1.25
200	200000	33.9	7.85	1.68
300	300000	48.5	5.27	1.17
400	400000	61.3	4.88	1.05

Table 8.2: Experimental results with simulated annealing. Problem instances are the same as in Table 8.1. Column $\bar{d}_{SA}$ contains the average class distances obtained by simulated annealing. The last column contains the quotients of $\bar{d}_{SA}$ and $\bar{d}_{\text{MIP}}$, which refers to the average class distances obtained with the P-R MIP with the distance and center heuristic (last column of Table 8.1).

K	1	50	100	150	200
time (min)	3.86	2.05	8.66	26.06	82.27
$\bar{d}$	6.54	5.88	5.67	5.61	5.24
cost for class change	100	90.0	86.8	85.8	80.2
cost for non-compactness	100	99.4	98.0	98.8	98.2
total cost	100	96.2	94.3	94.5	92.2

Table 8.3: Experimental results with intermediate scales according to Sect. 7.6.3, considering class change and compactness of shapes. The processed dataset covers an area of 22 km × 22 km and contains 5537 polygons (nodes). The parameter K defines the maximal number of nodes v with $w(v) < \theta(\gamma(v))$ that are processed in a single iteration by application of the P-R MIP with heuristics. Computation times are in minutes CPU time. Objective values (costs) are normalized with respect to the results for $K = 1$.

the iterative algorithm does quite well by greedily choosing a neighbor. However, the decrease of costs for class change is still considerable (19.8% for $K = 200$ compared to $K = 1$). Combining both objectives, we obtain an improvement by 7.8%. These tests clearly confirm the assumption that we obtain results of higher quality if we consider more areas in a single iteration. Consequently, the parameter K should be set as high as possible under given time constraints.

Comparing the average class distance for both experiments with $K = 200$ we see that, in order to consider compactness according to the defined weight setting, we need to accept an increase of $\bar{d}$ by 17%.

8.3 Discussion of processed samples

We now discuss generalized datasets that were obtained with the developed aggregation method. Sections 8.1 and 8.2 have shown that the proposed heuristic methods produce near-optimal results, where 'optimal' refers to the defined objective function. However, the appropriateness of this objective function cannot be verified without visually inspecting and discussing various samples. As the approach with intermediate scales (Algorithm 5) allows large datasets to be processed, we discuss the results of this method, applying the parameter $K = 200$. We solved the occurring independent sub-instances with the P-R MIP and applied both the center as well as the distance heuristic. We compare the results of this optimization approach with those of the iterative region-growing procedure (Algorithm 1).

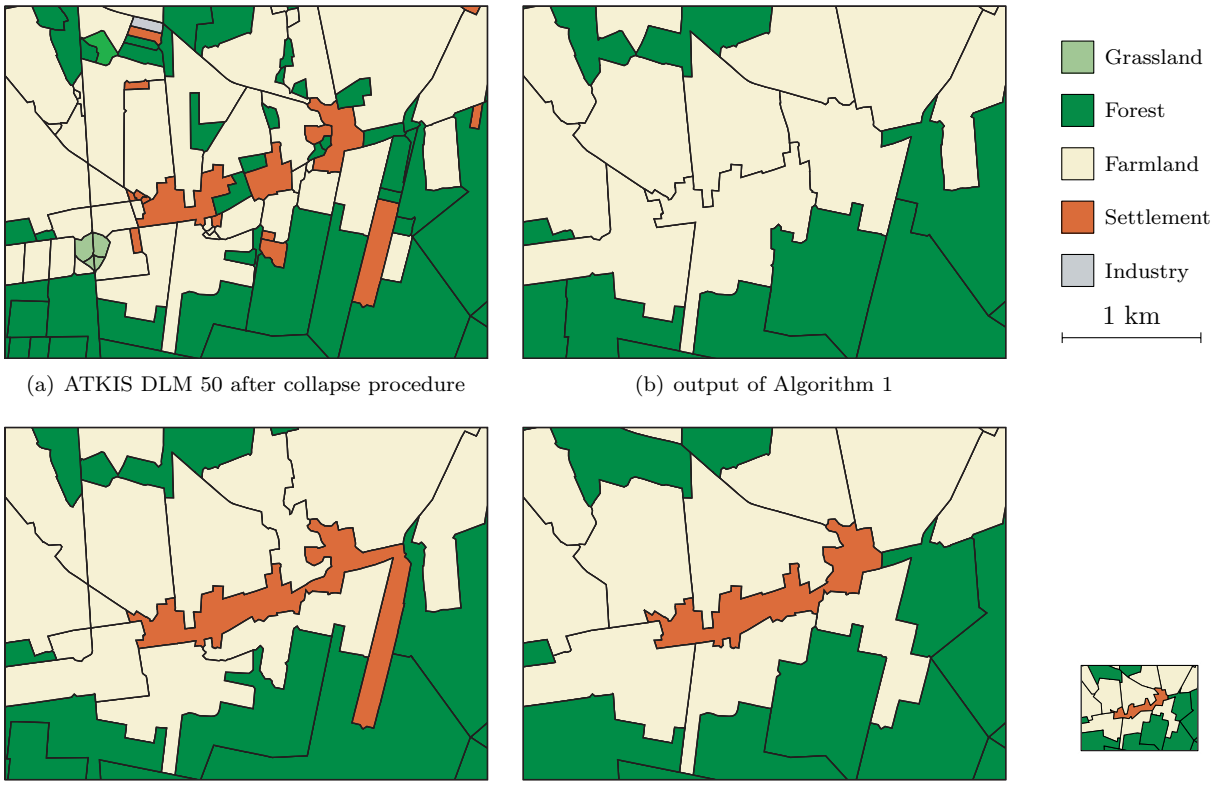
To define the trade-off between f_{class} and f_{comp}, as well as the trade-off between the compactness measures c_{dist} and c_{peri}, we tested ten different settings for the parameters s and s'. The setting $s = 0.000715$ and $s' = 0.000015$ satisfied our personal preferences best (given that areas are measured in m^2 and distances in m). In this case the average class distance $\bar{d}$ (and thus the cost for class change f_{class}) increased by 17% when compared to the result that was obtained without considering compactness. Of course our choice was influenced by subjective preferences, that is, other cartographers may apply different parameters. We first discuss the results that were obtained with the proposed setting, but then also discuss another setting.

Figures 8.3–8.6 show different samples from the ATKIS DLM 50. Each figure shows

- in subfigure (a) the sample after pre-processing with the collapse procedure,
- in subfigure (b) an aggregation result obtained by region-growing,
- two aggregation results of the optimization approach, namely
  - in subfigure (c) the result of minimizing class change and
  - in subfigure (d) the combined cost, and
- in subfigure (e) the result of applying the line simplification method to the latter aggregation result, shown at the target scale 1:250 000.

Figure 8.3 shows the same clipping as Fig. 1.1. There are some differences between Fig. 1.1 and Fig. 8.3(a) since the collapse procedure eliminated some narrow polygon parts. The pre-processed dataset in Fig. 8.3(a) is the input for the aggregation method. Applying Algorithm 1 results in the settlement being lost, as the input does not contain any sufficiently large, contiguous region of settlement areas (Fig. 8.3(b)). In contrast, the optimization approach allows most settlements to be saved by changing the classes of some connecting areas into settlement. This leads to a region of sufficient size for the target scale (Fig. 8.3(c)). However, minimizing changes of classes does not suffice to produce good generalization results: the small rectangular settlement in the right part of Fig. 8.3(a) was included in the region by creating a long, narrow corridor. This was needed to satisfy the constraint for connectivity. Obviously such complex shapes are not intended. Minimizing the combination of f_{class} and f_{path} leads to a better result (Fig. 8.3(d)). Including the rectangular settlement in the region would be too expensive, as the resulting boundary would be relatively long. Hence, the method produces a smaller and more compact region. The result is very similar to the manually generalized map in Fig. 1.2.

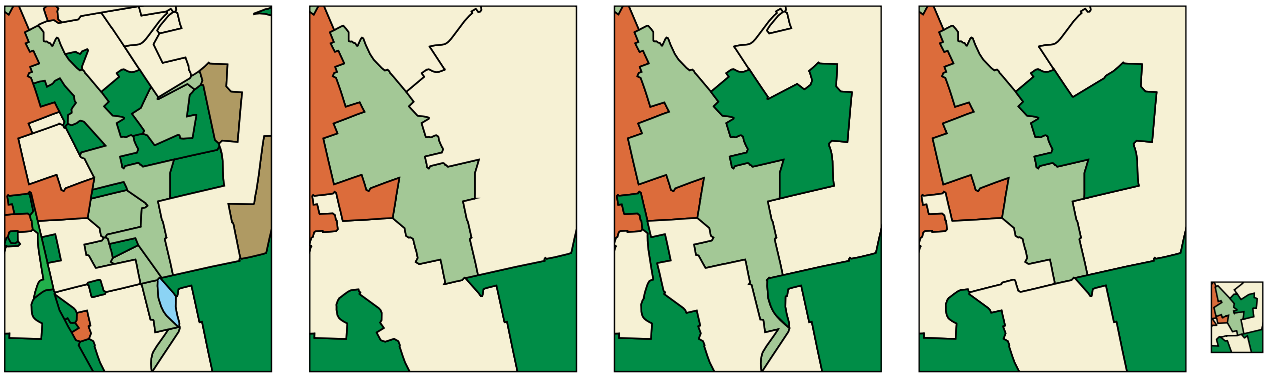
Figure 8.4 shows a sample with several forest areas. Algorithm 1 eliminates all forest areas in the upper part of the sample, as their size is insufficient for the target scale. This, however, results in relatively large amounts of class change. Applying the optimization method with the objective of minimum class change, leads to the result in Fig. 8.4(c). Now there is a forest of sufficient size in the upper part of the sample. This result is only possible as the forest includes parts that, in the input dataset, belong to other classes. Again the total change of classes is much lower than with Algorithm 1. However, we also find non-compact shapes in the result, for example, the forest in the lower left corner of the sample. Figure 8.4(d) shows the trade-off of minimum class change and maximum compactness; the result does not contain as many narrow polygon parts as before.



(a) ATKIS DLM 50 after collapse procedure (b) output of Algorithm 1

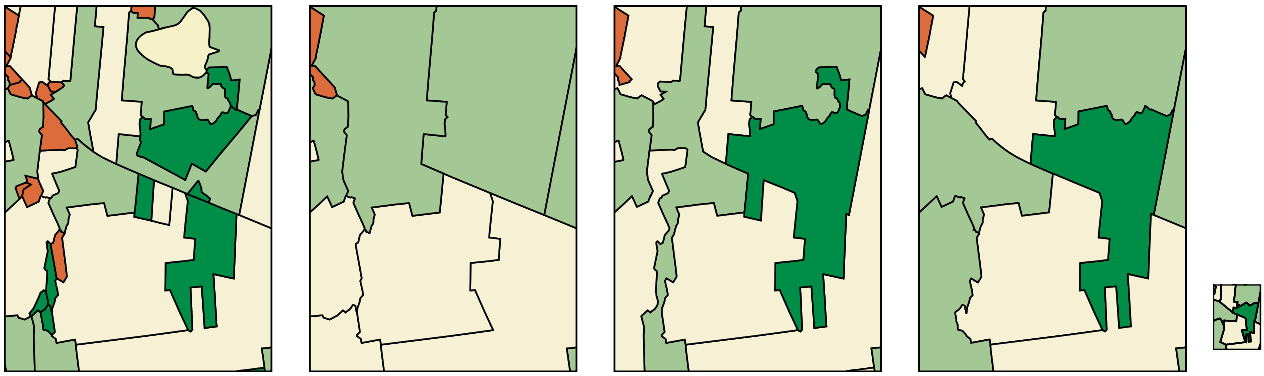
(c) minimizing class change (d) optimizing class change and compactness (e) lines simplified

Fig. 8.3: Input (a) and three aggregation results (b)–(d). The results in (c) and (d) were obtained using Algorithm 5, that is, applying the optimization approach with several scale steps.



(a) ATKIS DLM 50 after collapse procedure (b) output of Algorithm 1 (c) result of minimizing class change (d) optimizing class change and compactness (e)

Fig. 8.4: The optimization approach saves a forest that is slightly too small, by including areas belonging to other classes. Figure 8.4(e) shows the result in Fig. 8.4(d) after simplification of lines at scale 1:250 000.



