# Adjacency-Preserving Spatial Treemaps

Kevin Buchin[1], David Eppstein[2], Maarten Löffler[2],
Martin Nöllenburg[3], and Rodrigo I. Silveira[4]

[1] Dept. of Mathematics and Computer Science, TU Eindhoven.
[2] Dept. of Computer Science, University of California, Irvine.
[3] Institute of Theoretical Informatics, Karlsruhe Institute of Technology.
[4] Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya.

**Abstract.** Rectangular layouts, subdivisions of an outer rectangle into smaller rectangles, have many applications in visualizing spatial information, for instance in rectangular cartograms in which the rectangles represent geographic or political regions. A *spatial treemap* is a rectangular layout with a hierarchical structure: the outer rectangle is subdivided into rectangles that are in turn subdivided into smaller rectangles. We describe algorithms for transforming a rectangular layout that does not have this hierarchical structure, together with a clustering of the rectangles of the layout, into a spatial treemap that respects the clustering and also respects to the extent possible the adjacencies of the input layout.

## 1 Introduction

Spatial treemaps are an effective technique to visualize two-dimensional hierarchical information. They display hierarchical data by using nested rectangles in a space-filling layout. Each rectangle represents a geometric or geographic region, which in turn can be subdivided recursively into smaller regions. On lower levels of the recursion, rectangles can also be subdivided based on non-spatial attributes. Typically, at the lowest level some attribute of interest of the region is summarized by using properties like area or color. Treemaps were originally proposed to represent one-dimensional information in two dimensions [14]. However, they are well suited to represent spatial—two-dimensional—data because the containment metaphor of the nested rectangles has a natural geographic meaning, and two-dimensional data makes an efficient use of space [18].

Spatial treemaps are closely related to rectangular cartograms [13]: distorted maps where each region is represented by a rectangle whose area corresponds to a numerical attribute such as population. Rectangular cartograms can be seen as spatial treemaps with only one level; multi-level spatial treemaps in which every rectangle corresponds to a region are also known as *rectangular hierarchical cartograms* [15, 16]. Spatial treemaps and rectangular cartograms have in common that it is essential to preserve the recognizability of the regions shown [17]. Most previous work on spatial treemaps reflects this by focusing on the preservation of distances between the rectangular regions and their geographic counterparts (that is, they minimize the displacement of the regions). However, often small displacement does not imply recognizability (swapping the position of two small neighboring countries can result in small displacement, but a big loss of recognizability). In the case of cartograms, most emphasis has been put on preserving adjacencies between the geographic regions. It has also been shown that while preserving the topology it is possible to keep the displacement error small [4, 17].

In this paper we are interested in constructing high-quality spatial treemaps by prioritizing the preservation of topology, following a principle already used for rectangular cartograms. Previous work on treemaps has recognized that preserving neighborhood relationships and relative positions between the regions were important criteria [8,12,18], but we are not aware of treemap algorithms that put the emphasis on preserving topology.

The importance of preserving adjacencies in spatial treemaps can be appreciated by viewing a concrete example. Figure 1, from [15], shows a spatial treemap of property transactions in London between 2000 and 2008, with two levels formed by the boroughs and wards of London and colors representing average prices. To see whether housing prices of neighboring wards are correlated, it is important to preserve adjacencies: otherwise it is easy to draw incorrect conclusions, like seeing clusters that do not actually exist, or missing existing ones.

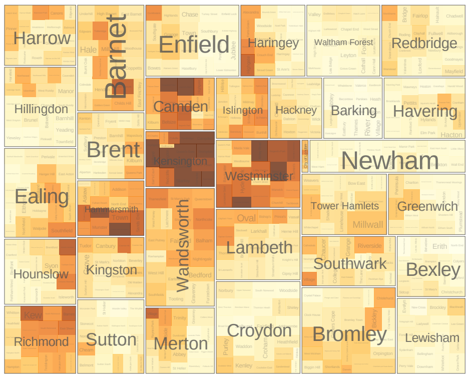


**Fig. 1.** A 2-level spatial treemap from [15]; used with permission.

Preserving topology in spatial treemaps poses different challenges than in (non-hierarchical) rectangular cartograms. Topology-preserving rectangular cartograms exist under very mild conditions and can be constructed efficiently [4, 17]. As we show in this paper, this is not the case when a hierarchy is added to the picture.

In this paper we consider the following setting: the input is a hierarchical rectangular subdivision with two levels. We consider only two levels due to the complexity of the general *m*-level case. However, the two-level case is interesting on its own, and applications that use only two-level data have recently appeared [15].

Furthermore, we adopt a 2-phase approach for building spatial treemaps. In the first phase, a base rectangular cartogram is produced from the original geographic regions. This can be done with one of the many algorithms for rectangular cartograms [4]. The result will contain all the bottom-level regions as rectangles, but the top-level regions will not be rectangular yet, thus will not represent the hierarchical structure. In the second phase, we convert the base cartogram into a treemap by making the top-level regions rectangles. It is at this stage that we intend to preserve the topology of the base cartogram as much as possible, and where our algorithms come in. See Figure 2 for an example.

