

RESEARCH ARTICLE

A cutting-plane method for contiguity-constrained spatial aggregation

Johannes Oehrlein and Jan-Henrik Haunert

Institute of Geodesy and Geoinformation, University of Bonn, Germany

Received: September 11, 2016; returned: November 17, 2017; revised: November 26, 2017; accepted: November 28, 2017.

Abstract: Aggregating areas into larger regions is a common problem in spatial planning, geographic information science, and cartography. The aim can be to group administrative areal units into electoral districts or sales territories, in which case the problem is known as districting. In other cases, area aggregation is seen as a generalization or visualization task, which aims to reveal spatial patterns in geographic data. Despite these different motivations, the heart of the problem is the same: given a planar partition, one wants to aggregate several elements of this partition to regions. These often must have or exceed a particular size, be homogeneous with respect to some attribute, contiguous, and geometrically compact. Even simple problem variants are known to be NP-hard, meaning that there is no reasonable hope for an efficient exact algorithm. Nevertheless, the problem has been attacked with heuristic and exact methods. In this article we present a new exact method for area aggregation and compare it with a state-of-the-art method for the same problem. Our method results in a substantial decrease of the running time and, in particular, allowed us to solve certain instances that the existing method could not solve within five days. Both our new method and the existing method use integer linear programming, which allows existing problem solvers to be applied. Other than the existing method, however, our method employs a cutting-plane method, which is an advanced constraint-handling approach. We discuss this approach in detail and present its application to the aggregation of areas in choropleth maps.

Keywords: area aggregation, districting, spatial unit allocation, optimization, integer linear programming, cutting-plane method, map generalization, choropleth map

1 Introduction

Planar subdivisions are frequently used to structure geographic space. In geographic information systems, they can be used as a basis for data acquisition, storage, analysis, and visualization. Since different applications require information on different scales, planar subdivisions are often hierarchical—unemployment rates, for example, can be analyzed on a county or country level. Often, one aims to compute a higher-level subdivision from a given one, by grouping areas of similar attribute values. With such an approach, one can reveal large-scale patterns in the data. In this article, we present a new method for area aggregation, which we discuss in the context of districting and spatial unit allocation. We distinguish these problems as follows:

- *Districting* [9, 36, 38, 62] is the problem of partitioning a set of minimum mapping units (e.g., postal code zones) to form larger regions or districts (e.g., school zones or electoral districts). The minimum mapping units (i.e., input areas) are assumed to form a planar subdivision. The applications of districting range from administrative to commercial purposes; an overview is provided by Shirabe [61].
- *Spatial unit allocation* [60, 61] subsumes districting, but it does not necessarily ask to assign every area to a district. A typical example of spatial unit allocation is to select a set of areas constituting a single region that is geometrically compact and requires minimal development costs [1].
- The term *area aggregation* has been used to refer to the aggregation of areas as a data abstraction or map generalization problem [34]. Just as districting, area aggregation requires a planar subdivision as input and asks to group the areas into larger regions. Districting problems in spatial planning, however, do not necessarily ask to group areas of similar attribute values, which is an essential criterion for generalization.

It is common to approach districting, spatial unit allocation, and area aggregation by optimization [7, 20, 29, 33, 45, 49, 62]. The existing approaches are quite similar, since the different problem variants often share some optimization objectives and constraints. For example, it is common to require that every output region must have a size or population within certain bounds and to favor geometrically compact shapes. Compactness is usually assessed quantitatively, which can be done with different measures [44, 47], and considered as an optimization objective. Additionally, in many problem variants, the output regions are required to be *contiguous* [14, 20, 61, 62, 69]:

Definition 1. *An area $A \subseteq \mathbb{R}^2$ is called contiguous if every two points in A are connected via a (not necessarily straight) line that is contained in A .*

Though the output regions tend to become contiguous when compactness is considered as an objective, there is generally no guarantee for contiguity without enforcing it. In fact, if compactness is not the primary objective, contiguity often has to be enforced to produce somehow reasonable output regions [62]. Therefore, we think that optimizing similarity (and compactness as a secondary criterion) subject to size constraints and contiguity is a particularly interesting challenge.

Attribute similarity is an important criterion for aggregation in *map generalization*, which means to generate a more abstract and less detailed representation of geographic space from a given one [8, 59, 68]. The problem occurs if the scale of a cartographic visualization has to be reduced, but map generalization does not necessarily assume that the input

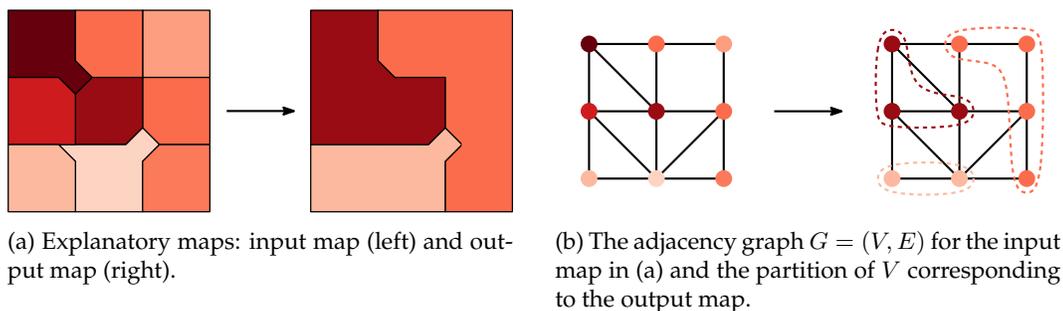


Figure 1: An example of area aggregation (see Haunert and Wolff [34]).

and output representations are visual graphics. Haunert and Wolff [34] have defined the area aggregation problem in map generalization formally and developed an optimization method for it, which is based on models for districting by Zoltners and Sinha [70] and Shirabe [62]. The problem not only requires to group the input areas into larger regions, but also to assign a value from the attribute domain to each output region; see Figure 1(a). The aim is to minimize a cost function that penalizes changes of attribute values to dissimilar values as well as geometrically non-compact output regions, subject to constraints concerning the size and contiguity of the output regions. The method relies on the definition of the adjacency graph $G = (V, E)$ whose vertex set V contains a vertex for each input area and whose edge set E contains an edge $\{u, v\}$ for each two adjacent areas $u, v \in V$; see Figure 1(b). We will use this definition of G throughout this article.

In this article, we revisit the problem defined by Haunert and Wolff [34], but we also consider the special case that similarity is neglected and compactness is the sole objective. In this case, the problem is more similar to a classical districting problem that simply demands geometrically compact and contiguous regions of sizes within certain bounds. Moreover, while Haunert and Wolff developed and tested their method for the generalization of categorical maps, we will use our method to generalize choropleth maps with a ratio-scaled variable, such as unemployment rates. This has the advantage that we can directly compute differences between attribute values and do not depend on the definition of a semantic distance between categories.

We focus on *exact* optimization methods based on *integer linear programming*, which is a common optimization approach for districting [9, 33, 36]. In particular, it is reasonable for NP-hard problems, for which the existence of an efficient and exact algorithm is extremely unlikely [28]. In fact, area aggregation falls into the class of NP-hard problems [34], and so do many problem variants of districting [2, 38, 57].

Defining an *integer linear program* (ILP) means setting up a *linear program* (LP) completed with an *integrality constraint*. While the LP defines a linear objective function and a set of linear inequality constraints over a set of variables, the integrality constraint requires that the variables receive integer values. An ILP is solved optimally if a variable assignment is found which optimizes the objective function without violating any constraint. Commonly, the LP corresponding to an ILP without integrality constraints is referred to as the *LP relaxation* of the ILP [52]. An illustrative example for both an LP and an ILP can be found in Figure 2.

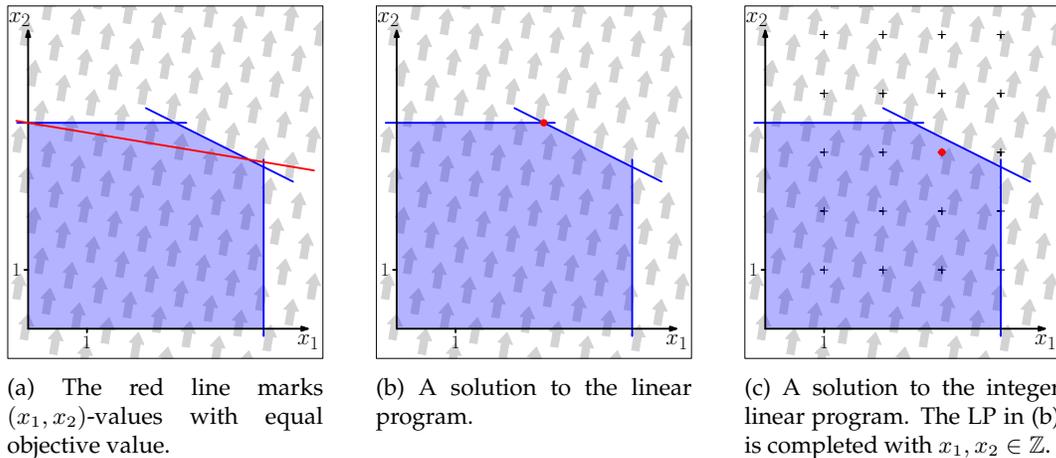


Figure 2: A linear program with variables x_1 and x_2 ; objective is to maximize $6x_1 + x_2$ with respect to $x_1 \geq 0, x_2 \geq 0, 2x_2 \leq 7, 2x_1 + 4x_2 \leq 18$ and $x_1 \leq 4$. The solution space, i.e., all pairs (x_1, x_2) for which every given inequality is true, is marked as a blue polytope. The gray arrows in the background indicate the objective. They are orthogonal to the line of equal objective values in (a).

Whereas efficient algorithms for linear programming exist, integer linear programming is NP-hard [13, 28]. Nevertheless, an approach based on integer linear programming is promising as it allows sophisticated optimization software (e.g., CPLEX [15] and Gurobi [31]) to be applied. Even though the exact methods for solving ILPs have an exponential worst-case running time, they can be relatively fast when applied to real-world instances. Moreover, solutions of an exact method can be used as quality benchmarks to evaluate the results of (faster) heuristic algorithms. Heuristic methods for districting have been developed by several researchers [7, 19, 37, 43, 50]. In contrast to exact methods, these do not guarantee to deliver an optimal solution.

Just as some criteria are shared by many problem variants of districting and spatial unit allocation, the ILP formulations for these problems often share some elementary components. Shirabe [63] has used this fact to integrate mathematical programming techniques and geographic information systems (GIS), such that a GIS user can assemble a model for a particular spatial unit allocation problem from a set of elementary model components and compute a solution to the problem with an ILP solver. Since contiguity is an important requirement in many spatial unit allocation problems, several works have focused on formalizing contiguity as one such elementary model component [61, 62, 69].

Usually, there exist multiple possibilities of encoding a particular problem as an ILP; choosing among these ILP formulations can highly influence the computation time. In geographic information (GI) science, it is common to choose a *compact* ILP formulation, which means that the size of the ILP is polynomial in the size of the input [24]. For example, in Section 2.2 we show that, when applying Shirabe's model [61] to area aggregation without prescribing the number of output regions, it has $\mathcal{O}(n^2)$ variables and constraints (where n is the number of input areas) and thus a polynomial size. A compact ILP formulation can be favorable because it permits a full instantiation of the model, that is, a file or data structure explicitly storing all variables and constraints. After the instantiation of the model, it can

be handed over to a solver, which computes an optimal solution without requiring any further interaction. It is therefore common to think of the solver as a black box [46].

Working with compact ILP formulations is relatively convenient. It is also known, however, that they are sometimes outperformed by *non-compact* ILP formulations, whose number of constraints can be exponential in the size of the input [56]. Such a large set of constraints forbids a full instantiation of the model. Therefore, one starts the computation by working with a reduced ILP, which is lacking a set of constraints of the original ILP. A simple approach is to solve this reduced ILP to optimality and to examine whether the solution violates constraints of the original ILP. If a violation of a constraint is found, that constraint is added to the ILP and the solution process is started anew. Drexler and Haase [18] and Duque et al. [20] use this approach to solve districting problems. In particular, Duque et al. deal with the *p-regions problem*, in which, other than in our problem, the number p of output regions is prescribed.