(a) ATKIS DLM 50 after collapse procedure (b) output of Algorithm 1 (c) result of minimizing class change (d) optimizing class change and compactness (e)

Fig. 8.5: The optimization approach saves a set of forests by including a connecting grassland area into the forest aggregate. Figure 8.5(e) shows the result in Fig. 8.5(d) after simplification of lines at scale 1:250 000.

(a) ATKIS DLM 50 after collapse procedure (b) output of Algorithm 1 (c) result of minimizing class change (d) optimizing class change and compactness (e)

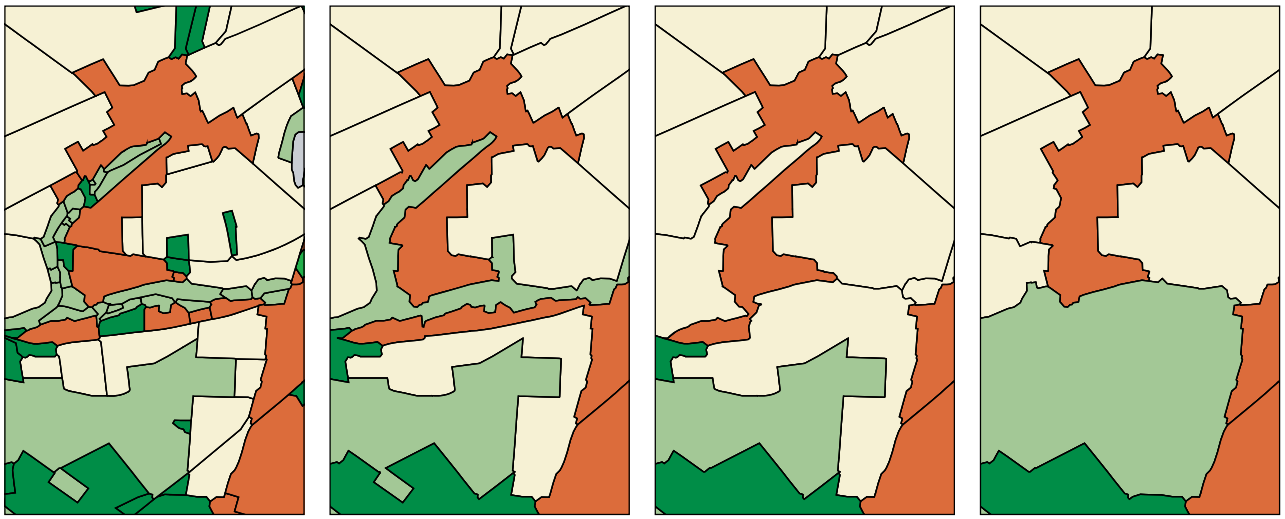
Fig. 8.6: The optimization approach saves a settlement by 'stealing' smaller forest areas. Figure 8.6(e) shows the result in Fig. 8.6(d) after simplification of lines at scale 1:250 000.

Figure 8.5 shows a similar example. The forest areas in the right part of Fig. 8.5(a) are lost when applying the simple iterative method (Fig. 8.5(b)), but can be saved using our approach by combinatorial optimization. The results with and without consideration of compactness (Figures 8.5(c) and 8.5(d), respectively) mainly differ in the left part of the sample. However, the forest in the right part appears slightly different in both results. In Fig. 8.5(d) two small forests were 'sacrificed': they changed into other classes, in order to create a short and simple outline.

The sample in Fig. 8.6 shows another interesting case. The settlement in the center is too small for the target scale, but can be saved by creating a connection to another settlement (Fig. 8.6(c)). This, however, results in a non-compact shape. Another possibility to save the settlement is to include small adjacent forest areas (Fig. 8.6(d)). This latter result reflects the idea of exaggerating important map features, which is a common approach in map generalization.

Next we discuss the results with different settings for the weight factors s and s', which were introduced to define the overall cost $f = s \cdot f_{\text{class}} + (1 - s) \cdot f_{\text{comp}}$ (see Eq. 5.4) and the cost $c = s' \cdot c_{\text{dist}} + (1 - s') \cdot c_{\text{peri}}$ for the non-compactness of a single region (see Eq. 7.9). Figure 8.7 shows a sample with three results of the optimization approach, each being obtained using different weights. Again, minimizing class change results in non-compact shapes (Fig. 8.7(b)). One settlement has an extremely long and narrow appendix and a non-compact grassland area separates two settlements. With the standard weights for different objectives ($s = 0.000715$, $s' = 0.000015$) we obtain, on the whole, more compact shapes (Fig. 8.7(c)). However, with this setting, the settlement in the

upper part of the sample gets a small appendix. Both parts of the settlement are connected only via a narrow constriction. We can avoid this problem by assigning higher weights to the objective for compactness. By setting $s = 0.000340$ and $s' = 0.000005$ we obtain very compact shapes (Fig. 8.7(d)). The decrease of s implies a higher preference for compactness; the decrease of s' means that the perimeter of a shape has a higher influence on the cost for compactness f_{comp}. Though the setting leads to more compact shapes it is questionable whether the result is an improvement. Of course, the compactness was improved at the expense of increasing class changes. For example, applying the higher weight for compactness to the sample in Fig. 8.6 led to a result similar to Fig 8.6(b), that is, the settlement was lost. For the complete map the higher weight for compactness leads to an increase of the average class distance $\bar{d}$ by 37.8%. On the whole, the setting that was used for the result in Fig. 8.7(c) was perceived to offer the best trade-off. In order to avoid two parts of the same aggregate being connected by only a narrow constriction, we could strictly forbid this case. For this we could simply exclude an edge from the adjacency graph $G(V, E)$ if the length of the corresponding boundary falls below a defined threshold. When applying the aggregation procedure to this graph, two areas can only be aggregated if they share a sufficiently long boundary. However, as compactness is not a hard constraint given by the database specifications, we did not follow this approach.



(a) input (ATKIS DLM 50 after pre-processing)

(b) minimizing class change ($s = 1$)

(c) standard weight setting ($s = 0.000715$, $s' = 0.000015$)

(d) more compact shapes ($s = 0.000340$, $s' = 0.000005$)

Fig. 8.7: A sample with results that were obtained by optimization of different objective functions.

Finally, Fig. 8.8 shows a larger sample that was processed using the developed method. To give an impression of the effectiveness of the proposed workflow of collapse, aggregation and simplification methods, it shows the original areas at scale 1:50 000 in Fig. 8.8(a), that is, the dataset before applying the collapse to narrow polygons. The output satisfies our idea of a good topographic database, which not only can be used to derive graphical maps, but is also suitable for applying analyses in a digital environment. The aim for a more abstract representation is clearly satisfied. The most dominant objects of the input dataset still exist in the result.

To improve the proposed method we could also consider additional classes of map objects, for example, roads that are represented by lines. As we generally aim to reduce the clutter on the map, we would probably prefer that region boundaries and roads are defined by the same lines. We could define this aim as a soft constraint in our mixed-integer programs: Based on the variables y_{ue} in the P-R MIP, we defined a benefit (a negative cost) for edges that are contained in regions. We could reduce this benefit (increase the cost) for edges that represent roads. Hence, it becomes more expensive to merge areas across roads. Furthermore, we could introduce hard constraints to forbid that two adjacent areas become part of the same region, if they are separated by a road.

Certainly, the result is not a perfect map for visualization, but this is not the aim of model generalization or database generalization. To derive a graphical map we would need to continue with methods for cartographic generalization. For example, we can apply a smoothing of lines to create a map that is visually more pleasing.

(a) input: areas from ATKIS DLM 50.

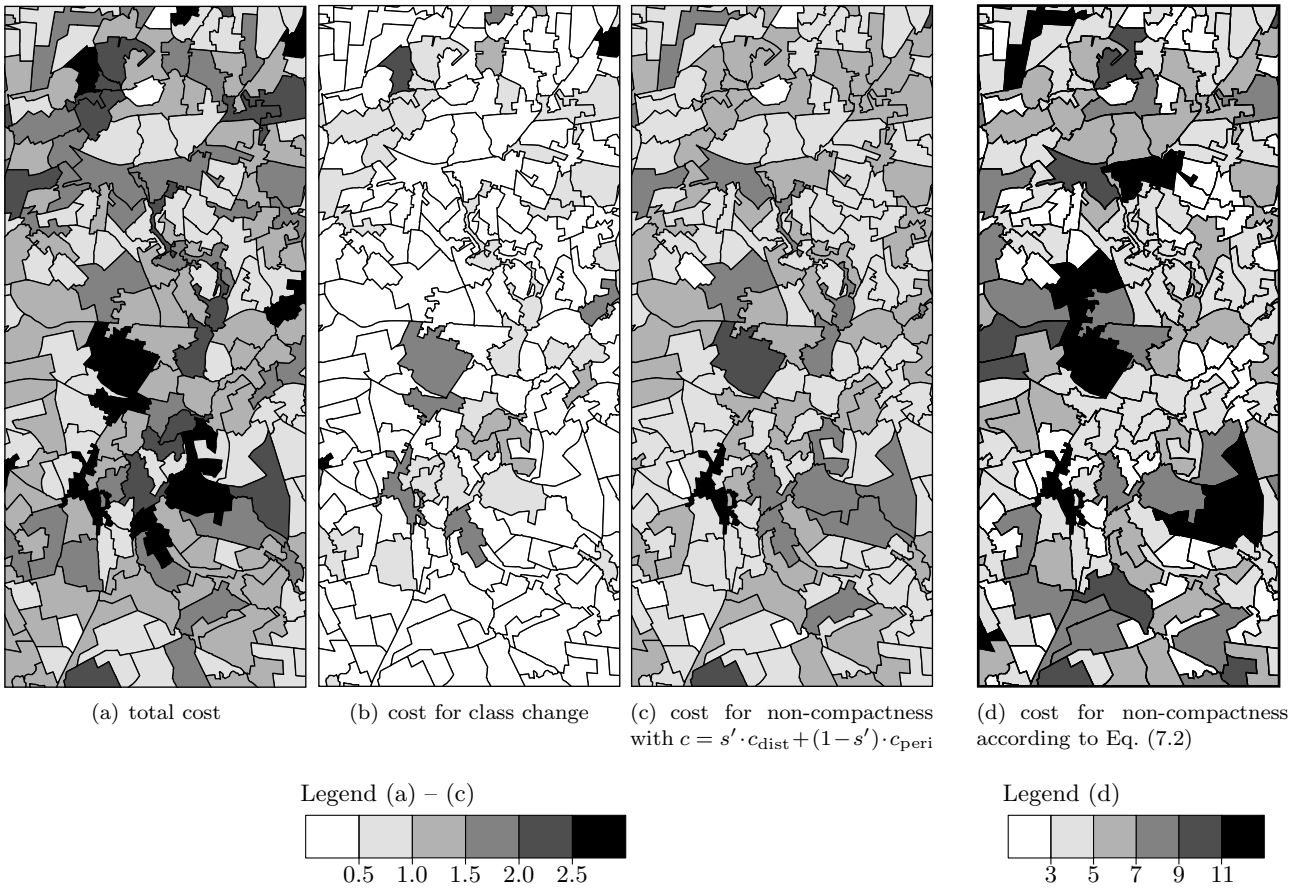
(b) output: areas satisfying specifications of ATKIS DLM 250.

Fig. 8.8: A sample processed using the presented methods for collapse, aggregation and line simplification. The aggregation was achieved using Algorithm 5, applying $K = 200$. The P-R MIP was used in conjunction with the center heuristic and the distance heuristic to solve the occurring problem instances. The clipping shows about 20% of the whole dataset. For the whole dataset, the aggregation required 82.27 min. Compared to the result with $K = 1$, the cost for class change f_{class} was reduced by 19.8%, the cost for non-compact shapes f_{comp} was reduced by 1.8%, and the overall cost f was reduced by 7.8%.

8.4 Quality assessment by visualization of cost and class distances

The proposed method yields generalized datasets together with costs that allow the quality of the result to be analyzed. In Fig. 8.2 we have already seen that the visualization of class distances helps the results of different methods to be compared. In this section we discuss further possibilities to make use of the defined quality measures.