The advantage of this 2-phase approach is that it allows for customization and user interaction. Interactive exploration of the data is essential when visualizing large amounts of data. The freedom to use an arbitrary rectangular layout algorithm in the first phase of the construction allows the user to prioritize the adjacencies that he or she considers most essential. In the second phase, our algorithm will produce a treemap that will try to preserve as many as the adjacencies in the base cartogram as possible.

In addition, we go one step further and consider preserving the *orientations* of the adjacencies in the base cartogram (that is, whether two neighboring regions share a vertical or horizontal edge, and which one is on which side). This additional constraint

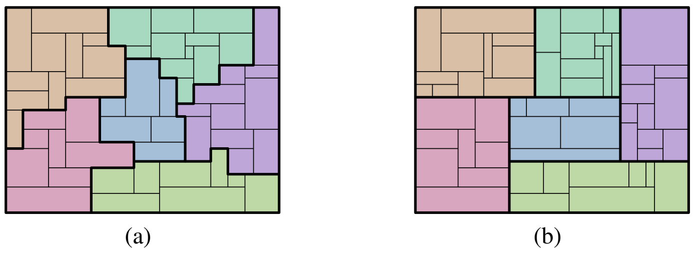


**Fig. 2.** (a) An example input: a full layout of the bottom level, but the regions at a higher level in the hierarchy are not rectangles. (b) The desired output: another layout, in which as many lower-level adjacencies as possible have been kept while reshaping the regions at a higher level into rectangles.

is justified by the fact that the regions represent geographic or political regions, and relative positions between regions are an important factor when visualizing this type of data [4, 17]. The preservation of orientations has been studied for cartograms [5], but to our knowledge, this is the first time they are considered for spatial treemaps.

We can distinguish three types of adjacency-relations: (i) top-level adjacencies, (ii) internal bottom-level adjacencies (adjacencies between two rectangles that belong to the same top-level region), and (iii) external bottom-level adjacencies (adjacencies between two rectangles that belong to different top-level regions). As we argue in the next section, we can always preserve all adjacencies of types (i) and (ii) under a mild assumption, hence the objective of our algorithms is to construct treemaps that preserve as many adjacencies of type (iii) as possible. We consider several variants of the problem, based on whether the orientations of the adjacencies have to be preserved, and whether the top-level layout is given in advance. In order to give efficient algorithms, we restrict ourselves to top-level regions that are orthogonally convex. This is a technical limitation that seems difficult to overcome, but that we expect does not limit the applicability of our results too much: our algorithms should still be useful for many practical instances, for example, by subdividing non-convex regions into few convex pieces.

**Results** In the most constrained case in which adjacencies and their orientations need to be preserved and the top-level layout is given, we solve the problem in $O(n)$ time, where $n$ is the total number of rectangles. The case in which the global layout is not fixed is much more challenging: it takes a combination of several techniques based on regular edge labelings to obtain an algorithm that solves the problem optimally in $O(k^4 \log k + n)$ time, for $k$ the number of top-level regions; we expect $k$ to be much smaller than $n$. Finally, we prove that the case in which the orientations of adjacencies do not need to be preserved is NP-hard; we give worst-case bounds and an approximation algorithm.

## 2 Preliminaries

**Rectangles and Subdivisions** All geometric objects like rectangles and polygons in this paper are defined as rectilinear (axis-aligned) objects in the Euclidean plane $\mathbb{R}^2$. A set of rectangles $\mathcal{R}$ is called a *rectangle complex* if the interiors of none of the rectangles overlap, and each pair of rectangles is either completely disjoint or shares part of an edge; no two rectangles may meet in a single point. Each rectangle of a rectangle complex is a *cell* of that complex. We represent rectangle complexes using a structure that has bidirectional pointers between neighboring cells. Let $\mathcal{R}$ be a rectangle complex. The *boundary* of $\mathcal{R}$ is the boundary of the the union of the rectangles in $\mathcal{R}$. $\mathcal{R}$ is *simple* if its boundary is a simple polygon, i.e., it is connected and has no holes. We say that $\mathcal{R}$ is *convex* if its boundary is orthogonally convex, i.e., the intersection of any horizontal or vertical line with $\mathcal{R}$ is either empty or a single line segment. We say that $\mathcal{R}$ is *rectangular* if its boundary is a rectangle. Let $\mathcal{R}'$ be another rectangle complex. We say that $\mathcal{R}'$ is an *extension* of $\mathcal{R}$ if there is a bijective mapping between the cells in $\mathcal{R}$ and $\mathcal{R}'$ that preserves the adjacencies and their orientations. Note that $\mathcal{R}'$ could have adjacencies not present in $\mathcal{R}$ though. We say that $\mathcal{R}'$ is a *simple extension* of $\mathcal{R}$ if $\mathcal{R}$ is not simple but $\mathcal{R}'$ is; similarly we may call it a *convex extension* or a *rectangular extension*. Every rectangle complex has a rectangular extension (proof in full version [3]).