In this article, we present a more sophisticated approach for area aggregation. We demonstrate the effectiveness of a *cutting-plane method*, which generally refers to a method that adds constraints during optimization without relying on an optimal solution to the reduced ILP. Violated constraints are found already in a preliminary stage of a solution, namely in an optimal solution to the LP relaxation of the reduced ILP. Such constraints are termed *cutting planes* (or simply *cuts*) because they cut away parts of the feasible region of the solution space defined by the current instantiation of the model. The number of constraints of our ILP formulation for area aggregation is exponential in the number n of input areas, but initially we instantiate the model with only $\mathcal{O}(n^2)$ constraints. We generate constraints ensuring contiguity during optimization, using what is generally termed a *separation algorithm* [52]. Though implementing a cutting-plane approach requires some understanding of how an ILP solver works and certainly more effort than using a compact ILP formulation and a black-box solver, we consider it practicable, also for researchers in GI science. This is because modern ILP solvers such as CPLEX or Gurobi offer programming libraries that include interfaces (usually termed *callbacks*) for intervening in the optimization process. Carvajal et al. [10] have developed a cutting-plane method based on an efficient separation algorithm for a problem of spatial unit allocation in forest management. We are not aware of such a method for area aggregation or districting, though.

We used the cutting plane method for contiguity-constrained spatial unit allocation by Carvajal et al. as a starting point, which, compared to the method of Drexler and Haase [18], is more recent and can be considered more sophisticated. However, we had to extend the method substantially for the case of an unknown number of output regions and a flexible set of centers. More precisely, Carvajal et al. consider two constraint formulations for the contiguity of a region, namely one that does not rely on the concept of a center of a region and one that requires a prescribed center to belong to the output region. While in the first model, contiguity is ensured by considering a set of constraints for each pair of nodes that are selected for the output region, in the second model, a set of constraints for each selected node ensures its connectivity to the prescribed center. Conceptually, our approach is more similar to the second model, as it also relies on the idea of centers. However, we do not know in advance which of the nodes become centers. Therefore, we use a constraint formulation that, in fact, is more similar to the first model of Carvajal et al. in the sense that it uses one set of constraints for each pair of nodes.

Álvarez-Miranda et al. [3] and, recently, Wang et al. [66] have presented theoretical findings on integer programming formulations for the problem of selecting a maximum-weight

connected subgraph of a given graph. Though their results cannot easily be transferred to other problems, they can be understood as hints on why a non-compact ILP formulation, such as the one of Carvajal et al. or ours, can outperform a compact ILP formulation.

To summarize our contribution, we discuss cutting-plane methods as a general constraint-handling technique that is rather unknown in GI science but well established in the field of combinatorial optimization [39, 65]. We show that our cutting-plane method outperforms the districting method of Shirabe [62] that was adapted by Haunert and Wolff [34] for the aggregation of areas in map generalization. For example, with our method we were able to solve various instances with 94 departments of France (excluding overseas department and the island of Corsica) in reasonable time, whereas using Shirabe's method produces a result in much longer time or not at all (see Section 5). We specifically apply this to generate a map that shows a structuring of France into a few (e.g., 10) regions of similar unemployment rates and thereby highlight the usefulness of the method for the generalization of choropleth maps. We do note that the applicability of our method is limited, since we were not able to process instances larger than our instances of France. The number of areas in these instances of France, however, can be considered typical for choropleth maps. Similar maps can be found, for example, in Bertin's fundamental textbook on visualization [6].

In the following, we review an existing ILP formulation for area aggregation (Section 2). Then, we give an overview of strategies for handling ILPs with large sets of constraints (Section 3). Subsequently (Section 4), we contribute an ILP applying cutting planes which extends the ILP formulation from Section 2. Afterwards, we let both models compete in a series of experiments (Section 5). We apply both ILP formulations on a real-world example with 94 input areas, discuss the solutions, and compare the running times for different settings. We finish this article with concluding remarks and ideas for further improvements (Section 6).

2 A state-of-the-art model

The huge amount of work on spatial unit allocation and districting disallows a comprehensive review in this article. Therefore, we refer to the survey by Ricca et al. [58] for an overview and discuss only the most relevant related work that has inspired our models. This in particular concerns a general districting model with assignments of areas to centers (Section 2.1) and a flow-based model to ensure contiguous output regions (Section 2.2). In Section 2.3, we briefly review the approach of Haunert and Wolff [34] for the aggregation of areas in map generalization, which is based on the models from Sections 2.1 and 2.2.

2.1 A compact ILP without contiguity

The ILPs that we use in this article differ only with respect to the constraints ensuring contiguity. If we drop those constraints, we obtain an ILP that has the same structure as the *basic ILP* defined by Haunert [33]. This basic ILP follows the approach of Zoltner [70] for districting, in the sense that in every output region one of the input areas is selected as the region's center. To encode this idea, the basic ILP uses a variable $x_{c,v}$ for each pair of areas



$c, v \in V$, which has the following meaning.

$$x_{c,v} = \begin{cases} 1, & \text{if area } v \text{ is assigned to the output region with center } c, \\ 0, & \text{otherwise.} \end{cases}$$

For $c = v$, the variable $x_{c,c}$ expresses whether area c is assigned to itself, meaning whether or not it is selected as a center. Note that with this model we do *not* prescribe the centers before computing an optimal assignment. Instead, every area can become a center.

Each variable $x_{c,v}$ is associated with an assignment cost $a_{c,v}$. The objective function is a weighted sum of the variables.

$$\min \sum_{c \in V} \sum_{v \in V \setminus \{c\}} a_{c,v} \cdot x_{c,v} \tag{1}$$

The aim for compact output regions, for example, can be expressed by minimizing this objective function with $a_{c,v} = w(v) \cdot d(c, v)$, where $w(v)$ is a weight for area v (reflecting size or population) and $d(c, v)$ is the Euclidean distance between the centroids of c and v [34].

To obtain a partition of the set V of areas into regions, we require with the following constraint that every area is assigned to exactly one center.

$$\sum_{c \in V} x_{c,v} = 1 \quad \text{for each } v \in V \tag{2}$$

Next, we make sure that an area v is assigned to a center $c \in V$ only if c is actually selected as a center.

$$x_{c,v} \leq x_{c,c} \quad \text{for each } c \in V, v \in V \setminus \{c\} \tag{3}$$

To impose constraints on the size or population of each output region, we use the weight $w(v)$ that we defined for Objective (1). The following constraint ensures that every output region has a weight of at least $w_{\min} \in \mathbb{R}^+$.

$$\sum_{v \in V} w(v) \cdot x_{c,v} \geq w_{\min} \cdot x_{c,c} \quad \text{for each } c \in V \tag{4}$$

Similarly, we could define an *upper* bound on the weights of the regions.

Interpreting both the previous constraints and the results makes sense only if the integrality constraint (Constraint (5)) is taken into account.

$$x_{c,v} \in \{0, 1\} \quad \text{for each } c \in V, v \in V \tag{5}$$

A solution satisfying this constraint is termed an *integral solution*.

2.2 Shirabe’s model for contiguity-constrained spatial unit allocation

The model presented in this section is a straightforward adaption of a model for spatial unit allocation by Shirabe [61, 62]. It extends the basic ILP from Section 2.1 with a set of continuous variables and an additional set of constraints to ensure contiguous regions. Since the additional variables are continuous, Shirabe’s model leads to a *mixed* integer linear program (MILP), which basically can be solved with the same solution techniques as an ILP.

In contrast to Shirabe, we neither demand a single output region [61] nor a partition of the input graph into a prescribed number of regions [62] and, thus, have to make small modifications. We denote the resulting MILP as the *flow MILP* and will use it as a benchmark to evaluate our new cutting-plane method.

The flow MILP relies on the definition of the directed graph $\bar{G} = (V, \bar{E})$, whose set \bar{E} of directed edges (or *arcs*) contains arc (u, v) as well as arc (v, u) for every edge $\{u, v\} \in E$. It uses the idea that multiple commodities are transported (or *flow*) on the arcs of this graph. By controlling the flow of the commodities with suitable constraints, the contiguity of the output regions is ensured.

In our adaption of Shirabe's model, there is one commodity for each potential region center—and thus for each vertex $c \in V$. We define a variable

$$y_{(u,v)}^c \in [0, |V| - 1] \quad \text{for each } (u, v) \in \bar{E}, c \in V \setminus \{u\},$$

which represents the amount of the commodity for center c that flows on arc (u, v) . Every area v that is assigned to a region center $c \neq v$ is a *source*, that is, it injects one unit of the commodity for c into the flow network. This unit flow is forced to find a way to the region center c —the sole *sink* for the commodity for c —by only passing through areas allocated to the same center. Guaranteeing these properties of the flow is equivalent to guaranteeing the contiguity of the resulting regions. This is done with the following constraints:

$$\sum_{(u,v) \in \bar{E}} y_{(u,v)}^c \leq (|V| - 1) \cdot x_{c,u} \quad \text{for each } c \in V, u \in V \setminus \{c\} \quad (6)$$

$$\sum_{(u,v) \in \bar{E}} y_{(u,v)}^c - \sum_{(v,u) \in \bar{E}} y_{(v,u)}^c = x_{c,u} \quad \text{for each } c \in V, u \in V \setminus \{c\} \quad (7)$$

If $x_{c,u} = 0$, i.e., u is not assigned to center c , Constraint (6) prohibits any outflow of the commodity for c from u ; Constraint (7) forces the inflow and the outflow of this commodity at u to be equal (and thus prohibits any inflow as well). If $x_{c,u} = 1$, i.e., u is assigned to center c , Constraint (7) ensures that u is a source contributing one unit of the commodity for c to the flow network; Constraint (6) is relaxed by setting its right-hand side to a sufficiently large number. Only the center c of a region can be a sink of the commodity for c , since Constraints (6) and (7) are not set up for the case $c = u$.

The flow MILP consists of Objective (1) as well as Constraints (2)–(7). Since G is planar, the flow MILP has $\mathcal{O}(n^2)$ variables and $\mathcal{O}(n^2)$ constraints, where $n = |V|$ is the number of input areas.

2.3 Area aggregation in map generalization

Area aggregation is an important sub-problem of map generalization, which (among others) also involves line simplification [17], selection [48], and displacement [4, 59]. While some approaches exist to treat all or multiple sub-problems of map generalization in a comprehensive way [27, 32, 67], research is also ongoing to improve the algorithmic solutions for each sub-problem.

Area aggregation can be driven by minimal graphic dimensions for a target scale, but in this article we consider it in the context of *statistical generalization* [8], which aims at a less detailed and non-graphical model to permit spatial analysis on a higher level of abstraction.



In particular, we consider the grouping of administrative regions with unemployment rates as an example, in which the aim is to reveal large spatial patterns of unemployment. Our aim is a high homogeneity with respect to the unemployment rate in each output region, thus we consider similarity of attributes as an important criterion for grouping. Additionally, we consider size and compactness in our model.

We do note that grouping areas based on unemployment rates is related to the delineation of *labor market areas*, which, however, are usually defined based on travel-to-work patterns rather than on attribute similarity [11, 25, 55, 64]. Our work is also related to the modifiable areal unit problem (MAUP) [54], which states that the delineation of districts is a source of statistical bias. With our aim for homogeneous output regions we try to keep this bias low. Obviously, aggregation not only reveals large spatial patterns but also suppresses fine-grained and possibly sensitive information. Therefore, we also see a relevance of area aggregation for privacy protection [41].

According to the definition of area aggregation by Haunert and Wolff [34], the areas (both in the input and in the output) have one attribute. For each input area $v \in V$, the attribute value is denoted by γ_v . The problem definition requires that in each output region one of the contained input areas c is selected as a center, which dictates the attribute value of that output region as γ_c . This means that no “new” attribute values arise. Furthermore, the basic ILP (with additional constraints ensuring contiguity) suffices for area aggregation— if the assignment costs $a_{c,v}$ are appropriately set. Haunert and Wolff [34] define $a_{c,v}$ to express two criteria.

First, since $x_{c,v} = 1$ implies that area v changes its attribute value (or *color*) from γ_v to γ_c and an objective is to keep such changes small, a cost f_{recolor} is charged that depends on the *distance* from γ_v to γ_c . This distance is generally defined with a function $\delta: \Gamma^2 \rightarrow \mathbb{R}_{\geq 0}$, where Γ is the set of all attribute values. It is chosen to reflect the *dissimilarity* of the attribute values, which implies that by minimizing f_{recolor} the objective for grouping similar areas is addressed. The cost for color change is defined as

$$f_{\text{recolor}} = \sum_{c \in V} \sum_{v \in V} w(v) \cdot \delta(\gamma_c, \gamma_v) \cdot x_{c,v}. \quad (8)$$

Second, the objective for geometrically compact output regions is generally modeled with a cost $f_{\text{non-compact}}$. In this article, we define

$$f_{\text{non-compact}} = \sum_{c \in V} \sum_{v \in V} w(v) \cdot d(c, v) \cdot x_{c,v}, \quad (9)$$

where $d: V^2 \rightarrow \mathbb{R}_{\geq 0}$ is the Euclidean distance between the centroids of the areas in V .