Figure 8.2 showed class distances d for each area of the input dataset. If we wish to investigate the combined cost f for a solution, we need to do this on another level of abstraction. This is because the cost for non-compact shapes f_{comp} is charged for each region in the target scale. It is not reasonable to break this cost down into smaller parts. Figure 8.9(a) shows the result of applying Eq. (5.4) to each individual aggregate, that is, each area in the aggregation result for the sample in Fig. 8.8. The total cost f for the dataset equals the sum of these values. Now we can divide these costs into two parts: costs for class changes f_{class} (Fig 8.9(b)) and costs for non-compact shapes f_{comp} (Fig 8.9(c)). As dark shades correspond to high costs, expensive regions attract our attention. Especially concerning the cost for class change, a few regions make up most of the costs: for 75% of areas the cost for class change is below 0.5 – the remaining 25% of all regions make up 69% of the total cost for class change f_{class}. This does not allow us to conclude that the aggregation method failed to find near-optimal solution in these parts of the dataset. Nevertheless, we may agree on a result unless the absolute cost does not exceed a certain limit. Because of this, the visualization of costs supports the quality assessment. We can now focus on expensive parts of the dataset and optionally try to find better solutions. For example, we could apply the exact optimization method for this part of the dataset, that is, the method without heuristics. Alternatively, we could manually edit the data.



(a) total cost (b) cost for class change (c) cost for non-compactness with $c = s' \cdot c_{\text{dist}} + (1 - s') \cdot c_{\text{peri}}$ (d) cost for non-compactness according to Eq. (7.2)

Fig. 8.9: Areas from Fig. 8.8 classified according to their costs. The areas in (d) were classified according to the compactness measure c_2. In contrast to the measures c_{peri} and c_{dist} (which can be expressed using linear terms and were therefore applied in the optimization method) the measure c_2 would not affect the size of aggregates.

In Fig. 8.9(d) the regions are classified according to the compactness measure c_2 in Eq. (7.2), that is, the product of the perimeter of a shape and the square root of its area. In Sect. 7.2 we argued that applying this measure would be favorable, since it does not affect the size of composite regions. However, we defined the cost for non-compact shapes f_{comp} by combining c_{peri} and c_{dist} – in contrast to c_2, these measures allowed mixed-integer programming to be applied. This is because c_{peri} and c_{dist} can be modeled using linear terms. We can now compare the actually applied combination of c_{peri} and c_{dist} (Fig. 8.9(c)) with the favored measure c_2 (Fig. 8.9(d)). We observe that both definitions of compactness have similar effects: in both figures the same regions appear in dark shades. In fact, the correlation coefficient between both measures is 0.81, that is, they are highly correlated. We can consider the applied cost for non-compact shapes as a good approximation of c_2.

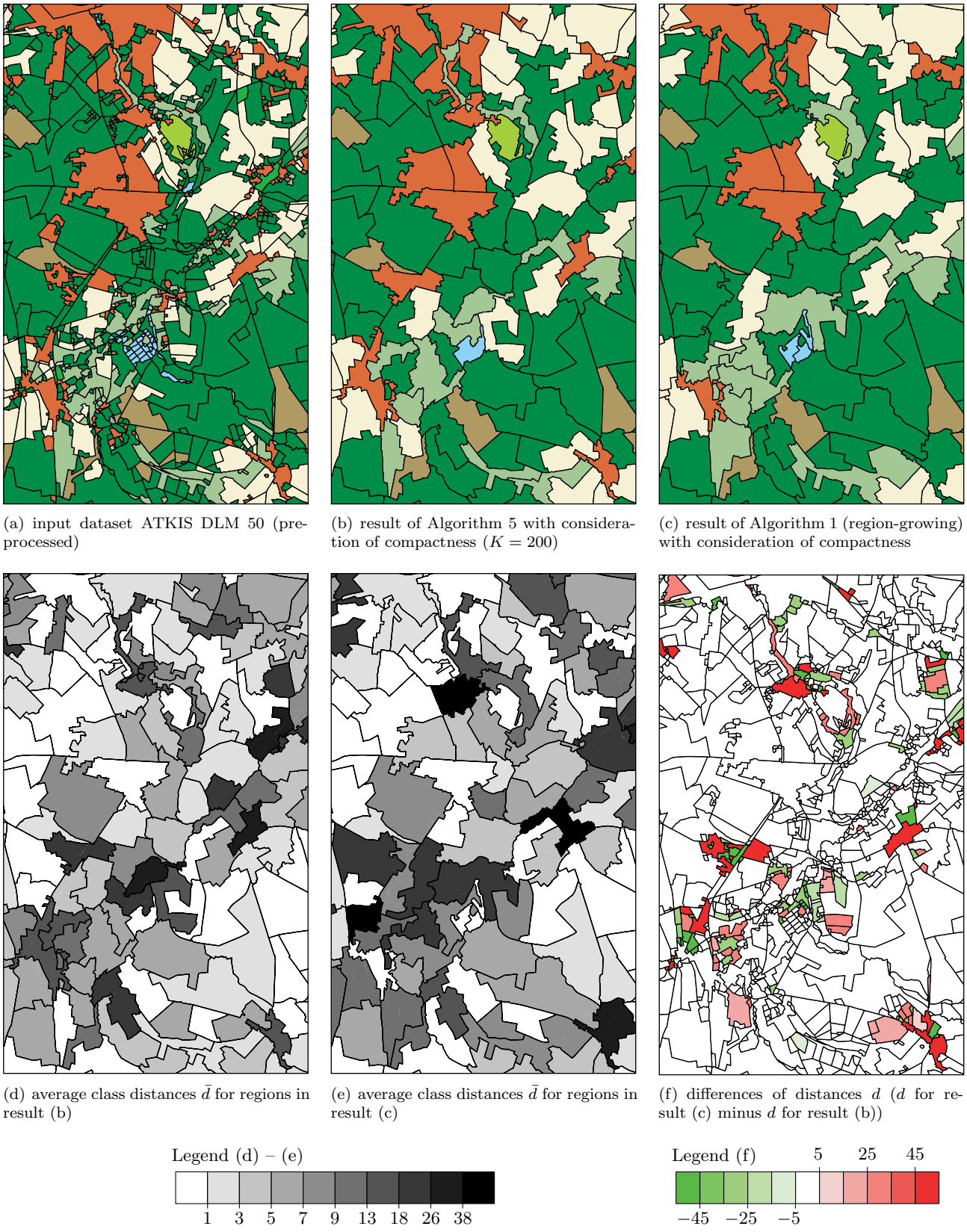
Finally, we visually compare results of Algorithm 5 that were obtained with $K = 200$ (Fig. 8.10(b)) and $K = 1$ (Fig. 8.10(c)). The sample is part of the dataset in Fig. 8.8. For each obtained region, Figures 8.10(d) and 8.10(e) show the average class distance $\bar{d}$. The distances d can be compared using the most detailed level, that is, for each area of the original dataset (Fig. 8.10(f)). Areas whose classes were kept more similar when setting $K = 1$ are green; red means that the classes were kept more similar with $K = 200$. Since we have seen earlier that a higher value of K yields a better solution, it is clear that red colors dominate the figure. Nevertheless, the figure reveals an interesting pattern: usually, red and green areas are in close vicinity to each other. In these parts of the dataset the optimization approach was able to keep the class of relatively large areas (red) by sacrificing smaller ones (green). Normally, such areas form isolated groups that have a limited extent. This indicates that good solutions can be obtained by optimally handling configurations with a small number of areas. Handling several hundreds of areas at a time, as done by Algorithm 5, is certainly a reasonable approach to find such solutions.

8.5 Summary

We summarize the most important results from this chapter:

- For samples of up to 50 areas the exact optimization approach (the Flow MIP) reduces the average class distance $\bar{d}$ by 64% compared to simple region-growing.
- An exact solution to larger instances is not possible as the time consumption increases exponentially.
- The P-R MIP in conjunction with the center heuristic and the distance heuristic can be solved for instances of several hundred areas (approx. 400) in modest time; the average class distance $\bar{d}$ increases only by approx. 9%, compared to the optimal solution.
- For large instances (a whole map sheet) we can apply simulated annealing or the iterative approach in several scale steps (Algorithm 5). With $K = 200$ the method with intermediate scales turned out to perform slightly better (-4% for $\bar{d}$).
- Optimizing class change, the iterative method consumes 37.5% less cost than region-growing. Optimizing a combined objective, it reduces the class change by 19.8% and the cost for non-compact shapes by 1.8%.
- In order to obtain reasonably compact shapes, we need to accept an increase of class change by 17.8%.

We discussed the effect of applying different methods and different objective functions on examples. Furthermore, we have seen that classifying areas and composite regions according to the defined objective functions helps to visually assess the quality of the result. The advantage of the developed optimization approach is due to its capability of sacrificing unimportant areas (that is, allowing their classes to be changed), in order to save more important ones (that is, to avoid expensive class changes).



(a) input dataset ATKIS DLM 50 (pre-processed)

(b) result of Algorithm 5 with consideration of compactness ($K = 200$)

(c) result of Algorithm 1 (region-growing) with consideration of compactness

(d) average class distances $\bar{d}$ for regions in result (b)

(e) average class distances $\bar{d}$ for regions in result (c)

(f) differences of distances d (d for result (c) minus d for result (b))

Fig. 8.10: A sample from the ATKIS DLM 50 with results of two different aggregation methods and resulting average class distances $\bar{d}$ for the obtained regions.

Chapter 9

Conclusion and Outlook

9.1 Conclusion

In order to conclude the thesis we repeat and answer our research questions from Sect. 1.4.

How can we measure quality, in particular semantic accuracy, in generalization?

We have seen that two main criteria are important for good aggregation results: class changes should be small and shapes should be compact. To quantify class change, a general distance measure between classes was proposed. We did not require symmetric distances, which was motivated by examples. A set-up of class distance measures was proposed, which incorporates expert knowledge and the class hierarchy given with the data model. Compactness was expressed by two sub-objectives, that is, shapes should have a short outline and distances to a center should be small. The defined cost function allows a quality assessment on three levels of aggregation: for each area in the input map, for each composite area in the output and, as a global measure of quality, for the entire dataset. Initial tests of the developed method showed that minimizing class change as the sole objective results in unwanted long and narrow connections. In order to define a good trade-off between the aims of compactness and minimal class change, an increase of class change by 17.8% was required.

For land cover data in a planar subdivision, the defined criteria are good indicators of spatial data quality. If we consider other generalization problems, additional criteria become important. However, these still can be subsumed by the known elements of spatial data quality. As discussed in the context of related work on aggregation and grouping, preserving patterns of objects is important, for example, alignments of houses and grid structures in road networks. Whether a map correctly reflects such patterns is also a matter of semantic accuracy. For this, additional quantitative measures are needed, which we discuss as an aim for future research.

According to these criteria, how can we approach generalization in order to create datasets of high quality, that comply with database specifications?

Obviously, this question can be answered briefly: by optimization. However, we need to specify this answer and put it into perspective. In order to approach the problem by optimization we need a clear distinction of soft constraints and hard constraints. The minimal dimensions given with database specifications, the contiguity of composite regions, and the simplicity of the subdivision are examples for hard constraints, which have been discussed in detail. Semantic accuracy is a soft constraint, which needs to be expressed by the objective function. As generalization includes discrete tasks like selection it is a constrained combinatorial optimization problem. Handling the whole generalization problem with all possibilities of data manipulation in one comprehensive, global optimization approach seems too ambitious. Because of this, three generalization operators have been independently developed: (1) collapse of areas and area parts to lines, (2) aggregation, and (3) line simplification. Though the developed algorithms were applied in succession, the same components could be used in other workflows. In particular, existing approaches for the integration of different generalization operators could be used to decide when to apply one of the developed methods, for example, the multi-agent approach of Barrault *et al.* (2001). Though the problem is broken down into three generalization operators, optimization is the most promising approach to obtain results of high quality.

Which techniques for combinatorial optimization and spatial allocation can we apply for aggregation, what are their benefits and limitations, and how can we improve on them?

To answer this question we first investigated the complexity of area aggregation. Due to its NP-hardness we focused on mixed-integer programming and heuristics. Two different heuristic approaches have been developed: a method by simulated annealing and a method that uses specialized heuristics in conjunction with the developed MIPs. Table 9.1 compares the mixed-integer programming approach and the approach by simulated annealing, according to their advantages and disadvantages.

Mixed-integer programming	Simulated annealing
+ allows all relevant hard constraints to be modeled.	– requires size constraint to be relaxed.
– restricts to linear objective functions.	+ allows non-linear objective functions.
+ is deterministic; results are reproducible.	– is randomized; results are influenced by chance.
+ offers exact optima for small samples.	– does not offer globally optimal solutions.
+ allows an exact solution without tuning parameters.	– requires definition of an annealing schedule.

Table 9.1: Advantages and disadvantages compared for the developed aggregation approaches by mixed-integer programming and simulated annealing.