The overall objective is to minimize

$$f = \alpha \cdot f_{\text{non-compact}} + (1 - \alpha) \cdot f_{\text{recolor}}, \quad (10)$$

where $\alpha \in [0, 1]$ is a parameter that weights the two criteria. Accordingly, we use Objective (1) with assignment costs

$$a_{c,v} = w(v) \cdot \left(\alpha \cdot d(c, v) + (1 - \alpha) \cdot \delta(\gamma_c, \gamma_v) \right). \quad (11)$$

Haunert and Wolff [34] assumed that Γ is a discrete set of land cover classes and that the distance δ measures the semantic dissimilarity of classes. In our example, however, we are

given areas with unemployment rates and thus a quantitative attribute. More specifically, $\Gamma = [0, 1]$ is the set of real numbers between 0 and 1. We simply define the distance

$$\delta(\gamma_1, \gamma_2) = |\gamma_2 - \gamma_1| \quad \text{for } \gamma_1, \gamma_2 \in \Gamma. \quad (12)$$

With this definition, the requirement that each region contains an area that does not change its attribute value has no influence on the optimal objective value. In particular, if all areas have the same weight, the cost f_{recolor} would be minimized by selecting the median of all attribute values in a region.

A problem that is similar to our problem variant of area aggregation with a quantitative attribute is the aggregation of 3D building models based on their heights. Guercke et al. [30] have approached this problem by integer linear programming, using a model that is similar to the model of Haunert and Wolff [34].

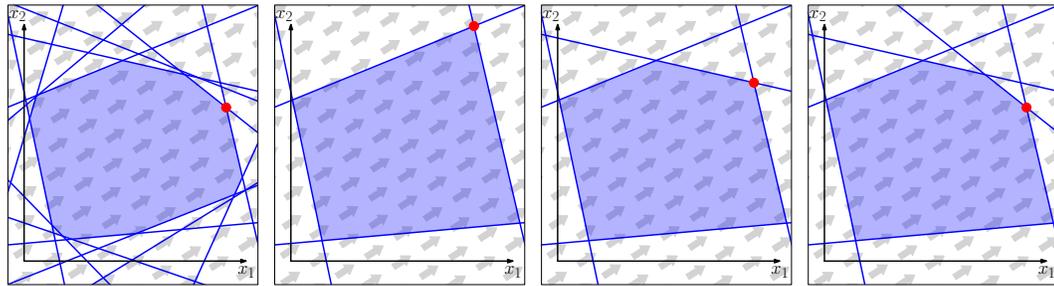
3 Handling ILPs with large sets of constraints

The cutting-plane method that we will present in this article is based on an ILP consisting of Objective (1), Constraints (2)–(5), and an exponential number of constraints that ensure contiguity, which we term *contiguity constraints*. Such a large set of constraints is only reasonable with special constraint-handling techniques, which we sketch in this section. Generally, the strategy is to first disregard some constraints of the original model and to consider a reduced model that contains only few constraints ensuring some very basic properties of a solution. In our example, we disregard the contiguity constraints.

The most simple approach following this strategy is to first compute an optimal solution to the reduced model. Then, it is often easy to check whether the solution is also a solution to the original model. For example, non-contiguous regions can be easily and efficiently detected using a breadth-first search [13]. Generally, if the solution satisfies all constraints of the original model, one is done. Else, one can augment the model with exactly those constraints that were found to be violated—and solve it again. This process is repeated until an optimal solution to the current model is found and asserted to satisfy all constraints of the original model. Drexler and Haase [18] and Duque [20], for example, use this approach in order to solve the problem of sales force deployment and the p -regions problem, respectively. An advantage of this approach is that the ILP solver can still be thought of and used as a black box. That is, after each augmentation of the model with constraints, the ILP solver can be applied with default settings and without any intervention. Also, the approach can be more efficient than an approach with a complete instantiation of the model. On the other hand, solving the model to optimality after each augmentation step can be quite inefficient.

A better approach is to repeatedly augment the model but to avoid computing an optimal solution after each augmentation step. Instead, one can halt some ILP solvers as soon as they find a new *incumbent solution*, that is, an integral solution that satisfies all constraints of the current model and that is better than any such solution found so far. As in the most simple approach, this incumbent solution needs to be inspected using an algorithm that is specifically designed and implemented for that purpose, e.g., in the case of area aggregation, an algorithm that checks whether the regions corresponding to the solution are contiguous. If a constraint violation (e.g., a non-contiguous region) is found, a new constraint is added and the optimization process is resumed—rather than repeated from scratch. This kind of intervention (i.e., halting the solver, inspecting the incumbent





(a) The LP with all constraints. (b) The LP with a selected subset of constraints. (c) The LP with an enhanced subset of constraints. (d) The LP after all the necessary separation steps (here: two).

Figure 3: A geometric interpretation of an LP with two variables $x_1, x_2 \in \mathbb{R}$ (constraints presented as blue lines, objective function presented as gray arrows) and its solution (red). While (a) depicts the complete LP and an optimal solution, (b) to (d) show the advantages of using a separation algorithm: Starting with only a subset of constraints (b) we extend the LP with constraints which are violated by the solution to the current LP (c) as long as we can find any (see Figure 4). When the separation algorithm cannot find any violations (d) the current solution is valid and optimal with respect to the complete LP of the original model.

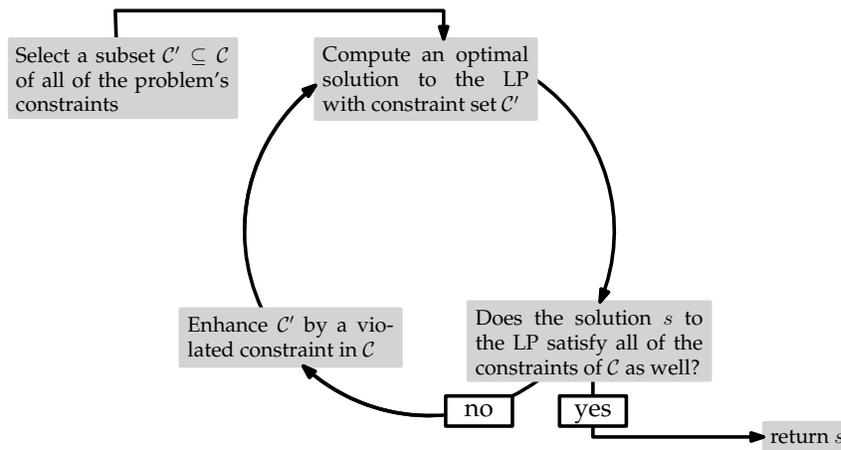


Figure 4: The way from Figure 3(b) to (d) as a diagram; the separation algorithm leads to a point of diversion depending on whether a violated constraint exists.

solution, augmenting the model, and resuming the optimization process) can be done via interfaces specified in the programming libraries of solvers like CPLEX and Gurobi. As the most simple iterative approach, this approach finally yields a globally optimal solution. Examples for the application of this method in computational cartography are presented by Haurert and Wolff [35] and Nöllenburg and Wolff [53].

Finally, the approach that we apply in this article generates relevant constraints without relying on incumbent (i.e., integral) solutions to the ILP. This is advantageous because finding an integral solution can be just as hard as finding an optimal solution to a model and,

therefore, may take very long. Instead of inspecting incumbent solutions for constraint violation, we inspect optimal solutions of *LP relaxations*—in the LP relaxation corresponding to a model the variables are allowed to take fractional values, which makes the problem less complex, even efficiently solvable [40]. Most ILP solvers begin with solving the LP relaxation of the given model to find a lower bound (or upper bound in case of a maximization problem) of the objective function, usually by applying a variant of the simplex algorithm [16], which is fast in practice. Furthermore, during the optimization process, most solvers also solve LP relaxations for sub-instances (i.e., branch-and-bound nodes) in which the values of some variables are fixed. In any case, we test whether a solution of an LP relaxation of the current model satisfies all constraints of the original model. More precisely, we use a *separation algorithm*, that is, an algorithm that either asserts that all constraints are satisfied or yields a violated constraint, with which we then augment the model (see Figures 3 and 4). For the Traveling Salesperson Problem¹, for example, this approach outperforms other ILP approaches [56]. In comparison to the previous approach based on incumbent solutions, one avoids an extensive exploration of the branch-and-bound tree. On the other hand, designing a separation algorithm can be a non-trivial task. If the constraints are of a certain type, the separation step can be done efficiently (in particular, without explicitly testing all inequalities), for example by computing a maximum flow in an appropriately defined graph. Carvajal et al. [10] deal with forest planning models using this method. In Section 4 we exemplify this approach in detail for area aggregation.

4 A new method for area aggregation using cutting planes

In Section 2 we have modeled all aspects of the problem that we aim to solve. That is, we consider Objective (1) with the assignment costs in Equation (11) and the definition of the distance δ in Equation (12). We minimize this objective subject to Constraints (2)–(5) and constraints ensuring contiguity. Obviously, we could ensure contiguity with Constraints (6) and (7), but in this section we introduce an alternative formulation that we use with our cutting-plane method. An advantage of this formulation is that we get along with the binary variables $x_{c,v}$ and without the additional variables $y_{(u,v)}^c$.

Instead of setting up the model completely, we let the solver begin with the basic ILP from Section 2.1. When the solution to the LP relaxation in a branch-and-bound node is found, we intervene and check whether some of the not yet added contiguity constraints presented in the following are violated and add at least one of them in that eventuality.

With the following constraints we ensure contiguity. They are inspired by the work of Carvajal et al. [10] and Drexler and Haase [18], where they were set up in order to solve different problems: forest planning and sales territory alignment. The algorithm of Carvajal et al. solves a problem asking for the selection of a single output region and is therefore not directly applicable to the area aggregation problem. Drexler and Haase deal with districting but add violated constraints based not on the solution to the LP relaxation in a branch-and-bound node but to an optimal solution to the current model. Afterwards, they run experiments to calculate only upper and lower bounds for larger instances, but no optimal solutions. We do not only present these constraints in combination but also emphasize the

¹Dealing with the Traveling Salesperson Problem means determining a shortest path visiting every city of a given set of cities. It serves as a basis for various common GI science problems like one of its extensions, the vehicle routing problem [5, 12, 42].

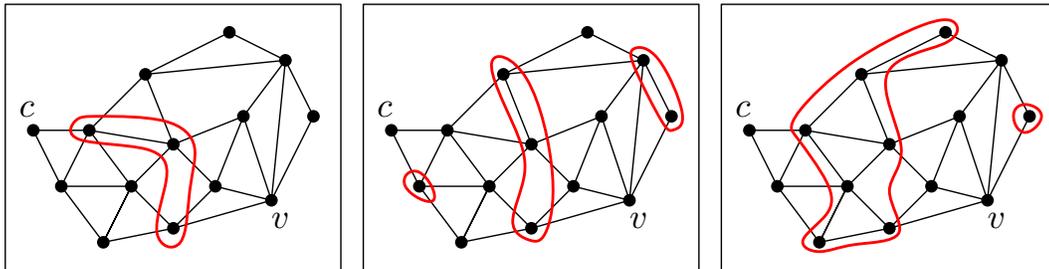


Figure 5: An example of an adjacency graph G with various (c, v) -separators (red): Every path from v to c contains at least one vertex of the respective separator.

advantages of cutting-plane methods. Furthermore, we contribute an ILP formulation for area aggregation.

4.1 Constraints completing the ILP formulation

In the following, we present two different kinds of constraints. While combining the basic ILP from Section 2.1 with the constraints presented in Section 4.1.1 is necessary and sufficient for area aggregation, the constraints in Section 4.1.2 work in a supporting way. These supportive constraints mainly enforce the minimum-weight constraints (Constraint (4)) in a more determined manner.

4.1.1 Contiguity constraints based on vertex separators

Let $c, v \in V$ be two arbitrary areas. In the following we consider c the center of a region and the possibility of assigning v to this region, which is represented with the binary variable $x_{c,v}$.

Furthermore, let $\mathcal{S}_{c,v} \subseteq 2^V$ be the set of all (c, v) -separators in G , where a set $S \subseteq V \setminus \{c, v\}$ is called a (c, v) -separator if every path from c to v in G contains at least one vertex in S (see Figure 5). If v is allocated to the region with center c , then—for the sake of contiguity—for each (c, v) -separator $S \in \mathcal{S}_{c,v}$ at least one area of S has to be allocated to this region as well. In linear terms this condition is expressed as follows:

$$\sum_{u \in S} x_{c,u} \geq x_{c,v} \quad \text{for each } S \in \mathcal{S}_{c,v}, c, v \in V. \tag{13}$$

If $c = v$ or $\{c, v\} \in E$, that is, areas c and v are identical or adjacent, the set of (c, v) -separators $\mathcal{S}_{c,v}$ is empty. Consequently, there is no constraint described in Formula (13) for these cases. In general, the number of separators is in $\mathcal{O}(2^{|V|})$.