The restriction of mixed-integer programming to linear objective functions is problematic, since size-invariant compactness measures cannot be applied. The problem was handled by fixing a set of region centers. With this approach, the alternative compactness measures had a limited influence on the size of regions.

Several components of existing MIPs for spatial allocation could be adapted; however, the newly developed Flow MIP turned out to be the best performing exact method. To improve the performance of these methods, several new heuristics have been developed. In conjunction with the center heuristic and the distance heuristic, the P-R MIP allowed near-optimal solutions to be produced for instances of approximately 400 areas, in modest time. Applying the P-R MIP in conjunction with the heuristic method that uses intermediate scale steps, we still obtain better results than with simulated annealing. Due to this reason and the majority of benefits in Table 9.1 on side of the mixed-integer programming approach, this method should be favored. Furthermore, the mixed-integer programming approach allows existing optimization software to be used.

How competitive are these techniques under realistic conditions, that is, if large datasets need to be processed and derived data needs to be frequently updated?

Firstly, we have seen that region-growing, that is, iteratively merging areas while only considering their direct neighbors, is not sufficient to find good results. We need to accept that high-quality digital landscape models cannot be produced in seconds. Hence, whether we can produce well-generalized DLMs is also a question of whether we are willing to spend additional minutes or hours for data processing. If we consider the time and the effort that is needed to collect the data, the processing time should not be our major concern. However, due to the NP-hardness of the aggregation problem and the exponential time performance of MIP solvers, we can only solve very small problem instances exactly (about 50 areas), even if we wait for several days; of course, this is not acceptable. Therefore, good heuristics are needed for map generalization. In order to process very large datasets, we need to define some kind of partitioning scheme that allows the aggregation method to be applied to independent subsets. Supposably, partitioning the dataset into arbitrary grid cells would be too rigorous. In contrast, when applying the center heuristic, a problem instance directly decomposes into several smaller instances, which means that the partitioning scheme is inherent to the data. The obtained problem instances can be processed independently, that is, in parallel. When allowing intermediate scale steps to further decrease the size of the occurring problem instances, the method scales well with the size of the input. Processing a complete map sheet (22 km × 22 km, 5537 areas) required 82 minutes, which we consider appropriate for applications. Compared to the pure region-growing approach, it consumed 37.5% less costs when defining class change as the sole objective. Optimizing a combined objective, it reduced class change by 19.8% and the cost for non-compact shapes by 1.8%.

We can solve the updating problem with the same approach, that is, we can use the partitioning scheme based on the center heuristic. Among the independent problem instances, we only need to solve those instances again that

contain updated areas, for example, areas whose classes have changed. This incremental approach guarantees a result that equals the result of processing the whole dataset. However, as the occurring problem instances can be large, it is reasonable to permanently maintain an intermediate scale. In this case, the occurring problem instances become smaller. We conclude that the developed method not only allows toy problems to be solved, but also copes with realistic conditions.

In conclusion, the developed aggregation method by mixed-integer programming guarantees results that satisfy requirements from database specifications. With respect to a cost-based measure of semantic accuracy, it offers optimal solutions for small samples. Large datasets can be processed by applying specialized heuristics; the obtained results are clearly better than those from the existing iterative algorithm. The defined quality measures allow the quality of the result to be assessed, both on the object level and for the whole dataset.

9.2 On-going research work

Several directions can be explored to continue this work. Two concrete problems are addressed by on-going research. For both problems, initial solutions were obtained that encourage further developments into similar directions: quality measures need to be defined and optimization approaches need to be developed, in order to obtain generalization results of high quality.

Figure 9.1 illustrates one of these two problems. A common approach to submit a spatial dataset from a server to a mobile client is to send a coarse representation first, which then is gradually refined. With this approach, users have fast access to coarse representations, which they can use, for example, to browse to their area of interest. This problem not only requires well-generalized datasets at a particular scale, but also asks for intermediate representations. When loading more detailed data, the already sent data should be reused. The simple iterative algorithm (Algorithm 1) has the nice property that it defines a gradual reduction of detail. The reversed sequence of operations defines a gradual refinement, which can be used for progressive data submission and zooming (van Oosterom, 2005). Unfortunately, we have seen that the simple algorithm does not produce generalization results of high quality. In contrast, the optimization approach produces better results for a particular scale, but does not define a sequence of incremental refinement operations. We therefore aim to combine the benefits of both methods (Haunert *et al.*, 2008).

A simple approach is to proceed in two steps. Firstly, a small-scale representation is generated using the generalization method that was developed in this thesis. Secondly, an iterative procedure similar to Algorithm 1 is provided with the detailed dataset and the result that was obtained by optimization. In contrast to Algorithm 1 it is constrained to apply merge operations that yield the given result. Again, the procedure yields a sequence that can be used for gradual refinement. However, we are sure that we have a well-generalized map at smallest scale. The result in Fig. 9.1 was obtained with this approach.

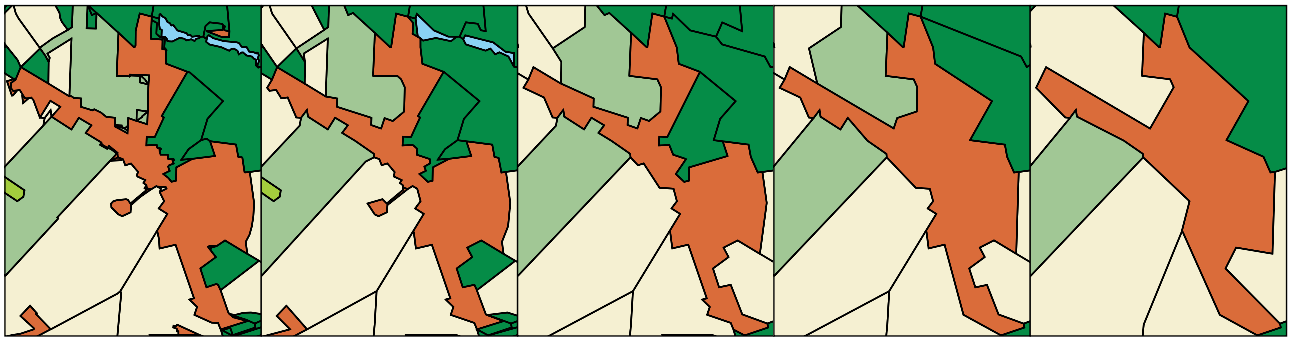
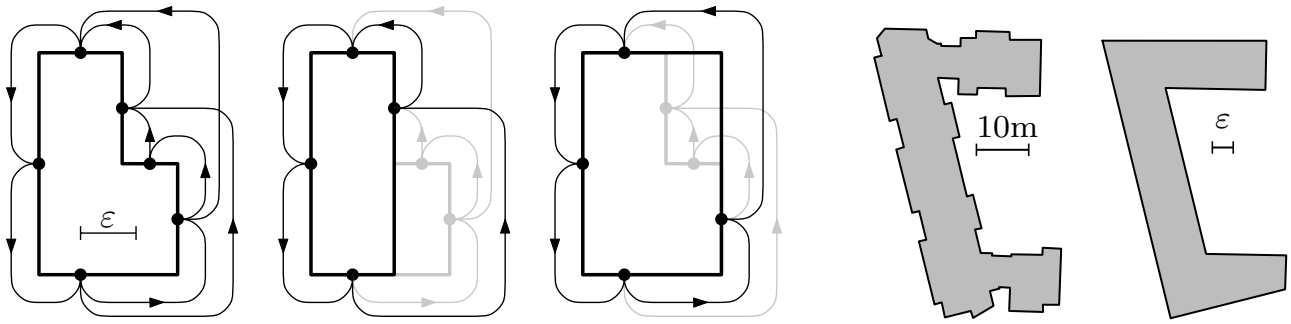


Fig. 9.1: A sequence of representations that can be used for progressive data submission via the Internet. The least detailed dataset (right) was obtained by generalization of the most detailed dataset (left), using the methods that were developed in this thesis. To generate intermediate representations, we defined a simple algorithm, which is similar to Algorithm 1 (Haunert *et al.*, 2008).

A problem of this simple method is that we may produce improper intermediate levels of generalization, that is, we optimally generated the representation at smallest scale, but there are still several possibilities to define a sequence that yields this result. Hence, we should also clearly define the quality of a generalization sequence. For example, we certainly would prefer generalization sequences in which each scale-step implies a similar cost for class change. If we obtained the cost for class change f_{class} for the smallest scale, then a sequence of n scale-steps would be perfect, if each of them implies a class change of $(f_{\text{class}})/n$. This would reflect the idea of a gradual reduction of detail. However, with our approach it is possible that a single step implies a relatively high cost for class change. To avoid this problem, it would be reasonable to define a sequence that yields the optimal result at smallest scale and, subject to this hard constraint, minimizes the maximal cost for class change that is charged in a single step. The current method does not allow such a sequence to be produced, but again the problem could be approached by optimization.

Figure 9.2 shows a second problem, the simplification of building ground plans. Though several approaches exist to attack this task we have started to develop a new approach (Haunert & Wolff, 2008). However, we do not aim at yet another building simplification method. In contrast to other methods, our method allows different optimization objectives and constraints to be applied. We assume that comparing the results will help to better understand the criteria that make up a well-generalized building. So, in addition to a new simplification method, we will gain measures for quality assessment.



(a) an input polygon and its shortcut graph

(b) a solution and its corresponding cycle in the shortcut graph

(c) another solution and its corresponding cycle

(d) a more complex input polygon

(e) building (d) after simplification

Fig. 9.2: Building simplification based on a shortest cycle formulation after Haunert & Wolff (2008). The parameter ε defines the error tolerance.

Our approach is similar to the optimization approach to line simplification from Sect. 6.2. In contrast, we reduce a building outline to a subsequence of its edges and not to a subsequence of its vertices. We allow the selected edges to be shortened or extended in order to generate a closed outline. With this approach we keep the edge slopes fixed and so give consideration to shape regularities. For example, if the original building is rectilinear, the simplified building will automatically be rectilinear, too. Certain requirements must hold for the simplified building, in particular, we do not allow self-intersections and we require a limited positional error ε. Our basic approach is to minimize the number of edges in the simplified building. To solve this problem we introduced a shortcut (di)graph. This contains an arc for each potential shortcut, which is a pair of edges that can become consecutive edges in the simplified building. The shortest cycle in the graph that satisfies a certain set of hard constraints defines the optimally simplified building. In our first publication of this approach (Haunert & Wolff, 2008), we also proved that the problem is NP-hard if the original building contains multiple holes and if we require simple polygons in the output. Therefore, we again proposed solutions by integer programming and heuristics.

A minimum number of edges can be important for data compression. However, to produce well-generalized building outlines, other quality criteria are important. Corners with 90 degree angles are typical for buildings, and so they should be preferred. Our model allows non-uniform costs to be introduced for arcs in the shortcut graph. A shortcut that implies a corner of 90 degrees should be relatively cheap. However, future research is still needed to find an appropriate cost setting.

9.3 General recommendations for future research

Though many generalization methods exist, still little is known about how to quantify the quality of the obtained results. Recent developments of pattern recognition techniques for vector data, for example, the approaches of Heinzle & Anders (2007) and Lüscher *et al.* (2007), will certainly improve the results of automatic generalization methods. However, we need to find good measures to appropriately characterize patterns. For example, rather than defining ad hoc solutions to find grid structures in road networks, a measure of 'grid-likeness' should be introduced. Then, similar to our optimization approach to find optimally compact shapes, we can also imagine optimization methods to find maximally 'grid-like' subsets of roads in the road network. To improve the proposed aggregation method, we could introduce shape measures of 'grassland-likeness', 'settlement-likeness', and 'lake-likeness' rather than a general shape measure for compactness. However, in order to quantify changes that are due to generalization, we must not only describe characteristic features in one representation, but also need to define additional similarity or distance measures to compare representations. For example, we could quantify the differences of two datasets with respect to the similarity of the contained grid structures. Such approaches will have impact beyond automatic map generalization. They will also be useful for related problems like data matching and integration and, furthermore, support more general spatial analysis and data mining tasks. For example, different designs of a planned road network could be compared with respect to their grid-likeness, in order to decide on one option.

Ideally, the definition of appropriate measures is independent from the selection of a particular optimization technique. However, when formalizing the aggregation problem, the difficulty to express size-invariant compactness measures by mixed-integer programming has guided our choice to a less perfect measure. When approaching any real-world problem by optimization we need to be aware that a model can only reflect parts of the reality. Such simplifications are justifiable if they prevent a problem from becoming intractable or severely more difficult. Nevertheless, they need to be laid open. In order to allow for a high flexibility in handling hard and soft constraints, research on map generalization requires more insight into theoretical aspects of optimization. Further research is needed to elaborate the applicability of different optimization approaches.