The basic model (see Section 2.1) with the constraints presented here is already fully operative: all output regions are contiguous if and only if none of the constraints based on vertex separators is violated. To see why, we only consider the case where the right-hand side equals 1, since otherwise the inequalities from Constraint (13) are always true. Suppose an arbitrary region containing a center c and another area v is contiguous. This means a path p between c and v exists in G that consists only of vertices belonging to the same region. Since each separator $S \in \mathcal{S}_{c,v}$ contains at least one vertex of any path

from c to v , it also contains a vertex of path p and thus an area of the region in question. Consequently, the inequalities from Constraint (13) are fulfilled. Now, let us assume that a region $R \subseteq V$ is not contiguous, i.e., R consists of at least two connected components. Thus, it is possible to take a look at the region's center c and an area v contained in another connected component than c . Then the set $V \setminus R$ is a (c, v) -separator and the inequality from Constraint (13) is violated for $S = V \setminus R \in \mathcal{S}_{c,v}$.

4.1.2 Supportive constraints inspired by the minimum-weight requirement

If a separator $S \in \mathcal{S}_{c,v}$ has certain properties, we define another constraint: since S is a (c, v) -separator, the set $V \setminus S$ consists of at least two connected components—one of which contains c and one of which contains v . Let $C(S, c)$ be the connected component in $V \setminus S$ containing c . If the total weight of $C(S, c)$, that is $\sum_{u \in C(S, c)} w(u)$, does not exceed the minimum weight w_{\min} demanded for a resulting region, then every region with center c has to contain at least one area of the separator S . Or stated as a linear inequality:

$$\sum_{u \in S} x_{c,u} \geq x_{c,c} \quad \text{for each } S \in \mathcal{S}_{c,v} \text{ with } \sum_{u \in C(S, c)} w(u) < w_{\min}, \quad c, v \in V \quad (14)$$

Again, the inequality is always fulfilled for $x_{c,c} = 0$. Only if $x_{c,c} = 1$ (i.e., c is declared a center) the allocation of at least one area in the respective separator is required.

Constraint (14) alone does not suffice to achieve contiguity. When combining it with Constraint (13), however, it acts in a supportive manner. To see why, we contrast the contiguity constraint (Constraint (13)) with the supportive constraint (Constraint (14)). We observe that they only differ with respect to their right-hand sides, which are $x_{c,v}$ and $x_{c,c}$, respectively. Constraint (3) ensures $x_{c,c} \geq x_{c,v}$ and, thus, Constraint (14) is at least as restrictive as Constraint (13) for any $v \in V$. That is, every solution to an LP relaxation considering Constraints (3) and (14) fulfills Constraint (13). However, the solution to an LP relaxation considering Constraints (3) and (13) cannot give this guarantee with respect to Constraint (14). Transferring this observation to Figure 3, we note that a more restrictive constraint cuts away more parts of the solution polytope (marked as a blue region in the figures). This results in a better performance of solvers based on branch and bound [52].

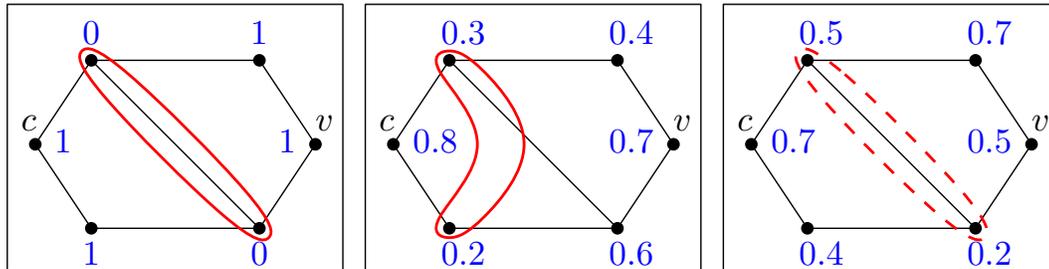
4.2 Adding the constraints

As described in Section 3, the ILP is built up step by step. In this section, we describe a separation algorithm, that is, an algorithm that allows us to find at least one violated contiguity constraint if there exists any; see Section 4.2.1. Then, we show how such a contiguity constraint may also provide a supportive constraint; see Section 4.2.2. Afterward, in Section 4.2.3, we discuss how to speed up the computation by restricting the search for violated constraints, and we argue why this does not harm the correctness. Finally, in Section 4.2.4, we present another method that may find additional violated constraints without much computational overhead.

A pseudocode formulation of these algorithms can be found in Section Appendix B.

4.2.1 Finding violated contiguity constraints using minimum-weight vertex separators

If we assume that the integrality constraints $x_{c,v} \in \{0, 1\}$ are satisfied for all variables (see Figure 6(a)), finding a violation of Constraint (13) is straightforward. For example, for a



(a) With integral $x_{c,u}$ -values, it is easy to find a vertex separator, for which the contiguity constraint is violated.

$$0 + 0 = \sum_{u \in S} x_{c,u} < x_{c,v} = 1$$

(b) For fractional $x_{c,u}$ -values, a violation of a contiguity constraint is harder to find. A violation exists, for example, for the depicted separator.

$$0.3 + 0.2 = \sum_{u \in S} x_{c,u} < x_{c,v} = 0.7$$

(c) The contiguity constraint is satisfied for the minimum-weight (c, v) -separator. Thus, no contiguity constraint is violated for any (c, v) -separator.

$$0.5 + 0.2 = \sum_{u \in S} x_{c,u} \geq x_{c,v} = 0.5$$

Figure 6: In this example with fixed $c, v \in V$, we examine the search for a (c, v) -separator, for which the contiguity constraint (Constraint (13)) is violated. For a vertex $u \in V$, we consider the value $x_{c,u}$ from the solution of an LP relaxation as its vertex weight (blue). It depends on these values whether a violation of a contiguity constraint exists. Separators which are interesting in this context are indicated in red.

fixed center c , we could consider the subgraph of G induced by all nodes v with $x_{c,v} = 1$. If c and a node v lie in different connected components of this graph, a violation of a contiguity constraint exists. However, since we want to find constraints in a preliminary stage of the solution (i.e., in the solution of the LP relaxation of the ILP), we need to deal with variables of fractional values (see Figure 6(b)).

To find contiguity constraints that are violated by the solution to the current LP relaxation, we proceed as follows (see also Algorithm 1 in Section Appendix B): taking a closer look at Constraint (13), we observe that separators providing a smaller sum on the constraint’s left-hand side are more likely to implicate a violation of the corresponding constraint, since the right-hand side does not depend on the choice of the separator. For every potential center c , we therefore take for every $v \in V$ the value $x_{c,v}$ of the current solution as the weight of v in G and focus on *minimum-weight* (c, v) -separators afterward—in Section Appendix A we describe how to compute minimum-weight vertex separators by using minimum edge cut algorithms [13,21,26]. The reason why this is effective is the following: if the inequality from Constraint (13) is violated for an arbitrary (c, v) -separator S , it is also violated for a minimum-weight (c, v) -separator S^* since $x_{c,c} > \sum_{u \in S} x_{c,u} \geq \sum_{u \in S^*} x_{c,u}$ holds in that case. Thus, for specific $c, v \in V$, looking at a minimum-weight (c, v) -separator guarantees to notice a violation if there is one (see Figure 6(c)). In particular, computing and examining minimum-weight (c, v) -separators for every potential center $c \in V$ and every $v \in V \setminus \{c\}$ solves the separation problem.

4.2.2 Finding supportive constraints

For each vertex separator computed with the algorithm from Section 4.2.1 we proceed as follows to find a supportive constraint. The vertex separator splits V into at least two

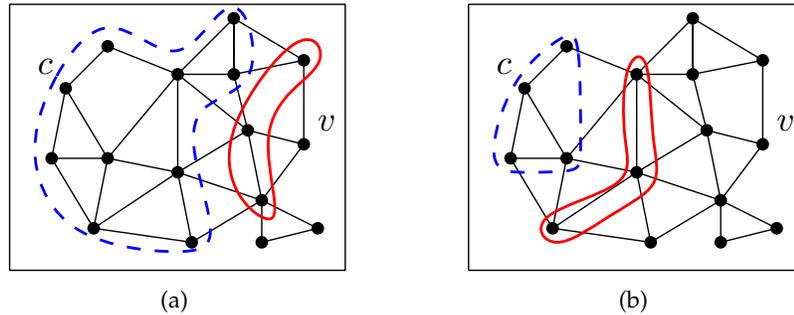


Figure 7: The (c, v) -separator (red) divides the graph into at least two connected components. The component of c (dashed, blue) in (b) is more likely to offer a Constraint (14) than the one in (a).

connected components in G . If one of these components forms a region with less than the minimum weight, the corresponding separator offers Constraint (14) for any potential center in that component. Therefore, we check for every area in the connected component of c whether the respective Constraint (14) is violated for this area as a center. If that is the case, we add the violated constraint to the model. This means we compute a separator principally for a certain center c but also use it to set up constraints independent from c .

The algorithm from Section 4.2.1 yields a minimum-weight (c, v) -separator. Such a separator is not unique. Among the minimum-weight (c, v) -separators we prefer those which are closer to c (see Figure 7). The reason why we prefer separators closer to c is Constraint (14): That way, it is more likely to detect violations of Constraint (14) and consequently more likely to add a supportive constraint. In order to find the minimum-weight (c, v) -separator closest to c we use the fact that our algorithm is based on the search for a minimum edge cut. Here, common algorithms follow the same scheme and return the minimum edge cut closest to a certain vertex [13].

4.2.3 Restricting the search for violated constraints

A straightforward implementation of the algorithm described in Section 4.2.1 implicates the computation of a quadratic number of vertex separators every time an LP relaxation is solved optimally. In order to reduce this number, we make restrictions described in the following.

We read the value of a variable $x_{c,v}$ for arbitrary $c, v \in V$ as the tendency of v to be allocated to c . With regard to Constraint (2), we see that, for a fixed $v \in V$, the average value of the variables $x_{c,v}$ is $\frac{1}{|V|}$. Therefore, we interpret $x_{c,v} \geq \frac{1}{|V|}$ as an indicator that allocating v to c (or, in the case $v = c$, declaring v a center) is—for the moment—considered reasonable. Conversely, a value less than $\frac{1}{|V|}$ implicates that an allocation to a different center is preferable. Hence, we impose the following restrictions on the method presented in Sections 4.2.1 and 4.2.2:

- We compute minimum-weight (c, v) -separators only for areas v with $x_{c,v} \geq \frac{1}{|V|}$.
- We take centers c with $x_{c,c} < \frac{1}{|V|}$ only into consideration if no violation is detected for more probable centers.

- We add a supportive constraint (Constraint (14)) only for areas v in the connected component (not fulfilling the minimum-size requirement) of the examined center for which $x_{v,v} \geq \frac{1}{|V|}$ holds.

Although we fail to notice violations of the inequality from Constraint (13) for areas with $x_{c,v} < \frac{1}{|V|}$ and, thus, to solve the separation problem properly, this fact does not pose a risk to finding a contiguous solution. As we handle these violations in a branch-and-bound node, the solver goes on branching and bounding in the node's sub-instances. It continues until either an integral solution is found or the problem becomes infeasible. If a sub-instance is infeasible, there is no need to worry about whether the number of added constraints is too small. In case of an integral solution, there are two possible scenarios for each potential region with center c : either $x_{c,c} = 0$ and with it $x_{c,v} \leq x_{c,c} = 0$ for every $v \in V$ (see Constraint (3)), i.e., the region is not considered in the solution and therefore in no need of a validation of contiguity. Or $x_{c,c} = 1 \geq \frac{1}{|V|}$, which results in a verification of the region's contiguity and implies that we find violations if existing.

4.2.4 Finding violated contiguity constraints using connected components

For the following method of finding a useful vertex separator (see also Algorithm 2 in Section Appendix B), we interpret for every probable center c (i.e., $x_{c,c} \geq \frac{1}{|V|}$) the values of the variables $x_{c,v}$ for $v \in V$ as described in Section 4.2.3, only stricter: if $x_{c,v} \geq \frac{1}{|V|}$, we consider v allocated to the region with center c , otherwise we do not. Therefore, we take a look at

$$V' = \left\{ v \in V \mid x_{c,v} \geq \frac{1}{|V|} \right\}$$

and to the subgraph of G induced by V' . In this subgraph we are able to detect connected components of the areas tending to be allocated to c . For every connected component $U \subseteq V' \subseteq V$ with $c \notin U$, we examine the set of adjacent areas in V , that is

$$B_U := \left\{ w \in V \setminus U \mid \exists u \in U : \{u, w\} \in E \right\}, \quad (15)$$

and take it as a vertex separator. Subsequently, we add a Constraint (13) for every $u \in U$ (and c). This way we have a good chance of finding additional violated contiguity constraints without much computational overhead.