Probably, the most explored optimization approach to map generalization is the application of meta-heuristics, for example, simulated annealing. However, often it is difficult to find an appropriate neighborhood that allows hard constraints to be handled. Furthermore, it is questionable whether the non-deterministic nature of simulated annealing is acceptable. With respect to the quality element lineage, we need to provide users with meta-information about applied map transformations such that reasons for changes are comprehensible. If we consider map generalization as a special map transformation we must not allow the results to be influenced by chance. Of course map generalization is more difficult to formalize than other map transformations, for example, geometric map projections. However, with deterministic optimization approaches to map generalization, we can concisely define how a database of reduced scale was derived from a given source. For example, as dependencies between both datasets become well-defined, we have the possibility of finding incremental methods for updating, that offer the same result as the complete generalization method. Therefore, these approaches need to be further developed.

Since quality assurance in map generalization is an important problem, we should not only consider heuristic approaches. Especially, the development of approximation algorithms and approximation schemes is promising. These give a performance guarantee, but also allow NP-hard combinatorial problems to be tackled. In particular, this thesis may encourage other researchers to find good approximation algorithms for AreaAggregation.

Appendix A

The P-R MIP for an Instance of AreaAggregation

To simplify the reproduction of the developed method, this appendix presents the P-R MIP for a simple instance of the aggregation problem. Figure A.1(a) shows this instance, which includes six areas of two different classes. The areas need to be aggregated, in order to satisfy the defined thresholds θ for the target scale. The optimization objective is to minimize class change. Figure A.1(b) shows the optimal solution: Only two small areas needed to change their classes, in order to end up with weight-feasible and contiguous regions.

(a) an instance of AreaAggregation

(b) the optimal solution for the instance in (a)

Fig. A.1: A processed example.

Table A.1 shows the P-R MIP for this instance in CPLEX LP file format, which can be read by the optimizers CPLEX and lp_solve. Only the definition of binary variables was truncated, in order to save space. The variable `x_i_j` refers to the variables $x_{v_i v_j}$ according to the definition in Sect. 7.4. The constraints `con1`–`con30` ensure the connectivity according to the precedence relationship, that is, they are equivalent to the constraint in Eq. (7.15). Constraints `con31`–`con36` ensure that a node is assigned to exactly one center, which corresponds to Eq. (7.11). Finally, constraints `con37`–`con42` ensure that the regions are weight-feasible, that is, the constraints are equivalent to Eq. (7.12).

Assuming that this IP is stored in a file 'input.lp', the following command can be used to solve the problem with lp_solve:

```
lp_solve -rxli xli_CPLEX input.lp > output.txt
```

The solution is written to file 'output.txt', which is shown in Table A.2. The values of the variables correspond to the solution in Fig. A.1(b). The centers according to this solution are displayed as squares.

Table A.1: The P-R MIP for the instance in Fig. A.1(a), formatted according to CPLEX LP file format.

```
Minimize
obj:
 + 3 x_2_1 + 3 x_4_1 + 3 x_6_1
 + 2 x_1_2 + 2 x_3_2 + 2 x_5_2
 + 1 x_2_3 + 1 x_4_3 + 1 x_6_3
 + 1 x_1_4 + 1 x_3_4 + 1 x_5_4
 + 2 x_2_5 + 2 x_4_5 + 2 x_6_5
 + 3 x_1_6 + 3 x_3_6 + 3 x_5_6

Subject To

con1: x_1_2 -  x_1_1 <= 0
con2: x_1_3 - x_1_2 <= 0
con3: x_1_4 -  x_1_1 <= 0
con4: x_1_5 - x_1_2 - x_1_4 <= 0
con5: x_1_6 - x_1_3 - x_1_5 <= 0
con6: x_2_1 -  x_2_2 <= 0
con7: x_2_3 -  x_2_2 <= 0
con8: x_2_4 - x_2_5 <= 0
con9: x_2_5 -  x_2_2 <= 0
con10: x_2_6 - x_2_3 - x_2_5 <= 0
con11: x_3_1 - x_3_2 <= 0
con12: x_3_2 -  x_3_3 <= 0
con13: x_3_4 - x_3_5 <= 0
con14: x_3_5 - x_3_2 - x_3_6 <= 0
con15: x_3_6 -  x_3_3 <= 0
con16: x_4_1 -  x_4_4 <= 0
con17: x_4_2 - x_4_1 - x_4_5 <= 0
con18: x_4_3 - x_4_2 <= 0
con19: x_4_5 -  x_4_4 <= 0
con20: x_4_6 - x_4_5 <= 0
con21: x_5_1 - x_5_2 - x_5_4 <= 0
con22: x_5_2 -  x_5_5 <= 0
con23: x_5_3 - x_5_2 <= 0
con24: x_5_4 -  x_5_5 <= 0
con25: x_5_6 -  x_5_5 <= 0
con26: x_6_1 - x_6_2 - x_6_4 <= 0
con27: x_6_2 - x_6_3 - x_6_5 <= 0
con28: x_6_3 -  x_6_6 <= 0
con29: x_6_4 - x_6_5 <= 0
con30: x_6_5 -  x_6_6 <= 0

con31:  + x_1_1 + x_2_1 + x_3_1 + x_4_1 + x_5_1 + x_6_1 = 1
con32:  + x_1_2 + x_2_2 + x_3_2 + x_4_2 + x_5_2 + x_6_2 = 1
con33:  + x_1_3 + x_2_3 + x_3_3 + x_4_3 + x_5_3 + x_6_3 = 1
con34:  + x_1_4 + x_2_4 + x_3_4 + x_4_4 + x_5_4 + x_6_4 = 1
con35:  + x_1_5 + x_2_5 + x_3_5 + x_4_5 + x_5_5 + x_6_5 = 1
con36:  + x_1_6 + x_2_6 + x_3_6 + x_4_6 + x_5_6 + x_6_6 = 1

con37: 3 x_1_1 - 2 x_1_2 - 1 x_1_3 - 1 x_1_4 - 2 x_1_5 - 3 x_1_6 <= 0
con38: 4 x_2_2 - 3 x_2_1 - 1 x_2_3 - 1 x_2_4 - 2 x_2_5 - 3 x_2_6 <= 0
con39: 5 x_3_3 - 3 x_3_1 - 2 x_3_2 - 1 x_3_4 - 2 x_3_5 - 3 x_3_6 <= 0
con40: 5 x_4_4 - 3 x_4_1 - 2 x_4_2 - 1 x_4_3 - 2 x_4_5 - 3 x_4_6 <= 0
con41: 4 x_5_5 - 3 x_5_1 - 2 x_5_2 - 1 x_5_3 - 1 x_5_4 - 3 x_5_6 <= 0
con42: 3 x_6_6 - 3 x_6_1 - 2 x_6_2 - 1 x_6_3 - 1 x_6_4 - 2 x_6_5 <= 0

Binaries

x_1_1
[...]
x_6_6

End
```

Table A.2: Output of the optimizer lp_solve for the input in Table A.1.

```
set_XLI: Successfully loaded 'xli_CPLEX'

Value of objective function: 2

Actual values of the variables:
x_2_1                           0
x_4_1                           0
x_6_1                           0
x_1_2                           0
x_3_2                           0
x_5_2                           0
x_2_3                           1
x_4_3                           0
x_6_3                           0
x_1_4                           0
x_3_4                           0
x_5_4                           1
x_2_5                           0
x_4_5                           0
x_6_5                           0
x_1_6                           0
x_3_6                           0
x_5_6                           0
x_1_1                           0
x_1_3                           0
x_1_5                           0
x_2_2                           1
x_2_4                           0
x_2_6                           1
x_3_1                           0
x_3_3                           0
x_3_5                           0
x_4_4                           0
x_4_2                           0
x_4_6                           0
x_5_1                           1
x_5_5                           1
x_5_3                           0
x_6_2                           0
x_6_4                           0
x_6_6                           0
```

Bibliography

Abramson, D., Dang, H., & Krishnamoorthy, M. 1996. A comparison of two methods for solving 0–1 integer programs using a general purpose simulated annealing algorithm. *Annals of Operations Research*, **63**(1), 129–150.

AdV. 2003. *ATKIS-Objektartenkatalog*. `http://www.atkis.de` (2007/11/09).

Aerts, J. C. J. H., & Heuvelink, G. B. M. 2002. Using simulated annealing for resource allocation. *International Journal of Geographical Information Science*, **16**(6), 571–587.

Afflerbach, S., Illert, A., & Sarjakoski, T. 2004. The harmonisation challenge of core national topographic data bases in the EU-project GiMoDig. *Pages 129–134 of: Proceedings of the XXth ISPRS Congress, July 12–23, 2004, Istanbul, Turkey*. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXV(Part B4:IV).

Ahlquist, O. 2005. Using semantic similarity metrics to uncover category and land cover change. *Pages 107–119 of: Proceedings of the 1st International Conference on GeoSpatial Semantics (GeoS'05), November 29–30, 2005, Mexico City, Mexico*. Lecture Notes in Computer Science, vol. 3799. Springer, Berlin, Germany.

Ai, T., & van Oosterom, P. J. M. 2002. GAP-tree extensions based on skeletons. *Pages 501–513 of: Proceedings of the 10th International Symposium on Spatial Data Handling (SDH'02), July 9–12, 2002, Ottawa, Canada*.

Aichholzer, O., Aurenhammer, F., Alberts, D., & Gärtner, B. 1995. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, **1**(12), 752–761.

Allouche, M. K., & Moulin, B. 2005. Amalgamation in cartographic generalization using Kohonen's feature nets. *International Journal of Geographical Information Science*, **19**(8–9), 899–914.

Anders, K.-H. 2003. A hierarchical graph-clustering approach to find groups of objects. *In: Proceedings of the 5th Workshop on Progress in Automated Map Generalization, ICA Commission on Map Generalization, April 28–30, 2003, Paris, France*.

Anders, K.-H. 2004. *Parameterfreies hierarchisches Graph-Clustering Verfahren zur Interpretation raumbezogener Daten*. Dissertation, Universität Stuttgart, Germany.

Anders, K.-H., & Bobrich, J. 2004. MRDB approach for automatic incremental update. *In: Proceedings of the 7th ICA Workshop on Generalisation and Multiple Representation, August 20–21, 2004, Leicester, UK*.

Anders, K.-H., & Haunert, J.-H. 2004. MRDB to connect 3D city models, cadastral, and topographic data together. *In: Proceedings of the 24th Urban Data Management Symposium (UDMS'04), October 27–29, 2004, Chioggia (Venice), Italy*.

ARD. 2006. *Bundeslandkalkulator*. `http://www.tagesschau.de/bundeslandkalkulator` (2006/07/18).

Bader, M. 2001. *Energy Minimization Methods for Feature Displacement in Map Generalization*. Ph.D. thesis, Universität Zürich, Switzerland.

Bader, M., & Weibel, R. 1997. Detecting and resolving size and proximity conflicts in the generalization of polygonal maps. *Pages 1525–1532 of: Proceedings of the 18th International Cartographic Conference (ICC'97), June 23–27, 1997, Stockholm, Sweden*.

Bard, S. 2004. Quality assessment of cartographic generalization. *Transactions in GIS*, **8**(1), 63–81.

Barrault, M., Regnauld, N., Duchêne, C., Haire, K., Baeijs, C., Demazeau, Y., Hardy, P., Mackaness, W., Ruas, A., & Weibel, R. 2001. Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalisation. *Pages 2110–2116 of: Proceedings of the 20th International Cartographic Conference (ICC'01), August 6–10, 2001, Beijing, China*, vol. 3.

Beard, M. K. 1991. Constraints on rule formulation. *Chap. 7, pages 121–149 of:* Buttenfield, B. P., & McMaster, R. B. (eds), *Map Generalization: Making Rules for Knowledge Representation*. Longman Scientific & Technical, Harlow, UK.

Bland, R. G. 1977. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, **2**(2), 103–107.

Bobrich, J. 1996. *Ein neuer Ansatz zur kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells*. Dissertation, Technische Universität Darmstadt, Deutsche Geodätische Kommission, Reihe C, vol. 455, Munich, Germany.

Bobzien, M. 2001. Flächenzusammenfassungen in der Modellgeneralisierung. *Pages 19–29 of: Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 20: Arbeitsgruppe Automation in der Kartographie – Tagung 2000*. Verlag des Bundesamtes für Kartographie und Geodäsie, Frankfurt am Main, Germany.