5 Results and discussion

For a series of experiments we used a computer with a 3.2 GHz Intel Core i5-4570 Processor and 16 GB RAM. Our program is written in Java and besides the Gurobi [31] interface for Java, we use in particular the library JGraphT [51] for building the adjacency graph and performing operations on it (see also Section Appendix B). The data for the experiments is provided by the European statistical service (Eurostat, [23]) and the European Observation Network for Territorial Development and Cohesion (ESPON, [22]). This data is gathered with regard to the NUTS (Nomenclature des unités territoriales statistiques) subdivision of the European Union. As an example, we examine the unemployment rate [22] for the NUTS-3 subdivision of continental France (see Figure 8(b)) which means processing a subdivision of 94 departments. In contrast to Haunert and Wolff's original work [33, 34] we

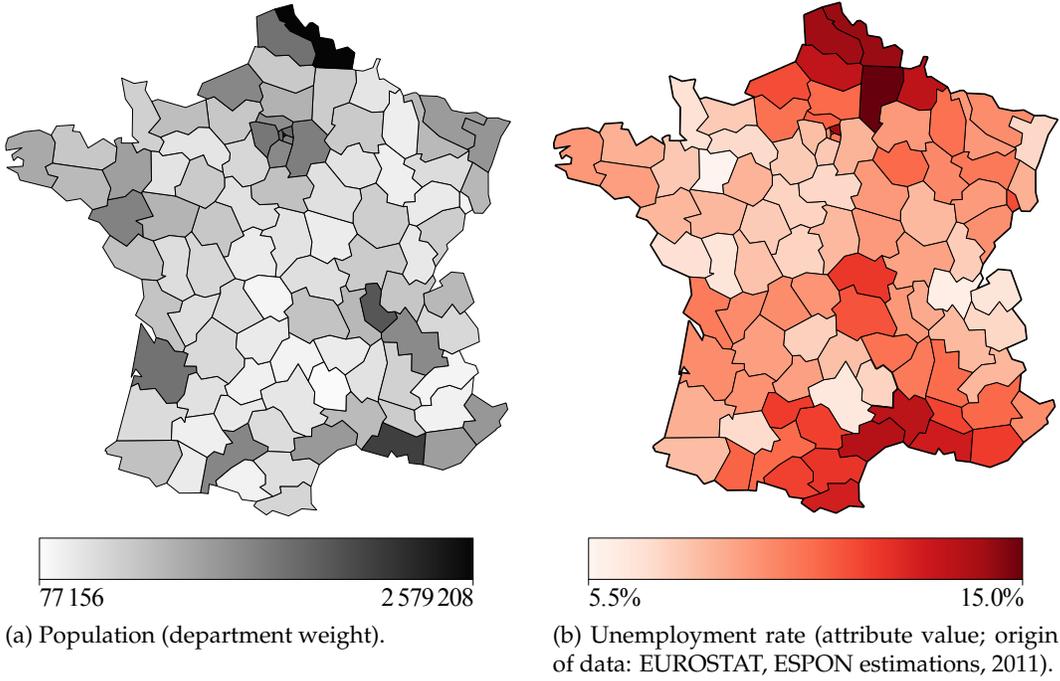


Figure 8: input data for statistical aggregation of France's departments

take the population of a department [23] (see Figure 8(a)) instead of its area as the department's weight. This means we compare the methods described in Sections 2.2 and 4 on the third level administrative subdivision of a major European country. We do not only compare the effects of α , the factor weighting the cost functions f_{recolor} and $f_{\text{non-compact}}$ in the objective function, on the result and the computation time, but also examine the influence of the choice of w_{\min} , the minimum weight required of a region in the output.

As attribute domain, $\Gamma = [0, 1]$ is given and we write $\gamma_v \in \Gamma$ for the attribute value, i.e., unemployment rate, of an area $v \in V$. According to Section 2.3, we define $\delta(\gamma_1, \gamma_2) = |\gamma_1 - \gamma_2|$ for arbitrary $\gamma_1, \gamma_2 \in \Gamma$ as the cost for recoloring. The cost for non-compactness is defined through the Euclidean distance d between centroids. Observing Equations (1) and (8)–(10) we get the following overall cost function as the objective function:

$$\sum_{c \in V} \sum_{v \in V \setminus \{c\}} w(v) \cdot \left(\alpha \cdot d(c, v) + (1 - \alpha) \cdot \delta(\gamma_c, \gamma_v) \right) \cdot x_{c,v} \quad (16)$$

As for the weighting factor α , we make the following choice: with $\alpha \in \{0, 1\}$ we want to present the extreme solutions. For $\alpha = 1$ we receive compact regions of a certain weight ignoring any statistical similarities between the areas. For $\alpha = 0$ we see a solution with least recoloring costs but consisting of regions of mostly unconventional shape. According to the experiments that we present in this section, a weighting factor of $\alpha = 2 \cdot 10^{-5}$ still yields almost optimally compact shapes and $\alpha = 1.25 \cdot 10^{-6}$ almost optimally homogeneous regions. Therefore, we particularly tested our method for $\alpha = 0$ and $\alpha = 1$ as well as for $1.25 \cdot 10^{-6} \leq \alpha \leq 2 \cdot 10^{-5}$.

In our experiments, we set w_{\min} to 5% and 10% respectively of the total population of continental France. With this setting, we end up with a partitioning into a number of larger regions which is approximately at the same scale as the actual NUTS-2 subdivision of continental France, which consists of 12 regions.

Figure 9	α	t_{cut}	t_{flow}	$t_{\text{flow}}/t_{\text{cut}}$	#out
(a)	1	30 s	497 s	16.6	18
(b)	$2.0 \cdot 10^{-5}$	21 s	502 s	23.9	18
(c)	$5.0 \cdot 10^{-6}$	50 s	$14\,229\text{ s} \approx 4.0\text{ h}$	284.6	17
(d)	$2.5 \cdot 10^{-6}$	100 s	> 5 d	> 4 320.0	16
(e)	$1.25 \cdot 10^{-6}$	451 s	> 5 d	> 957.9	16
(f)	0	$18\,413\text{ s} \approx 5.1\text{ h}$	> 5 d	> 23.4	14

(1) Running times for the experiments with minimum weight $w_{\min} = \frac{1}{20} \sum_{v \in V} w(v)$.

Figure 10	α	t_{cut}	t_{flow}	$t_{\text{flow}}/t_{\text{cut}}$	#out
(a)	1	20 s	35 s	1.8	9
(b)	$2.0 \cdot 10^{-5}$	59 s	267 s	4.5	9
(c)	$5.0 \cdot 10^{-6}$	194 s	$14\,464\text{ s} \approx 4.0\text{ h}$	74.6	9
(d)	$2.5 \cdot 10^{-6}$	495 s	> 5 d	> 872.7	9
(e)	$1.25 \cdot 10^{-6}$	$86\,981\text{ s} \approx 1\text{ d}$	> 5 d	> 4.9	9
(f)	0	> 5 d	> 5 d	—	g^2

(2) Running times for the experiments with minimum weight $w_{\min} = \frac{1}{10} \sum_{v \in V} w(v)$.

Tables (1) and (2): These tables provide information about the running times in seconds (s), hours (h) or days (d) for the corresponding experiments with different minimum weights (see Equation (4)). Experiments were aborted after five days if no optimal solution was found; this is indicated with “> 5 d.” The first column refers to the visual presentation of the result. The second column denotes the α -value used in this experiment (see Equation (16)). In the column marked with t_{cut} one can find the running times of our approach described in Section 4, in the column marked with t_{flow} the times of the state-of-the-art approach of Section 2. The fifth column compares those values by giving the ratio $t_{\text{flow}}/t_{\text{cut}}$. Finally, the column marked with #out gives additional information about the structure of the result by providing the number of resulting regions.

Considering the results in Tables (1) and (2), it comes to attention that our cutting-plane approach outperforms the flow model in every instance. In our experiments, the flow MILP is competitive only in the situation where w_{\min} is 10% of the total population and $\alpha = 1$. In every other case, our ILP is many times faster, as the fifth column of each table indicates. With α declining, i.e., focusing on reducing f_{recolor} , the advantages of the cutting-plane formulation become clearer since the problem becomes harder to solve. This is caused by the fact that focusing on compactness supports contiguity: a compact region is more likely to be contiguous than a region of areas with similar attributes.

In particular, the three last rows of each table deserve our special attention. For both settings of w_{\min} , the solver is not able to return a result using the flow model for $\alpha \leq 2.5 \cdot 10^{-6}$. Leaving the instance with $\alpha = 0$, $w_{\min} = \frac{1}{10} \sum_{v \in V} w(v)$ out of account, each of

²Since the calculations are incomplete for both the flow model and the cutting-plane approach, the result presented here is the best one found with no guarantee of optimality. It is the result of five days of calculation with the cutting-plane approach. The optimality gap is 6.89%, i.e., the resulting cost of an optimal solution is at least $1 - 0.0689 = 0.9311$ times the cost of the solution presented here. (Using the flow MILP, the remaining gap is 51.4%.)

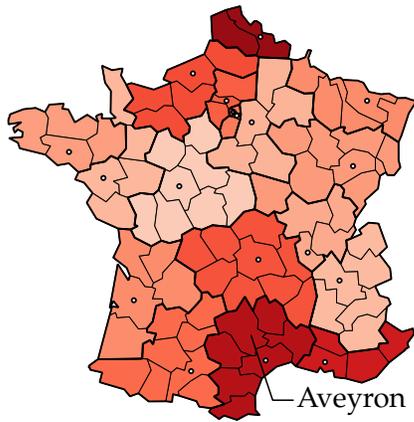
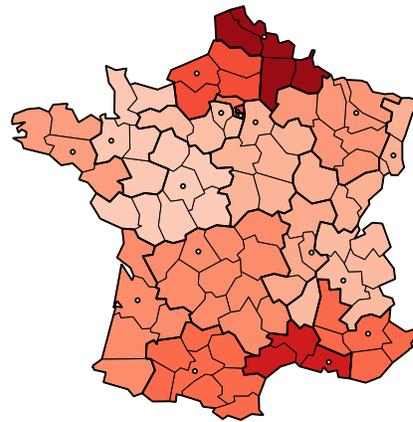
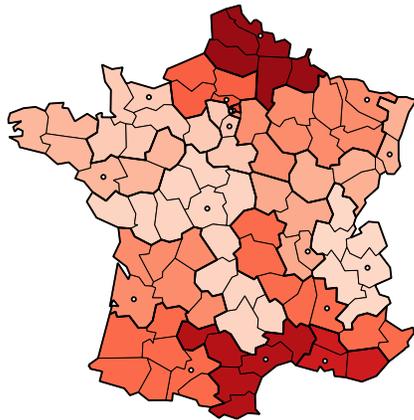
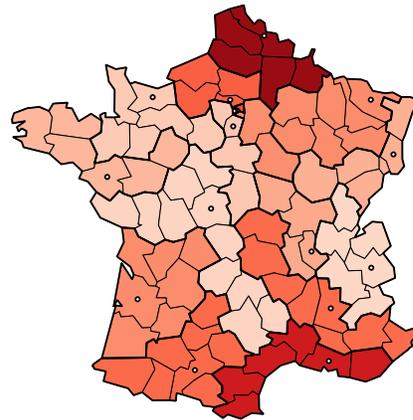
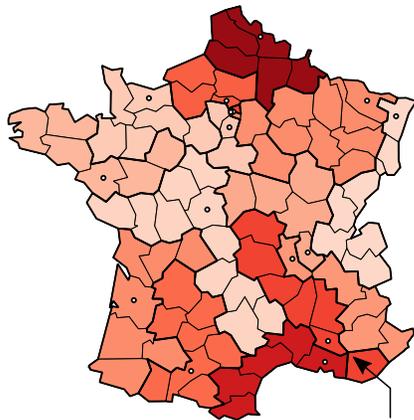
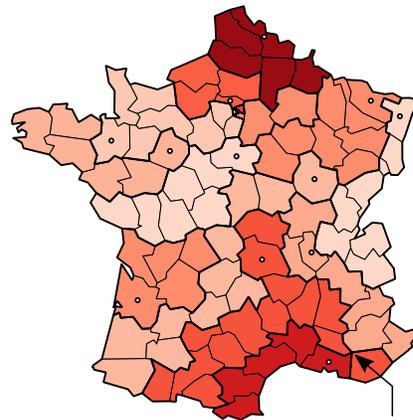
(a) $\alpha = 1$ (b) $\alpha = 2.0 \cdot 10^{-5}$ (c) $\alpha = 5.0 \cdot 10^{-6}$ (d) $\alpha = 2.5 \cdot 10^{-6}$ (e) $\alpha = 1.25 \cdot 10^{-6}$ (f) $\alpha = 0$

Figure 9: Output with w_{\min} equaling 5% of France's total population; thick lines mark aggregated regions colored according to the unemployment rate of their center (\circ). The arrows in Figures (e) and (f) point to extremely narrow parts of contiguous regions.

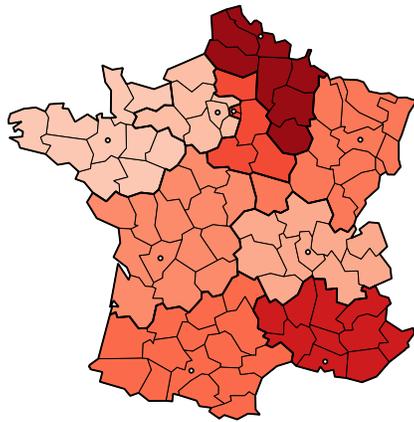
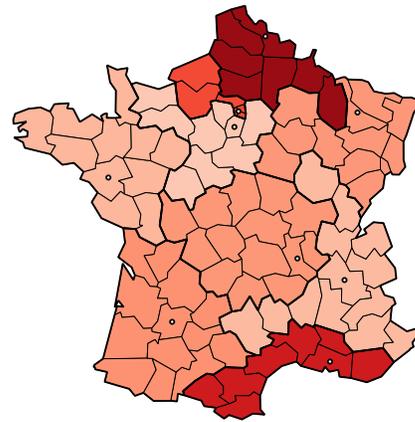
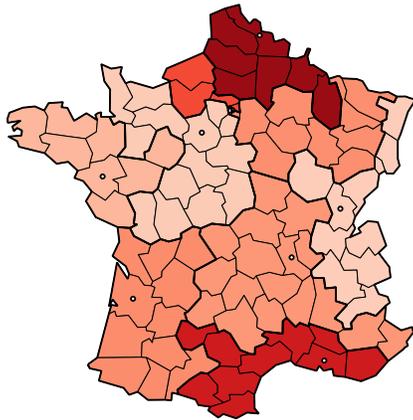
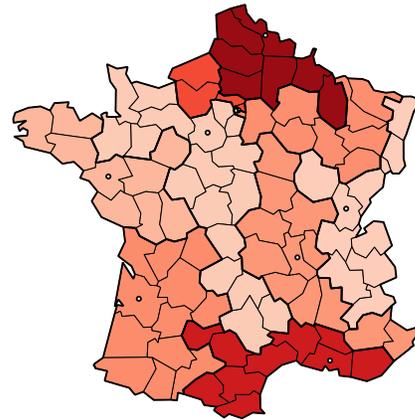
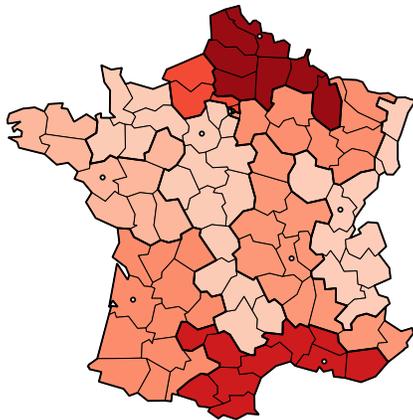
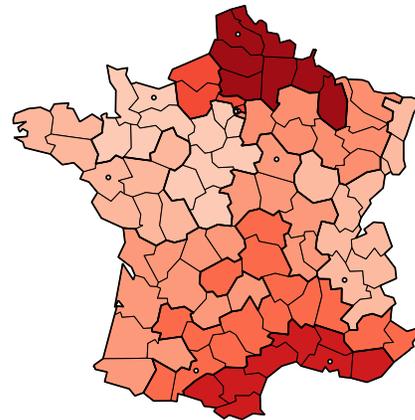
(a) $\alpha = 1$ (b) $\alpha = 2.0 \cdot 10^{-5}$ (c) $\alpha = 5.0 \cdot 10^{-6}$ (d) $\alpha = 2.5 \cdot 10^{-6}$ (same result as (e))(e) $\alpha = 1.25 \cdot 10^{-6}$ (same result as (d))(f) $\alpha = 0$ (*interim result*²)

Figure 10: Output with w_{\min} equaling 10 % of France's total population; thick lines mark aggregated regions colored according to the unemployment rate of their center (o).

these instances is solvable with the cutting-plane algorithm. For three of these instances, the solver returns an optimal solution within a few minutes.

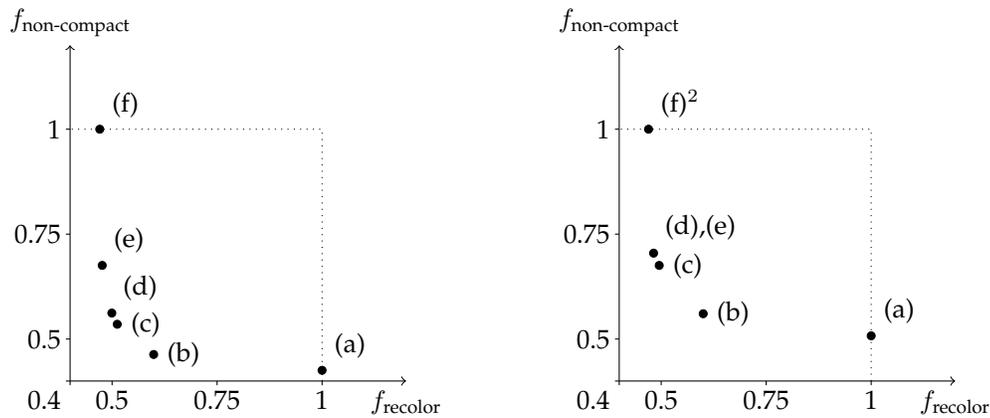
Considering the output in detail, we find that the noticeable higher unemployment rates in northern and southern France (former Nord-Pas-de-Calais and on the Mediterranean coast) are identified and aggregated in every example where similarity is considered (i.e., $\alpha \neq 1$). In general, for most of the departments, the resulting color (unemployment rate) is close to the input. Nevertheless, one also notices larger differences between input and output coloring for various departments. This phenomenon occurs especially for $\alpha = 1$, but also in other cases. For $\alpha \neq 1$, however, this only causes minor changes since these departments have a comparatively small population (see Figure 8(a)) and consequently contribute only little cost. Take, for example, Aveyron, a department in south-central France (see Figure 9(a)) with one of the lowest unemployment rates of all departments. For both $\alpha = 1$ and $\alpha = 0$ it has undergone seemingly expensive recoloring. Considering the objective, this makes sense as less than 0.5 % of France's total population lives in Aveyron.

Another negative aspect is the fact that parts of some of the resulting regions are very narrow. Understandably, this occurs especially for $\alpha = 0$, where resulting regions reach diagonally from one border to another, e.g., in both Figures 9(f) and 10(f) the region containing the department in the very south-west (Pyrénées-Atlantiques). For $\alpha \neq 0$ this phenomenon occurs less distinctly. A problem that arises independent of α is bottlenecks. A bottleneck is a very narrow part of a region connecting two larger parts. In order to resolve this, one has to manipulate the input data or rather its interpretation. Building the adjacency graph G , we consider departments adjacent as soon as they share a boundary. When dealing with a map of France's departments, we have to handle several pairs of departments sharing a borderline of less than 10 km. Without adjusting the adjacency rule, we see results with constellations such as in Figures 9(e) and 9(f), where a region bordering the Mediterranean Sea (marked with an arrow in the south-east) even seems to consist of two components. Here, the departments Var and Vaucluse share less than 1 km of borderline.

Figure 11 gives us additional arguments to disqualify $\alpha = 0$ or $\alpha = 1$ as reasonable weight factors. Let us take a look at the situation for w_{\min} equaling 5 % of the total population (i.e., Figure 11(a)). As $\alpha = 0$ (here: (f)) results in an optimal solution with respect to the cost for recoloring, these costs are higher for every other value of α . But for $\alpha = 1.25 \cdot 10^{-6}$ (here: (e)), recoloring is only approximately 1.2 % more expensive (47.06 % versus 47.65 % of the maximum value) whereas the cost for non-compactness decreases by 32.4 %. The situation is the same for the other value of w_{\min} and similar with regard to the cost for non-compactness. Also, this argument supports our choice for $\alpha \approx 10^{-5}$. In this range, we get near-optimal results for either cost without unreasonable expenses for the other.

6 Conclusion

Our method allows us to solve instances as large as France's third level NUTS division in a reasonable amount of time, which the existing method based on a flow model does not. With our cutting-plane approach, we are able to partition about a hundred areas into nine to eighteen regions multiple times faster than with the compact flow model. In our experiments, the problem becomes harder the fewer output regions are asked for, as Tables (1) and (2) show. With increasing the number of areas of the input, our algorithm reaches its limits as well. Applying our model to Germany's third level division—with 402 districts



(a) Comparison for w_{\min} equaling 5% of France's total population, see Table (1) and Figure 9; absolute maximum values are $8.0 \cdot 10^{12}$ (non-compactness) and $6.4 \cdot 10^7$ (recoloring).

(b) Comparison for w_{\min} equaling 10% of France's total population, see Table (2) and Figure 10; absolute maximum values are $9.2 \cdot 10^{12}$ (non-compactness) and $8.4 \cdot 10^7$ (recoloring).

Figure 11: Costs for non-compactness and recoloring as a percentage of the maximum occurring cost; as 100% of costs for non-compactness we take those arising while minimizing the cost for recoloring (i.e., $\alpha = 0$) and vice versa.

the largest in the European Union—the solver returns no optimal solution within five days. This problem occurs even for $\alpha = 1$, i.e., focusing on compactness only.

Nevertheless, our approach offers unprecedented opportunities. Five of the instances from our experiments are set up too hard to be solved with the flow MILP within five days but are solvable with the cutting-plane algorithm within one day. In particular, the solver is capable to find an optimal solution for each of these settings which consider not only recoloring but also compactness, i.e., $\alpha \neq 0$. Moreover, three of these four cases are solved within only a few minutes. Due to these positive results, we argue that we have increased the range of problem instances that can be solved with proof of optimality such that it includes use cases that are of relevance. In particular, we consider it promising to use our method to produce benchmark solutions and to compare such solutions with solutions of efficient heuristic methods. Such a comparison may help to decide whether or not a heuristic is a justifiable choice. Therefore, we think that both exact and heuristic methods for area aggregation can complement each other and will play their roles in the future.

In future work, we plan to support the ILP solver by calling heuristics during branching or in order to generate an initial solution. Furthermore, we consider it promising to apply our cutting-plane approach to the p -region problem [20], in which the number of output regions is prescribed.

References

- [1] AERTS, J. C. J. H., EISINGER, E., HEUVELINK, G. B. M., AND STEWART, T. J. Using Linear Integer Programming for Multi-Site Land-Use Allocation. *Geographical Analysis* 35, 2 (2003), 148–169. doi:10.1111/j.1538-4632.2003.tb01106.x.

- [2] ALTMAN, M. The computational complexity of automated redistricting: Is automation the answer? *Rutgers Computer and Law Technology Journal* 23, 1 (1997), 81–142. http://heinonline.org/HOL/Page?handle=hein.journals/rutcomt23&g_sent=1&id=87&collection=journals.
- [3] ÁLVAREZ-MIRANDA, E., LJUBIĆ, I., AND MUTZEL, P. The maximum weight connected subgraph problem. In *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, M. Jünger and G. Reinelt, Eds. Springer, Berlin, Germany, 2013, pp. 245–270. doi:10.1007/978-3-642-38189-8_11.
- [4] BADER, M., BARRAULT, M., AND WEIBEL, R. Building displacement over a ductile truss. *International Journal of Geographical Information Science* 19, 8-9 (2005), 915–936. doi:10.1080/13658810500161237.
- [5] BELLMORE, M., AND NEMHAUSER, G. L. The traveling salesman problem: A survey. *Operations Research* 16, 3 (1968), 538–558. doi:10.1287/opre.16.3.538.
- [6] BERTIN, J. *Semiology of Graphics: Diagrams, Networks, Maps*, 1st ed. The University of Wisconsin Press, 1983.
- [7] BOZKAYA, B., ERKUT, E., AND LAPORTE, G. A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research* 144 (2003), 12–26. doi:10.1016/S0377-2217(01)00380-0.
- [8] BRASSEL, K. E., AND WEIBEL, R. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems* 2, 3 (1988), 229–244. doi:10.1080/02693798808927898.
- [9] CARO, F., SHIRABE, T., GUIGNARD, M., AND WEINTRAUB, A. School redistricting: Embedding GIS tools with integer programming. *Journal of the Operational Research Society* 55, 8 (2004), 836–849. doi:10.1057/palgrave.jors.2601729.
- [10] CARVAJAL, R., CONSTANTINO, M., GOYCOOLEA, M., VIELMA, J. P., AND WEINTRAUB, A. Imposing connectivity constraints in forest planning models. *Operations Research* 61, 4 (2013), 824–836. doi:10.1287/opre.2013.1183.
- [11] CASADO-DÍAZ, J. M. Local labour market areas in Spain: A case study. *Regional Studies* 34, 9 (2000), 843–856. doi:10.1080/00343400020002976.
- [12] CHRISTOFIDES, N. The vehicle routing problem. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle* 10, 1 (1976), 55–70. doi:10.1051/ro/197610V100551.
- [13] CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. *Introduction To Algorithms*. MIT Press, 1990.
- [14] COVA, T. J., AND CHURCH, R. L. Contiguity constraints for single-region site search problems. *Geographical Analysis* 32, 4 (Sep 2010), 306–329. doi:10.1111/j.1538-4632.2000.tb00430.x.
- [15] CPLEX, IBM ILOG. 12.2 User's Manual, 2010.