Boffet, A., & Serra, S. R. 2001. Identification of spatial structures within urban blocks for town characterisation. *Pages 1974–1983 of: Proceedings of the 20th International Cartographic Conference (ICC'01), August 6–10, 2001, Beijing, China*, vol. 3.

Bose, P., Cabello, S., Cheong, O., Gudmundsson, J., van Krefeld, M., & Speckmann, B. 2006. Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, **4**(4), 554–556.

Brassel, K. E., & Weibel, R. 1988. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems*, **2**(3), 229–244.

Burghardt, D. 2005. Controlled line smoothing by snakes. *GeoInformatica*, **9**, 237–252.

Buttenfield, B. P., & McMaster, R. B. 1991. *Map Generalization: Making Rules for Knowledge Representation*. Longman Scientific & Technical, Harlow, UK.

Campbell, G. M., & Cromley, R. G. 1991. Optimal simplification of cartographic lines using shortest-path formulations. *The Journal of the Operational Research Society*, **42**(9), 793–802.

Caro, F., Shirabe, T., Guignard, M., & Weintraub, A. 2004. School redistricting: embedding GIS tools with integer programming. *Journal of the Operational Research Society*, **55**(8), 836–849.

Cecconi, A. 2003. *Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping*. Ph.D. thesis, Universität Zürich, Switzerland.

Chan, W. S., & Chin, F. Y. 1996. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry and Applications*, **6**(1), 59–77.

Chen, D., Daescu, O., Hershberger, J., Kogge, P., Mi, N., & Snoeyink, J. 2005. Polygonal path simplification with angle constraints. *Computational Geometry*, **32**(3), 173–187.

Cheng, T., & Li, Z. 2006. Toward quantitative measures for the semantic quality of polygon generalization. *Cartographica*, **41**, 135–147.

Chin, F. Y., Snoeyink, J., & Wang, C. A. 1995. Finding the medial axis of a simple polygon in linear time. *Pages 382–391 of: Proceedings of the 6th International Symposium on Algorithms and Computation (ISAAC'95), December 4–5, 1995, Cairns, Australia*. Lecture Notes in Computer Science, vol. 1004. Springer, Berlin, Germany.

Chithambaram, R., Beard, M. K., & Barrera, R. 1991. Skeletonizing polygons for map generalization. *Pages 44–55 of: Auto-Carto 10: Technical papers of the 1991 ACSM-ASPRS Annual Convention, March 25–28, 1991, Baltimore, MD, USA*, vol. 2.

Cloonan, J. B. 1972. A note on the compactness of sales territories. *Management Science*, **19**(4), 469–470.

Cobb, M. A., Chung, M. J., Foley, H., Petry, F. E., Shaw, K. B., & Miller, H. V. 1998. A rule-based approach for the conflation of attributed vector data. *GeoInformatica*, **2**(1), 7–35.

Cook, S. A. 1971. The complexity of theorem-proving procedures. *Pages 151–158 of: Proceedings of the 3rd annual ACM symposium on theory of computing, May 3–5, 1971, Shaker Heights, OH, USA.*

Cromley, R. G. 1986. A spatial allocation analysis of the point annotation problem. *Pages 38–49 of: Proceedings of the 2nd International Symposium on Spatial Data Handling (SDH'86), July 6–10, 1986, Seattle, WA, USA.*

Dantzig, G. B. 1963. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, USA.

de Berg, M., van Kreveld, M., & Schirra, S. 1998. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems*, **25**(4), 243–257.

de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. 2008. *Computational Geometry: Algorithms and Applications.* Springer, Berlin, Germany.

Deveau, T. 1985. Reducing the number of points in a plane curve representation. *Pages 152–160 of: Proceedings of Auto-Carto VII, March 11–14, 1985, Washington D.C., USA.*

Devogele, T., Trevisan, J., & Raynal, L. 1996. Building a multi-scale database with scale-transition relationships. *Pages 19–33 of: Proceedings of the 7th International Symposium on Spatial Data Handling (SDH'96), August 12–16 , 1996, Delft, The Netherlands*, vol. I:6.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Douglas, D. H., & Peucker, T. K. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its character. *The Canadian Cartographer*, **10**(2), 112–122.

Dumitrescu, I., & Boland, N. 2001. Algorithms for the weight constrained shortest path problem. *International Transactions in Operational Research*, **8**(1), 15–29.

Dunkars, M. 2004. *Multiple Representation Databases for Topographic Information.* Ph.D. thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.

Egenhofer, M. J. 1989. A formal definition of binary topological relationships. *Pages 457–472 of: Proceedings of the 3rd International Conference on Foundations of Data Organization and Algorithms (FODO'89), June 21–23, 1989 Paris, France.* Lecture Notes in Computer Science, vol. 367. Springer, Berlin, Germany.

Eppstein, D., & Erickson, J. 1998. Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions. *Pages 58–67 of: Proceedings of the 14th Annual Symposium on Computational Geometry (SCG'98), June 7–10, 1998, Minneapolis, MN, USA.*

Estkowski, R., & Mitchell, J. S. B. 2001. Simplifying a polygonal subdivision while keeping it simple. *Pages 40–49 of: Proceedings of the 17th Annual Symposium on Computational Geometry (SCG'01), June 3–5, 2001, Medford, MA, USA.*

Felkel, P., & Obdržálek, Š. 1998. Straight skeleton implementation. *Pages 210–218 of: Proceedings of Spring Conference on Computer Graphics, April 22–24, 2004, Budmerice, Slovakia.*

Fischer, D. T., & Church, R. L. 2003. Clustering and compactness in reserve site selection: an extension of the biodiversity management area selection model. *Forest Science*, **49**(4), 555–565.

Frank, A. U., Volta, G. S., & McGranaghan, M. 1997. Formalization of families of categorical coverages. *International Journal of Geographical Information Science*, **11**(3), 215–231.

Frank, R., & Ester, M. 2006. A quantitative similarity measure for maps. *Pages 435–450 of:* Riedl, A., Kainz, W., & Elmes, G. A. (eds), *Progress in Spatial Data Handling: Proceedings of the 12th International Symposium on Spatial Data Handling (SDH'06), July 12–14, 2006, Vienna, Austria.* Springer, Berlin, Germany.

Gaffuri, J. 2007. Outflow preservation of the hydrographic network on the relief in map generalisation. *In: Proceedings of the 23rd International Cartographic Conference (ICC'07), August 4–10, 2007, Moscow, Russia.*

Galanda, M. 2003. *Automated Polygon Generalization in a Multi Agent System.* Ph.D. thesis, Universität Zürich, Switzerland.

Garey, M. R., & Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman & Co. New York, NY, USA.

Garey, M. R., Johnson, D. S., & Stockmeyer, L. 1974. Some simplified NP-complete problems. *Pages 47–63 of: Proceedings of the 6th annual ACM symposium on theory of computing, April 30 – May 02, 1974, Seattle, WA, USA.*

Geoscience Australia. 2006. *Topographic data and map specifications.* `http://www.ga.gov.au/mapspecs/250k100k/` (2007/11/09).

Gomory, R. E. 1958. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, **64**(5), 275–278.

Gonzalez, R. C., & Woods, R. E. 2002. *Digital Image Processing.* 2nd edn. Addison-Wesley, Boston, MA, USA.

Grünreich, D. 2003. The decentralised approach to mapping of the Federal Republic of Germany – a model for establishing the European Spatial Data Infrastructure (ESDI)? *In: Proceedings of Cambridge Conference, July 21–25, 2003, Cambridge, UK.*

Gudmundsson, J., Narasimhan, G., & Smid, M. H. M. 2007. Distance-preserving approximations of polygonal paths. *Computational Geometry*, **36**(3), 183–196.

Haala, N., & Brenner, C. 1997. Generation of 3D city models from airborne laser scanning data. *In: Proceedings of the 3rd EARSeL Workshop on Lidar Remote Sensing of Land and Sea, July 17–19, 1997, Tallinn, Estonia.*

Hajek, B. 1988. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, **13**(2), 311–329.

Hake, G., Grünreich, D., & Meng, L. 1994. *Kartographie. Visualisierung raum-zeitlicher Informationen.* Walter de Gruyter & Co, Berlin, Germany.

Hampe, M. 2007. *Integration einer multiskaligen Datenbank in eine Webservice-Architektur.* Dissertation, Leibniz Universität Hannover, Deutsche Geodätische Kommission, Reihe C, vol. 605, Munich, Germany.

Hampe, M., Anders, K.-H., & Sester, M. 2003. MRDB applications for data revision and real-time generalisation. *Pages 192–202 of: Proceedings of the 21st International Cartographic Conference (ICC'03), August 10–16, 2003, Durban, South Africa.*

Hampe, M., Sester, M., & Harrie, L. 2004. Multiple representation to support visualisation on mobile devices. *Pages 135–140 of: Proceedings of the XXth ISPRS Congress, July 12–23, 2004, Istanbul, Turkey.* International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXV.

Hardy, P. J., Monnot, J.-L., & Lee, D. 2007. An optimization approach to constraint based generalization in a commodity GIS framework. *In: Proceedings of the 23rd International Cartographic Conference (ICC'07), August 4–10, 2007, Moscow, Russia.*

Harrie, L. 1999. The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and Geographic Information Science*, **26**(1), 55–69.

Harrie, L., & Hellström, A.-K. 1999. A prototype system for propagating updates between cartographic data sets. *The Cartographic Journal*, **36**(2), 133–140.

Harrie, L., & Sarjakoski, T. 2002. Simultaneous graphic generalization of vector data sets. *GeoInformatica*, **6**(3), 233–261.

Harrie, L., & Weibel, R. 2007. Modelling the overall process of map generalization. *Chap. 4, pages 67–88 of:* Mackaness, W., Ruas, A., & Sarjakoski, T. L. (eds), *Generalisation of geographic information: Cartographic modelling and applications.* Elsevier, Oxford, UK.

Håstad, J. 2001. Some optimal inapproximability results. *Journal of the ACM*, **48**(4), 798–859.

Haunert, J.-H. 2005a. Geometrietypwechsel in einer Multi-Resolution-Datenbank. *In: Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 34: Arbeitsgruppe Automation in der Kartographie – Tagung 2004.* Verlag des Bundesamtes für Kartographie und Geodäsie, Frankfurt am Main, Germany.

Haunert, J.-H. 2005b. Link based conflation of geographic datasets. *In: Proceedings of the 8th ICA Workshop on Generalisation and Multiple Representation, July 7–8, 2005, A Coruña, Spain.*

Haunert, J.-H. 2006. Aktualisierung von Geodaten in einer Multiple Representation Database. *In: Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., DGPF Jahrestagung 2006, Berlin, Germany.*

Haunert, J.-H. 2007a. Efficient area aggregation by combination of different techniques. *In: Proceedings of the 10th ICA Workshop on Generalisation and Multiple Representation, August 2–3, 2007, Moscow, Russia.*

Haunert, J.-H. 2007b. A formal model and mixed-integer program for area aggregation in map generalization. *Pages 161–166 of: Photogrammetric Image Analysis (PIA'07), September 19–21, 2007, Munich, Germany.* International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXVI(Part 3/W49A). ISPRS.

Haunert, J.-H. 2007c. Optimization methods for area aggregation in land cover maps. *In: Proceedings of the 23rd International Cartographic Conference (ICC'07), August 4–10, 2007, Moscow, Russia.*

Haunert, J.-H., & Sester, M. 2004. Using the straight skeleton for generalisation in a multiple representation environment. *In: Proceedings of the 7th ICA Workshop on Generalisation and Multiple Representation, August 20–21, 2004, Leicester, UK.*

Haunert, J.-H., & Sester, M. 2005. Propagating updates between linked data sets of different scale. *In: Proceedings of the 22nd International Cartographic Conference (ICC'05), July 11–16, 2005, A Coruña, Spain.*

Haunert, J.-H., & Sester, M. 2008a. Area collapse and road centerlines based on straight skeletons. *GeoInformatica*, **12**(2), 169–191.

Haunert, J.-H., & Sester, M. 2008b. Assuring logical consistency and semantic accuracy in map generalization. *Photogrammetrie - Fernerkundung - Geoinformation (PFG)*, **2008**(3), 165–173.