- [16] DANTZIG, G. B. *Linear Programming and Extensions*. Princeton University Press, 1963.
- [17] DE BERG, M., VAN KREVELD, M., AND SCHIRRA, S. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems* 25, 4 (1998), 243–257. doi:10.1559/152304098782383007.
- [18] DREXL, A., AND HAASE, K. Fast approximation methods for sales force deployment. *Management Science* 45, 10 (1999), 1307–1323. doi:10.1287/mnsc.45.10.1307.
- [19] DUQUE, J. C., ANSELIN, L., AND REY, S. J. The max-p-regions problem. *Journal of Regional Science* 52, 3 (Aug 2012), 397–419. doi:10.1111/j.1467-9787.2011.00743.x.
- [20] DUQUE, J. C., CHURCH, R. L., AND MIDDLETON, R. S. The p-regions problem. *Geographical Analysis* 43, 1 (2011), 104–126. doi:10.1111/j.1538-4632.2010.00810.x.
- [21] EDMONDS, J., AND KARP, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM* 19, 2 (1972), 248–264. doi:10.1145/321694.321699.
- [22] EUROPEAN OBSERVATION NETWORK FOR TERRITORIAL DEVELOPMENT AND COHESION. European Regions 2010: Economic Welfare and Unemployment, 2011. http://www.espon.eu/main/Menu_Publications/Menu_MapsOfTheMonth/map1103.html.
- [23] EUROSTAT. Population by single year of age and NUTS 3 region, 2015. http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=cens_11ag_r3&lang=en.
- [24] FISCHETTI, M., AND MONACI, M. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation* 4, 3 (Sep 2012), 239–273. doi:10.1007/s12532-012-0039-y.
- [25] FLÓREZ-REVUELTA, F., CASADO-DÍAZ, J. M., AND MARTÍNEZ-BERNABEU, L. An evolutionary approach to the delineation of functional areas based on travel-to-work flows. *International Journal of Automation and Computing* 5, 1 (2008), 10–21. doi:10.1007/s11633-008-0010-6.
- [26] FORD, L. R., AND FULKERSON, D. R. Maximal flow through a network. *Journal canadien de mathématiques* 8, 0 (1956), 399–404. doi:10.4153/CJM-1956-045-5.
- [27] GALANDA, M. *Automated Polygon Generalization in a Multi Agent System*. dissertation, University of Zurich, Switzerland, 2003.
- [28] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., San Francisco, 1990.
- [29] GARFINKEL, R. S., AND NEMHAUSER, G. L. Optimal political districting by implicit enumeration techniques. *Management Science* 16, 8 (1970), B-495–B-508. doi:10.1287/mnsc.16.8.B495.
- [30] GUERCKE, R., GÖTZELMANN, T., BRENNER, C., AND SESTER, M. Aggregation of LoD 1 building models as an optimization problem. *ISPRS Journal of Photogrammetry and Remote Sensing* 66, 2 (2011), 209–222. doi:10.1016/j.isprsjprs.2010.10.006.

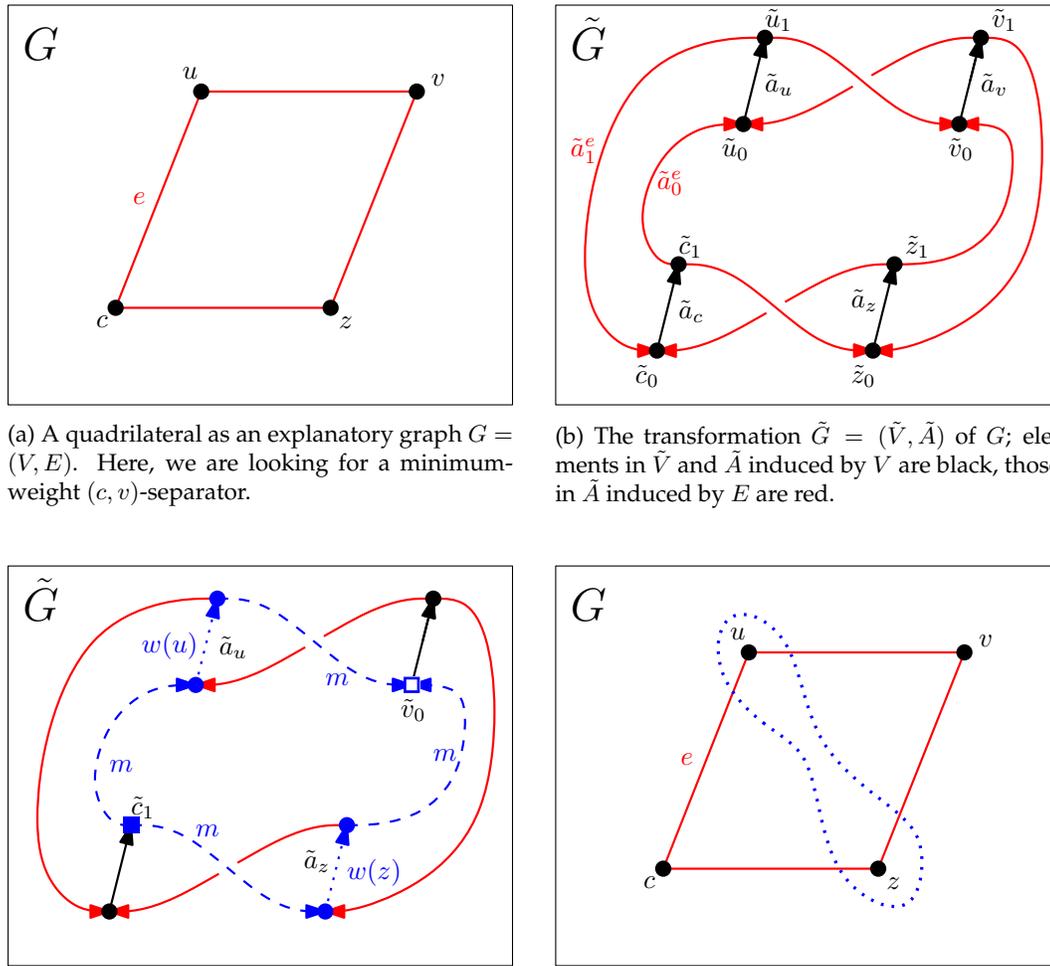
- [31] GUROBI OPTIMIZATION, INC. Gurobi optimizer reference manual, 2015. <http://www.gurobi.com>.
- [32] HARRIE, L., AND WEIBEL, R. Modelling the overall process of generalisation. In *Generalisation of Geographic Information: Cartographic Modelling and Applications*, W. Mackaness, A. Ruas, and L. T. Sarjakoski, Eds. Elsevier, 2007, ch. 4, pp. 67–87. doi:10.1016/B978-008045374-3/50006-5.
- [33] HAUNERT, J.-H. *Aggregation in Map Generalization by Combinatorial Optimization*. Dissertation, Leibniz Universität Hannover, Germany, 2008.
- [34] HAUNERT, J.-H., AND WOLFF, A. Area aggregation in map generalisation by mixed-integer programming. *International Journal of Geographical Information Science* 24, 12 (2010), 1871–1897. doi:10.1080/13658810903401008.
- [35] HAUNERT, J.-H., AND WOLFF, A. Optimal and topologically safe simplification of building footprints. In *Proc. 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACMGIS)* (2010), pp. 192–201. doi:10.1145/1869790.1869819.
- [36] HESS, S. W., AND SAMUELS, S. A. Experiences with a sales districting model: Criteria and implementation. *Management Science* 18, 4, Part II (1971), 41–54. doi:10.1287/mnsc.18.4.P41.
- [37] HESS, S. W., WEAVER, J. B., SIEGFELDT, H. J., WHELAN, J. N., AND ZITLAU, P. A. Nonpartisan political redistricting by computer. *Operations Research* 13, 6 (1965), 998–1006. doi:10.1287/opre.13.6.998.
- [38] HOJATI, M. Optimal political districting. *Computers & Operations Research* 23, 12 (1996), 1147–1161. doi:10.1016/S0305-0548(96)00029-9.
- [39] JÜNGER, M., REINELT, G., AND THIENEL, S. Practical problem solving with cutting plane algorithms in combinatorial optimization. *Dimacs Series in Discrete Mathematics and Theoretical Computer Science* 20 (1995), 111–152. <http://e-archive.informatik.uni-koeln.de/156/>.
- [40] KARMARKAR, N. A new polynomial-time algorithm for linear programming. *Combinatorica* 4, 4 (1984), 373–395. doi:10.1007/BF02579150.
- [41] KWAN, M.-P., CASAS, I., AND SCHMITZ, B. Protection of geoprivacy and accuracy of spatial information: How effective are geographical masks? *Cartographica: The International Journal for Geographic Information and Geovisualization* 39, 2 (2004), 15–28. doi:10.3138/X204-4223-57MK-8273.
- [42] LAPORTE, G. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59, 2 (Jun 1992), 231–247. doi:10.1016/0377-2217(92)90138-Y.
- [43] LI, W., CHURCH, R. L., AND GOODCHILD, M. F. The p-compact-regions problem. *Geographical Analysis* 46, 3 (Jul 2014), 250–273. doi:10.1111/gean.12038.

- [44] LI, W., GOODCHILD, M. F., AND CHURCH, R. L. An efficient measure of compactness for two-dimensional shapes and its application in regionalization problems. *International Journal of Geographical Information Science* 27, 6 (Jun 2013), 1227–1250. doi:10.1080/13658816.2012.752093.
- [45] LIGMANN-ZIELINSKA, A., CHURCH, R. L., AND JANKOWSKI, P. Spatial optimization as a generative technique for sustainable multiobjective land-use allocation. *International Journal of Geographical Information Science* 22, 6 (2008), 601–622. doi:10.1080/13658810701587495.
- [46] LINDEROTH, J. T., AND RALPHS, T. K. Noncommercial software for mixed-integer linear programming. *Integer Programming: Theory and Practice* 3 (2005), 253–303. doi:10.1201/9781420039597.ch10.
- [47] MACEACHREN, A. M. Compactness of geographic shape: Comparison and evaluation of measures. *Geografiska Annaler. Series B, Human Geography* 67, 1 (1985), 53. doi:10.2307/490799.
- [48] MACKANESS, W. A., AND BEARD, M. K. Use of graph theory to support map generalization. *Cartography and Geographic Information Systems* 20 (1993), 210–221. doi:10.1559/152304093782637479.
- [49] MEHROTRA, A., JOHNSON, E. L., AND NEMHAUSER, G. L. An optimization based heuristic for political districting. *Management Science* 44, 8 (1998), 1100–1114. doi:10.1287/mnsc.44.8.1100.
- [50] MUYLDERMANS, L., CATTRYSSSE, D., VAN OUDHEUSDEN, D., AND LOTAN, T. Districting for salt spreading operations. *European Journal of Operational Research* 139, 3 (2002), 521–532. doi:10.1016/S0377-2217(01)00184-9.
- [51] NAVEH, B. JGraphT, 2016. <http://jgrapht.org/>.
- [52] NEMHAUSER, G. L., AND WOLSEY, L. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1988. doi:10.1002/9781118627372.
- [53] NÖLLENBURG, M., AND WOLFF, A. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE transactions on visualization and computer graphics* 17, 5 (2010), 626–641. doi:10.1109/TVCG.2010.81.
- [54] OPENSHAW, S. The modifiable areal unit problem. *Concepts and Techniques in Modern Geography* 38 (1984), 1–41.
- [55] PAPPS, K. L., AND NEWELL, J. O. Identifying functional labour market areas in new zealand: A reconnaissance study using travel-to-work data. *IZA Discussion Papers* 443, 1 (2002). <http://ftp.iza.org/dp443.pdf>.
- [56] PATAKI, G. Teaching integer programming formulations using the traveling salesman problem. *SIAM Review* 45, 1 (2003), 116–123. doi:10.1137/S00361445023685.
- [57] PUPPE, C., AND TASNÁDI, A. A computational approach to unbiased districting. *Mathematical and Computer Modelling* 48, 9–10 (2008), 1455–1460. doi:10.1016/j.mcm.2008.05.024.