Haunert, J.-H., & Wolff, A. 2006. Generalization of land cover maps by mixed integer programming. *Pages 75–82 of: Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems (GIS'06), November 11–16, 2006, Arlington, VA, USA.*

Haunert, J.-H., & Wolff, A. 2008. Optimal simplification of building ground plans. *In: Proceedings of the XXIst ISPRS Congress, July 3–11, 2008, Beijing, China.*

Haunert, J.-H., Anders, K.-H., & Sester, M. 2006. Hierarchical structures for rule-based incremental generalisation. *Pages 49–57 of: Proceedings of the ISPRS Workshop on Multiple Representation and Interoperability of Spatial Data, February 22–24, 2006, Hannover, Germany.* International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXVI(Part 2/W40).

Haunert, J.-H., Dilo, A., & van Oosterom, P. 2008. *Constrained set-up of the tGAP structure for progressive vector data transfer.* Under review at *Computers & Geosciences.*

Heinzle, F., & Anders, K.-H. 2007. Characterising space via pattern recognition techniques: Identifying patterns in road networks. *Chap. 12, pages 233–254 of:* Mackaness, W., Ruas, A., & Sarjakoski, T. L. (eds), *Generalisation of geographic information: Cartographic modelling and applications.* Elsevier, Oxford, UK.

Heinzle, F., Anders, K.-H., & Sester, M. 2006. Pattern recognition in road networks on the example of circular road detection. *Pages 153–167 of: Proceedings of the 4th International Conference on Geographic Information Science (GIScience'06), September 20–23, 2006, Münster, Germany.* Lecture Notes in Computer Science, vol. 4197. Springer, Berlin, Germany.

Hess, S. W., & Samuels, S. A. 1971. Experiences with a sales districting model: Criteria and implementation. *Management Science*, **18**(4, Part II), 41–54.

Hoffman, W., & Pavley, R. 1959. A method for the solution of the Nth best path problem. *Journal of the ACM*, **6**(4), 506–514.

Hojati, M. 1999. Optimal political districting. *Computers & Operations Research*, **23**(12), 1147–1161.

ILOG. 2008. *ILOG CPLEX: High-performance software for mathematical programming and optimization.* `http://www.ilog.com/products/cplex/` (2008/01/15).

Jaakkola, O. 1997a. Automatic iterative generalization for land cover data. *Pages 277–286 of: AutoCarto 13: Proceedings of the 13th International Symposium on Computer-Assisted Cartography, April 5–10, 1997, Seattle, WA, USA.*

Jaakkola, O. 1997b. Multi-scale land cover databases by automatic generalization. *Pages 709–716 of: Proceedings of the 18th International Cartographic Conference (ICC'97), June 23–27, 1997, Stockholm, Sweden.*

Jaakkola, O. 1997c. *Quality and Automatic Generalization of Land Cover Data.* Ph.D. thesis, Department of Geography, University of Helsinki, Finland.

Jaakkola, O. 1998. Multi-scale categorical data bases with automatic generalization transformations based on map algebra. *Cartography and Geographic Information Systems*, **25**(4), 195–207.

Jansen, M., & van Kreveld, M. 1998. Evaluating the consistency of cartographic generalization. *Pages 668–678 of: Proceedings of the 8th International Symposium on Spatial Data Handling (SDH'98), July 11–15, 1998, Vancouver, Canada.*

Kada, M., & Luo, F. 2006. Generalisation of building ground plans using half-spaces. *In: Proceedings of the International Symposium on Geospatial Databases for Sustainable Development, September 25–30, 2006, Goa, India.* International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXVI(Part 4). ISPRS Technical Commission IV.

Kainz, W. 1995. Logical consistency. *Chap. 6, pages 109–137 of:* Guptill, S. C., & Morrison, J. L. (eds), *Elements of spatial data quality.* Elsevier Science, Oxford, UK.

Karmarkar, N. 1984. A new polynomial time algorithm for linear programming. *Combinatorica*, **4**(4), 373–395.

Karp, R. M. 1972. Reducibility among combinatorial problems. *Pages 85–103 of:* Miller, R. E., & Thatcher, J. W. (eds), *Complexity of Computer Computations.* Plenum Press, New York, NY, USA.

Kelmelis, J. 2003. To The National Map and beyond. *Cartography and Geographic Information Science*, **30**(2), 185–198.

Kilpeläinen, T. 2000. Maintenance of multiple representation databases for topographic data. *The Cartographic Journal*, **37**, 101–107.

Kilpeläinen, T., & Sarjakoski, T. 1995. Incremental generalization for multiple representations of geographic objects. *Pages 209–218 of:* Müller, J.-C., Lagrange, J.-P., & Weibel, R. (eds), *GIS and Generalization - Methodology and Practice.* GISDATA, no. 1. Taylor & Francis, London, UK.

Kirkpatrick, S., Jr., C. D. Gelatt, & Vecchi, M. P. 1983. Optimization by simulated annealing. *Science*, **220**(4598), 671–680.

Koch, A. 2007. *Semantische Integration von zweidimensionalen GIS-Daten und Digitalen Geländemodellen.* Dissertation, Universität Hannover, Deutsche Geodätische Kommission, Reihe C, vol. 601, Munich, Germany.

Koch, A., & Heipke, C. 2005. Semantically correct 2.5D GIS data – the integration of a DTM and topographic vector data. *Pages 509–526 of:* Fisher, P. F. (ed), *Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial Data Handling (SDH'04), August 23–25, 2004, Leicester, UK.* Springer, Berlin, Germany.

Lazimy, R. 1982. Mixed-integer quadratic programming. *Mathematical Programming*, **22**(1), 332–349.

Li, Z. 2007. *Algorithmic Foundation of Multi-Scale Spatial Representation.* CRC Press, Taylor & Francis Group, Boca Raton, Florida, USA.

LPSolve. 2008. *lp_solve reference guide*. `http://lpsolve.sourceforge.net/5.5/` (2008/01/15).

Lüscher, P., Burghardt, D., & Weibel, R. 2007. Ontology-driven enrichment of spatial databases. *In: Proceedings of the 10th ICA Workshop on Generalisation and Multiple Representation, August 2–3, 2007, Moscow, Russia*.

MacEachren, A. M. 1985. Compactness of geographic shape: Comparison and evaluation of measures. *Geografiska Annaler. Series B, Human Geography*, **67**(1), 53–67.

Mackaness, W. A., & Purves, R. S. 2001. Automated displacement for large numbers of discrete map objects. *Algorithmica*, **30**, 302–311.

Mackaness, W. A., Ruas, A., & Sarjakoski, L. T. 2007. *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier, Oxford, UK.

Mantel, D., & Lipeck, U. W. 2004. Matching cartographic objects in spatial databases. *Pages 172–176 of: Proceedings of the XXth ISPRS Congress, July 12–23, 2004, Istanbul, Turkey*. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXV(Part B4).

Michalewicz, Z., & Fogel, D. B. 2004. *How to Solve It, Modern Heuristics*. Springer, Berlin, Germany.

Michiels, W., Aarts, E., & Korst, J. 2007. *Theoretical Aspects of Local Search*. Monographs in Theoretical Computer Science. Springer, Berlin, Germany.

Mitchell, J. E. 2002. Branch-and-cut algorithms for combinatorial optimization problems. *Pages 65–77 of:* Pardalos, P. M., & Resende, M. G. C. (eds), *Handbook of Applied Optimization*. Oxford University Press, Oxford, UK.

Müller, J. C., Lagrange, J. P., & Weibel, R. 1995. *GIS and Generalisation*. GISDATA, no. 1. Taylor & Francis, London, UK.

Monnot, J.-L., Hardy, P., & Lee, D. 2007. Topological constraints, actions and reflexes for generalization by optimization. *In: Proceedings of the 10th ICA Workshop on Generalisation and Multiple Representation, August 2–3, 2007, Moscow, Russia*.

Morrison, J. L. 1995. Spatial data quality. *Chap. 1, pages 1–12 of:* Guptill, S. C., & Morrison, J. L. (eds), *Elements of Spatial Data Quality*. Elsevier Science, Oxford, UK.

Mustière, S., & Devogele, T. 2008. Matching networks with different levels of detail. *GeoInformatica*, **12**(4), 435–453.

Natural Resources Canada. 1996. *National topographic data base: data dictionary*. `http://ftp2.cits.rncan.gc.cs/pub/bndt/doc/dictntd3_en.pdf` (2007/11/09).

Neun, M., Weibel, R., & Burghardt, D. 2004. Data enrichment for adaptive generalisation. *In: Proceedings of the 7th ICA Workshop on Generalisation and Multiple Representation, August 20–21, 2004, Leicester, UK*.

Nickerson, B. G., & Freeman, H. 1986. Development of a rule-based system for automatic map generalization. *Pages 537–556 of: Proceedings of the 2nd International Symposium on Spatial Data Handling (SDH'86), July 5–10, 1986, Seattle, WA, USA*.

Nöllenburg, M., & Wolff, A. 2006. A mixed-integer program for drawing high-quality metro maps. *Pages 321–333 of:* Healy, P., & Nikolov, N. S. (eds), *Proceedings of the 13th International Symposium on Graph Drawing (GD'05), September 12–14, Limerick, Ireland*. Lecture Notes in Computer Science, vol. 3843. Springer, Berlin, Germany.

Papadimitriou, C. H., & Steiglitz, K. 1998. *Combinatorial Optimization*. Dover Publications, Inc., Mineola, NY, USA.

Penninga, F., Verbree, E., Quak, W., & van Oesterom, P. 2005. Construction of the planar partition postal code map based on cadastral registration. *GeoInformatica*, **9**(2), 181–204.

Perkal, J. 1966. An attempt at objective generalization. *In:* Nystuen, J. D. (ed), *Michigan Inter-University Community of Mathematical Geographers, Discussion Paper 10*. Ann Arbor, MI, USA: Department of Geography, University of Michigan.

Peter, B., & Weibel, R. 1999. Using vector and raster-based techniques in categorical map generalization. *In: Proceedings of the 3rd ICA Workshop on Progress in Automated Map Generalization, August 12–14, 1999, Ottawa, Canada.*

Podolskaya, E. S., Anders, K.-H., Haunert, J.-H., & Sester, M. 2007. Quality assessment for polygon generalization. *In: Proceedings of the 5th International Symposium on Spatial Data Quality, June 13–15, 2007, Enschede, The Netherlands.*

Podrenek, M. 2002. *Aufbau des DLM50 aus dem Basis-DLM und Ableitung der DTK50 – Lösungsansatz in Niedersachsen.* Kartographie als Baustein moderner Kommunikation, Kartographische Schriften, Band 6, 126–130, Kirschbaum Verlag, Bonn, Germany.

Reeves, C. R. 1993. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Press, Oxford, UK.

Regnauld, N. 1996. Recognition of building clusters for generalization. *Pages 1–14 of: Proceedings of the 7th International Symposium on Spatial Data Handling (SDH'96), August 12–16 , 1996, Delft, The Netherlands*, vol. I:4B.

Revell, P. 2007. Generic tools for generalising ordnance survey base scale landcover data. *In: Proceedings of the 10th ICA Workshop on Generalisation and Multiple Representation, August 2–3, 2007, Moscow, Russia.*

Rieger, M. K., & Coulsen, M. R. C. 1993. Consensus or confusion: cartographers' knowledge of generalization. *Cartographica*, **30**(2 & 3), 69–80.

Roberts, S. A., Hall, G. B., & Boots, B. 2005. Street centerline generation with an approximated area Voronoi diagram. *Pages 435–446 of:* Fisher, P. F. (ed), *Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial Data Handling (SDH'04), August 23–25, 2004, Leicester, UK.* Springer, Berlin, Germany.

Rodríguez, M. A., & Egenhofer, M. J. 2004. Comparing geospatial entity classes: an asymmetric and context dependent similarity measure. *International Journal of Geographical Information Science*, **18**(3), 229–256.

Ruas, A. 1999. The role of meso level for urban generalisation. *In: Proceedings of the 3rd ICA Workshop on Progress in Automated Map Generalization, August 12–14, 1999, Ottawa, Canada.*

Salgé, F. 1995. Semantic accuracy. *Chap. 7, pages 139–151 of:* Guptill, S. C., & Morrison, J. L. (eds), *Elements of spatial data quality.* Elsevier Science, Oxford, UK.

Sarjakoski, T., & Kilpeläinen, T. 1999. Holistic cartographic generalization by least squares adjustment for large data sets. *In: Proceedings of the 19th International Cartographic Conference (ICC'99), August 14–21, 1999, Ottawa, Canada.*

Schröder, M. 2001. *Gebiete optimal aufteilen.* Dissertation, Universität Karlsruhe, Germany.

Schwering, A. 2008. Approaches to Semantic Similarity Measurement for Geo-Spatial Data: A Survey. *Transactions in GIS*, **12**(1), 5–29.

Schylberg, L. 1993. *Computational Methods for Generalization of Cartographic Data in a Raster Environment.* Ph.D. thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.

Sester, M. 2005. Optimization approaches for generalization and data abstraction. *International Journal of Geographical Information Science*, **19**(8–9), 871–897.

Sester, M., Anders, K.-H., & Walter, V. 1998. Linking objects of different spatial data sets by integration and aggregation. *GeoInformatica*, **2**(4), 335–358.

Shea, K. S. 1991. Design considerations for an artificially intelligent system. *Chap. 1, pages 121–149 of:* Buttenfield, B. P., & McMaster, R. B. (eds), *Map Generalization: Making Rules for Knowledge Representation.* Longman Scientific & Technical, Harlow, UK.

Sheth, A. P., & Larson, J. A. 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, **22**(3), 183–236.

Shirabe, T. 2003. *Decomposing Integer Programming Models for Spatial Allocation*. Ph.D. thesis, University of Pennsylvania, USA.

Shirabe, T. 2005a. Classification of spatial properties for spatial allocation modeling. *GeoInformatica*, **9**(3), 269–287.

Shirabe, T. 2005b. A model of contiguity for spatial unit allocation. *Geographical Analysis*, **37**, 2–16.

Skogan, D., & Skagestein, G. 2005. An automated incremental generalization method. *In: Multi-Resolution Geographic Data and Consistency (Dissertation by David Skogan)*. University of Oslo, Norway.

Spiess, E. 1983. Revision of topographic maps: photogrammetric and cartographic methods of the Fribourg test. *The Photogrammetric Record*, **11**(61), 29–42.

Staufenbiel, W. 1973. *Zur Automation der Generalisierung topographischer Karten mit besonderer Berücksichtigung großmaßstäbiger Gebäudedarstellungen*. Ph.D. thesis, Technische Universität Hannover, Deutsche Geodätische Kommission, Reihe C, vol. 51, Munich, Germany.

Steiniger, S., & Weibel, R. 2007. Relations among map objects in cartographic generalization. *Cartography and Geographic Information Science*, **34**(3), 175–197.

Su, B., Li, Z., & Lodwick, G. 1998. Morphological models for the collapse of area features in digital map generalization. *GeoInformatica*, **2**(4), 359–382.

Tănase, M. 2005. *Shape Decomposition and Retrieval*. Ph.D. thesis, Universiteit Utrecht, The Netherlands.

Tănase, M., & Veltkamp, R. C. 2003. Polygon decomposition based on the straight line skeleton. *Pages 221–244 of: Geometry, Morphology, and Computational Imaging*. Lecture Notes in Computer Science, vol. 2616. Springer, Berlin, Germany.

Tănase, M., & Veltkamp, R. C. 2004. A straight skeleton approximating the medial axis. *Pages 809–821 of: Proceedings of the 12th Annual European Symposium on Algorithms (ESA'04), September 14–17, 2004, Bergen, Norway*. Lecture Notes in Computer Science, vol. 3221. Springer, Berlin, Germany.

Tavares-Pereira, F., Figueira, J. Rui, Mousseau, V., & Roy, Bernard. 2007. Multiple criteria districting problems: The public transportation network pricing system of the Paris region. *Annals of Operations Research*, **154**, 69–92.

Thomson, R. C., & Brooks, R. 2002. Exploiting perceptual grouping for map analysis, understanding and generalization: The case of road and river networks. *Pages 148–157 of: Graphics Recognition. Algorithms and Applications: Proceedings of the 4th International Workshop, GREC 2201, September 7–8, 2001, Kingston, Ontario, Canada.* Lecture Notes in Computer Science, vol. 2390. Springer, Berlin, Germany.

Thomson, R. C., & Richardson, D. E. 1995. A graph theory approach to road network generalization. *Pages 1871–1880 of: Proceedings of the 17th International Cartographic Conference (ICC'95), September 3–9, 1995, Barcelona, Spain*.

Timpf, S. 1998. *Hierarchical Structures in Map Series*. Ph.D. thesis, Technical University Vienna, Austria.

Töpfer, F., & Pillewizer, W. 1966. The principles of selection. *The Cartographic Journal*, **3**(1), 10–16.

van Kreveld, M., & Peschier, J. 1998. On the automated generalization of road network maps. *In: Proceedings of the 3rd International Conference on GeoComputation, September 17–19, 1998, Bristol, UK*.

van Oosterom, P. J. M. 1995. The GAP-tree, an approach to 'on-the-fly' map generalization of an area partitioning. *In:* Müller, J.-C., Lagrange, J.-P., & Weibel, R. (eds), *GIS and Generalization - Methodology and Practice*. GISDATA, no. 1. Taylor & Francis, London, UK.

van Oosterom, P. J. M. 2005. Variable-scale topological data structures suitable for progressive data transfer: The GAP-face tree and GAP-edge forest. *Cartography and Geographic Information Science*, **32**(4), 331–346.

van Smaalen, J. W. N. 1996. A hierarchic rule model for geographic information abstraction. *Pages 31–41 of: Proceedings of the 7th International Symposium on Spatial Data Handling (SDH'96), August 12–16, 1996, Delft, The Netherlands*, vol. I:4B.

van Smaalen, J. W. N. 2003. *Automated Aggregation of Geographic Objects*. Ph.D. thesis, Wageningen University, The Netherlands.

Walsh, S. J., Crews-Meyer, K. A., Crawford, T. W., & Welsh, W. F. 2004. Population and environment interactions: Spatial considerations in landscape characterization and modeling. *Chap. 2, pages 41–65 of:* Sheppard, E., & McMaster, R. B. (eds), *Scale and Geographic Inquiry: Nature, Society, and Method*. Blackwell Publishing, Malden, MA, USA.

Walter, V., & Fritsch, D. 1999. Matching spatial data sets: A statistical approach. *International Journal of Geographical Information Science*, **13**(5), 445–473.

Ware, J. M., & Jones, C. B. 1998. Conflict reduction in map generalization using iterative improvement. *GeoInformatica*, **2**(4), 383–407.

Ware, J. M., Jones, C. B., & Bundy, G. L. 1995. A triangulated spatial model for cartographic generalisation of areal objects. *Pages 173–192 of: Proceedings of the International Conference COSIT '95, Spatial Information Theory, a Theoretical Basis for GIS, September 21–23, 1995, Semmering, Austria*. Lecture Notes in Computer Science, vol. 988. Springer, Berlin, Germany.

Ware, J. M., Wilson, I. D., Ware, J. A., & Jones, C. B. 2002. A tabu search approach to automated map generalisation. *Pages 101–106 of: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems (GIS'02), November 8–9, 2002, McLean, VA, USA*.

Ware, J. M., Jones, C. B., & Thomas, N. 2003. Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographical Information Science*, **17**(8), 743–769.

Weibel, R. 1997. Generalization of spatial data: Principles and selected algorithms. *Pages 99–152 of:* van Kreveld, M., Nievergelt, J., Roos, T., & Widmayer, P. (eds), *Algorithmic Foundations of Geographic Information Systems*. Lecture notes in computer science, vol. 1340. Springer, Berlin, Germany.

Weibel, R., & Dutton, G. 1998. Constraint-based automated map generalization. *Pages 214–224 of: Proceedings of the 8th International Symposium on Spatial Data Handling (SDH'98), July 11–15, 1998, Vancouver, Canada*.

Weibel, R., Keller, S., & Reichenbacher, T. 1995. Overcoming the knowledge acquisition bottleneck in map generalization: The role of interactive systems and computational intelligence. *Pages 139–156 of: Proceedings of the International Conference COSIT '95, Spatial Information Theory, a Theoretical Basis for GIS, September 21–23, 1995, Semmering, Austria*. Lecture Notes in Computer Science, vol. 988. Springer, Berlin, Germany.

Wertheimer, M. 1938. Laws of organization in percetional forms. *Pages 71–88 of:* Ellis, W. (ed), *A source book of Gestalt psychology*. Routledge & Kegan Paul, London, UK. English translation.

Williams, J. C. 2002. A zero-one programming model for contiguous land acquisition. *Geographical Analysis*, **34**(4), 330–349.

Wright, J., ReVelle, C., & Cohon, J. 1983. A multiobjective integer programming model for the land acquisition problem. *Regional Science and Urban Economics*, **13**, 31–53.

Yan, H., Weibel, R., & Yang, B. 2008. A multi-parameter approach to automated building grouping and generalization. *GeoInformatica*, **12**(1), 73–89.

Yaolin, L., Molenaar, M., & Kraak, M.-J. 2002. Semantic similarity evaluation model in categorical database generalization. *In: Proceedings of the ISPRS Commission IV Symposium "Geospatial Theory, Processing and Applications", July 9–12, 2002, Ottawa, Canada*. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, no. XXXIV(Part 4).

Zhang, Q. 2005. Road network generalization based on connection analysis. *Pages 343–353 of:* Fisher, P. F. (ed), *Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial Data Handling (SDH'04), August 23–25, 2004, Leicester, UK*. Springer, Berlin, Germany.

Zoltners, A. A., & Sinha, P. 1983. Sales territory alignment: A review and model. *Management Science*, **29**(11), 1237–1256.

Acknowledgments

I thank Prof. Dr. Monika Sester for her excellent mentoring. Throughout the last four years I could always rely on her support in my research, even when the actual value of a new idea was still uncertain.

It was Dr. Alexander Wolff who opened the door to the world of combinatorial optimization for me – thank you, Sascha!

I also express my sincere thanks to Prof. Dr. Hansjörg Kutterer for his comments and our discussions.

At the beginning of my research work, Dr. Karl-Heinrich Anders was my first contact person for any problems related to my tasks. His advice had a major influence on the direction of my research. Many thanks, Charly!

Many thanks to Patrick Revell, who helped me to improve the English language in this thesis, and as an expert in map generalization, contributed with his very useful comments.

To perform the experimental tests presented in this thesis, I used computing resources of the Faculty of Informatics at University of Karlsruhe. I would like to thank Prof. Dr. Dorothea Wagner for the opportunity.

For the most part, my dissertation presents results of the project "updating of geographic data in a multiple representation database", which was funded by the German Research Foundation (DFG), grant SE 645/2-1. I gratefully acknowledge this support. Without it, this thesis would not have been possible.

The project was part of a larger bundle project entitled "abstraction of geographic information within multiscale acquisition, administration, analysis, and visualization". Thanks to all the project members for our regular discussions!

I very much appreciate to work at the institute of cartography and geoinformatics of Leibniz Universität Hannover and do enjoy to be a member of its innovative and creative research group. Thanks to the whole team for letting me be a part of it.

Finally, I want to thank my family and all my friends who have encouraged and supported me in following my research interests.

Curriculum Vitae

Personnel Details

Name	Jan-Henrik Haunert
Address	Alte Döhrener Straße 27 30173 Hannover Germany
Date and Place of Birth	1978/05/09, Hannover
Citizenship	German

Education

1984/08 – 1988/06	Elementary school in Hannover
1988/07 – 1997/06	Carl-Friedrich-Gauß-Schule in Hemmingen, graduated with high school diploma *(Abitur)*
1998/10 – 2002/08 and 2003/01 – 2003/12	Studies in the degree program 'geodesy and geoinformatics', Leibniz Universität Hannover, graduated with engineering diploma *(Dipl.-Ing.)*
2002/08 – 2003/01	Guest studies in the degree program 'surveying', Helsinki University of Technology, Finland

Research Work

2001/11 – 2002/07	Student assistant at the Institute of Photogrammetry and GeoInformation, Leibniz Universität Hannover
Since 2004/01	Scientific assistant at the Institute of Cartography and Geoinformatics, Leibniz Universität Hannover
2006/08/13 – 2006/08/16	Research guest of Dr. Alexander Wolff, Universität Karlsruhe, Germany
Since 2007/03	Secretary of the ISPRS Working Group II/3 – Multiple Representation of Image and Vector Data
2007/03/04 – 2007/03/08	Research guest of Dr. Alexander Wolff, TU Eindhoven, The Netherlands
2007/06/18 – 2007/07/13	Research guest of Prof. Dr. Peter van Oosterom, TU Delft, The Netherlands

Other Work Experience

1997/07 – 1998/07	Civilian service at the German Red Cross, Hannover
1998/07 – 1998/09	Internship at the engineering company ÖbVI Heubner/Rohardt, Hannover