- [58] RICCA, F., SCOZZARI, A., AND SIMEONE, B. Political districting: From classical models to recent approaches. *Annals of Operations Research* 204 (2013), 271–299. doi:10.1007/s10479-012-1267-2.
- [59] SESTER, M. Optimization approaches for generalization and data abstraction. *International Journal of Geographical Information Science* 19, 8–9 (2005), 871–897. doi:10.1080/13658810500161179.
- [60] SHIRABE, T. Classification of spatial properties for spatial allocation modeling. *GeoInformatica* 9, 3 (2005), 269–287. doi:10.1007/s10707-005-1285-1.
- [61] SHIRABE, T. A model of contiguity for spatial unit allocation. *Geographical Analysis* 37, 1 (2005), 2–16. doi:10.1111/j.1538-4632.2005.00605.x.
- [62] SHIRABE, T. Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design* 36, 6 (2009), 1053–1066. doi:10.1068/b34104.
- [63] SHIRABE, T., AND TOMLIN, C. D. Decomposing integer programming models for spatial allocation. In *Geographic Information Science*, M. J. Egenhofer and D. M. Mark, Eds. Springer, Berlin, Germany, 2002, ch. 21, pp. 300–312. doi:10.1007/3-540-45799-2_21.
- [64] SMART, M. W. Labour market areas: Uses and definition. *Progress in Planning* 2 (1974), 239–353. doi:10.1016/0305-9006(74)90008-7.
- [65] VAN ROY, T. J., AND WOLSEY, L. A. Solving mixed integer programming problems using automatic reformulation. *Operations Research* 35, 1 (1987), 45–57. doi:10.1287/opre.35.1.45.
- [66] WANG, Y., BUCHANAN, A., AND BUTENKO, S. On imposing connectivity constraints in integer programs. *Mathematical Programming* 166, 1 (Nov 2017), 241–271. doi:10.1007/s10107-017-1117-8.
- [67] WARE, J. M., JONES, C. B., AND THOMAS, N. Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographical Information Science* 17, 8 (2003), 743–769. doi:10.1080/13658810310001596085.
- [68] WEIBEL, R. Generalization of spatial data: Principles and selected algorithms. In *Algorithmic Foundations of Geographic Information Systems*. Springer, Berlin, Germany, 1997, pp. 99–152. doi:10.1007/3-540-63818-0_5.
- [69] WILLIAMS, J. C. A zero-one programming model for contiguous land acquisition. *Geographical Analysis* 34, 4 (2002), 330–349. doi:10.1111/j.1538-4632.2002.tb01093.x.
- [70] ZOLTNER, A. A., AND SINHA, P. Sales territory alignment: A review and model. *Management Science* 29, 11 (1983), 1237–1256. doi:10.1287/mnsc.29.11.1237.

Appendix A Minimum-weight vertex separators

For defining the constraints in Section 4.1.1 we use minimum-weight vertex separators. In this section we explain what these separators are and how to calculate them by using established minimum edge cut algorithms [13,21,26].



(a) A quadrilateral as an explanatory graph $G = (V, E)$. Here, we are looking for a minimum-weight (c, v) -separator.

(b) The transformation $\tilde{G} = (\tilde{V}, \tilde{A})$ of G ; elements in \tilde{V} and \tilde{A} induced by V are black, those in \tilde{A} induced by E are red.

(c) Blue arcs represent a maximum flow with corresponding weights from \tilde{c}_1 (■) to \tilde{v}_0 (□); since m is a sufficiently large constant, the maximum flow is $w(u) + w(z)$, limited by arcs induced by V (here: the dotted arcs \tilde{a}_u, \tilde{a}_z with weights $w(u), w(z)$ respectively).

(d) Figure (c) finally provides a minimum-weight edge cut $\{\tilde{a}_u, \tilde{a}_z\}$. Transferring this result back to G , we get the wanted minimum-weight (c, v) -separator $\{u, z\}$.

Figure 12: Computing a minimum-weight vertex separator.

Let $G = (V, E)$ be an undirected graph and $w: V \rightarrow \mathbb{R}^{\geq 0}$ a vertex-weight function. For $c, v \in V$, a *minimum-weight (c, v) -separator* $S \subseteq V \setminus \{c, v\}$ is a set of vertices with the follow-

ing property: Every path in G from c to v contains at least one vertex in S . Furthermore, any set with this property has at least the same weight as S with respect to w .

If $c = v$ or $\{c, v\} \in E$ then there exists no such vertex and $S = \emptyset$. In every other case we make use of minimum (c, v) -cut algorithms and apply them to a modified graph \tilde{G} .

In order to use edge cut algorithms we need to transform G into a directed graph $\tilde{G} = (\tilde{V}, \tilde{A})$ (see Figure 12(a) and (b)). Since only vertex weights are given and we want to use edge weights, we have to introduce two vertices $\tilde{v}_0, \tilde{v}_1 \in \tilde{V}$ and an arc $\tilde{a}_v = (\tilde{v}_0, \tilde{v}_1) \in \tilde{A}$ for every $v \in V$ with $\tilde{w}(\tilde{a}_v) = w(v)$, where $\tilde{w}: \tilde{A} \rightarrow \mathbb{R}^{\geq 0}$ is an edge-weight function. Furthermore, we replace every edge $e \in E$ with two arcs $\tilde{a}_0^e, \tilde{a}_1^e \in \tilde{A}$ going in opposite directions to maintain the structure of the graph. With defining $\tilde{w}(\tilde{a}_0) = \tilde{w}(\tilde{a}_1) = m \in \mathbb{R}$ with $m > \sum_{v \in V} w(v)$ we complete the transformation.

Let $\tilde{A}_V = \{\tilde{a}_v \mid v \in V\}$ be the set of all arcs induced by V (in the example, $\tilde{A}_V = \{\tilde{a}_c, \tilde{a}_u, \tilde{a}_v, \tilde{a}_z\}$, see Figure 12(b)). As, in general, $\tilde{A}_V \setminus \{\tilde{a}_c, \tilde{a}_v\}$ is a (c, v) -cut with weight $\sum_{a \in \tilde{A}_V \setminus \{\tilde{a}_u, \tilde{a}_v\}} \tilde{w}(a)$ less than $m = \tilde{w}(\tilde{a}_v)$ for every $\tilde{a} \in \tilde{A} \setminus \tilde{A}_V$ we do not only know a cut without any arcs induced by E , but also that any minimum-weight cut is a subset of $\tilde{A}_V \setminus \{\tilde{a}_u, \tilde{a}_v\}$. Hence we are able to conclude from the computed minimum-weight cut to a minimum-weight vertex separator.

Appendix B Algorithms

In the following section we present the algorithms corresponding to the methods of adding constraints to the model which are described in Section 4.2. For both Algorithms 1 and 2, we use—in addition to the adjacency graph $G = (V, E)$ —a matrix of ILP variables indexed in the same manner as throughout the article ($x[c][v]$ corresponds to $x_{c,v}$). As already mentioned, we only consider x -values exceeding a certain bound; in Section 4.2.3 we choose $\frac{1}{|V|}$ which we use in these algorithms.

With Algorithm 1 we are able to find cuts using minimum vertex separators as described in Sections 4.2.1 and 4.2.2. Some parts of the algorithm need explanatory remarks:

- To find a vertex separator in G , we use a *VertexSeparatorGenerator*. That is an object of the class *MinSourceSinkCut* of the library JGraphT [51] applied to an auxiliary graph as described in Section Appendix A.
- The boolean variable *hasAdditionalCuts* guarantees a sufficiently neat solution of the separation problem. The search for vertex separators (and consequently additional constraints) continues until a cut is added to the current LP relaxation, but at least as long as the value of the x -variable of the center is greater or equal $\frac{1}{|V|}$. *hasAdditionalCuts* is the output of Algorithm 1 and this output is an argument of Algorithm 2. In Algorithm 2, this boolean variable is used for the same purpose.
- The methods *getValue()* and *getValues()* return values of the solution of the LP relaxation to the currently examined branch-and-bound node; *getValue()* is applied to one variable and returns its value whereas *getValues()* is applied to an array of variables concerning a center c (i.e., $x[c]$) and, thus, actually returns a set of values $\{x_{c,v} \in [0; 1] \mid v \in V\}$.



Algorithm 1: addVertexSeparatorCuts

Data: Adjacency graph $G = (V, E)$, V sorted such that $v \geq w \Leftrightarrow x_{v,v} \geq x_{w,w}$ for $v, w \in V$, Variable[[]] x

Result: Cuts added by means of vertex separators (see Sections 4.2.1 and 4.2.2)

- 1 $Gen \leftarrow VertexSeparatorGenerator(G)$;
- 2 $c \leftarrow V.first()$;
- 3 $hasAdditionalCuts \leftarrow false$;
- 4 **while** $x[c][c].getValue() \geq \frac{1}{V.size()} \vee \neg hasAdditionalCuts$ **do**
- 5 $Gen.setWeights(V, x[c].getValues())$;
- 6 $U \leftarrow \left\{ v \in V \mid v \neq c \wedge (v, c) \notin E \wedge x_{c,v} \geq \frac{1}{V.size()} \right\}$;
- 7 **foreach** $u \in U$ **do**
- 8 $S \leftarrow Gen.getVertexSeparator(c, u)$;
- 9 **if** $\sum_{v \in S} x[c][v].getValue() < x[c][u].getValue()$ **then**
- 10 $addCut(\text{"}\sum_{v \in S} x[c][v] \geq x[c][u]\text{"})$;
- 11 $hasAdditionalCuts \leftarrow true$;
- 12 $C \leftarrow connectivityInspector((V \setminus S, E|_{V \setminus S}).getConnectedComponent(c))$;
- 13 **if** $\sum_{v \in C} w(v) < w_{\min}$ **then**
- 14 **foreach** $v \in C$ with $x[v][v].getValue() \geq \frac{1}{V.size()}$ **do**
- 15 **if** $\sum_{u \in S} x[v][u].getValue() < x[v][v].getValue()$ **then**
- 16 $addCut(\text{"}\sum_{i \in S} x[v][i] \geq x[v][v]\text{"})$;
- 17 $hasAdditionalCuts \leftarrow true$;
- 18 $c = V.next()$;
- 19 **return** $hasAdditionalCuts$;

- Given $G = (V, E)$ and a subset $U \subseteq V$, we define $E|_U$ as the set of edges restricted to the set of vertices U , that is

$$E|_U := \left\{ \{u, v\} \in E \mid u, v \in U \right\} = E \cap 2^U.$$

Consequently, $(U, E|_U)$ describes the subgraph of G induced by the vertex set U . In order to find the connected component containing c in this subgraph, we use an object of the class *ConnectivityInspector* which also is part of the library JGraphT.

Algorithm 2: addConnectedComponentsCuts

Data: Adjacency graph $G = (V, E)$, V sorted such that $v \geq w \Leftrightarrow x_{v,v} \geq x_{w,w}$ for $v, w \in V$, Variable[][] x , *hasAdditionalCuts*

Result: Cuts added based on connected components (see Section 4.2.4)

```

1  $c \leftarrow V.first();$ 
2 while  $x[c][c].getValue() \geq \frac{1}{V.size()} \vee \neg hasAdditionalCuts$  do
3    $V' \leftarrow \left\{ v \in V \mid x[c][v] \geq \frac{1}{V.size()} \right\};$ 
4    $G' \leftarrow (V', E|_{V'});$ 
5   foreach ConnectedComponent  $U$  in  $G'$  with  $c \notin U$  do
6      $B_U \leftarrow \{w \in V \setminus U \mid \exists u \in U : \{u, w\} \in E\};$ 
7     foreach  $i \in C$  do
8       if  $\sum_{j \in B_U} x[c][j].getValue() < x[c][i].getValue()$  then
9          $addCut(\text{"}\sum_{j \in B_U} x[c][j] \geq x[c][i]\text{"});$ 
10         $hasAdditionalCuts \leftarrow true;$ 

```

Algorithm 2 allows us to add cuts using connected components of the adjacency graph as explained in Section 4.2.4. All the methods and sets (except $B_U \subseteq V$) described here are known from Algorithm 1. A further description of B_U , the set of vertices in V adjacent to $U \subseteq V$, can be found in Section 4.2.4.