We define $\mathbb{D} = \{\text{left}, \text{right}, \text{top}, \text{bottom}\}$ to be the set of the four cardinal directions. For a direction $d \in \mathbb{D}$ we use the notation $-d$ to refer to the direction opposite from $d$. We define an object $O \subset \mathbb{R}^2$ to be *extreme* in direction $d$ with respect to a rectangle complex $\mathcal{R}$ if there is a point in $O$ that is at least as far in direction $d$ as any point in $\mathcal{R}$. Let $R \in \mathcal{R}$ be a cell, and $d \in \mathbb{D}$ a direction. We say $R$ is *$d$-extensible* if there exists a rectangular extension $\mathcal{R}'$ of $\mathcal{R}$ in which $R$ is extreme in direction $d$ with respect to $\mathcal{R}'$ (or in other words, if its $d$-side is part of the boundary of $\mathcal{R}'$). A set of simple rectangle complexes $\mathcal{L}$ is called a (rectilinear) *layout* if the boundary of the union of all complexes is a rectangle, the interiors of the complexes are disjoint, and no point in $\mathcal{L}$ belongs to more than three cells. If all complexes are rectangular we say that $\mathcal{L}$ is a *rectangular layout*. We call the rectangle bounding $\mathcal{L}$ the *root box*. Let $\mathcal{L}$ be a rectilinear layout. We define the *global layout* $\mathcal{L}'$ of $\mathcal{L}$ as the subdivision of the root box of $\mathcal{L}$, in which the (*global*) regions are defined by the boundaries of the complexes in $\mathcal{L}$. We say $\mathcal{L}'$ is *rectangular* if all regions in $\mathcal{L}'$ are rectangles.

**Dual Graphs of Rectangle Complexes** The *dual graph* of a rectangular complex is an embedded planar graph with one vertex for every rectangle in the complex, and an edge between two vertices if the corresponding rectangles touch (have overlapping edge pieces). The *extended dual graph* of a rectangular complex with a rectangular boundary has four additional vertices for the four sides of the rectangle, and an edge between a normal vertex and an additional vertex if the corresponding rectangle touches the corresponding side of the bounding box. We will be using dual graphs of the whole rectangular layout, of individual complexes, and of the global layout (ignoring the bottom level subdivision); Figure 3 shows some examples. Extended dual graphs of rectangular rectangle complexes are fully triangulated (except for the outer face which is a quadrilateral), and the graphs that can arise in this way are characterized by the following lemma [9, 11, 17]:

**Lemma 1.** *A triangulated plane graph $G$ with a quadrilateral outer face is the dual graph of a rectangular rectangle complex if and only if $G$ has no separating triangles.*

Now, consider the three types of adjacencies we wish to preserve: 1) (top-level) adjacencies between global regions, 2) internal (bottom-level) adjacencies between the

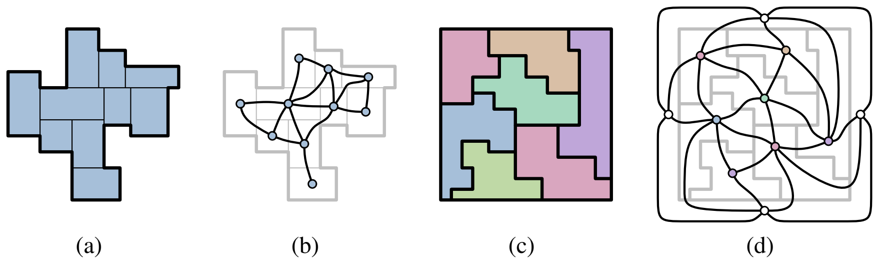


**Fig. 3.** (a) A bottom level rectangle complex. (b) The dual graph of the complex. (c) A global layout. (d) The extended dual graph of the global layout.

cells in one rectangle complex, and 3) external (bottom-level) adjacencies between cells of adjacent rectangle complexes.

**Observation 1.** *It is always possible to keep all internal bottom-level adjacencies.*

**Observation 2.** *It is possible to keep all top-level adjacencies if and only if the extended dual graph of the global input layout has no separating triangles.*

Observation 1 is proven in the full version [3], and Observation 2 follows from Lemma 1 since the extended dual graph of the global regions is fully triangulated.

From now on we assume that the dual graph of the global regions has no separating triangles, and we will preserve all adjacencies of types 1 and 2. Unfortunately, it is not always possible to keep adjacencies of type 3—see Figure 4—and for every adjacency of type 3 that we fail to preserve, another adjacency that was not present in the original layout will appear. Therefore, our aim is to preserve as many of these adjacencies as possible.

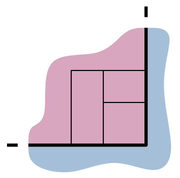


**Fig. 4.** Not all external adjacencies can be kept.

## 3 Preserving orientations

We begin studying the version of the problem where all internal adjacencies have to be preserved respecting their original orientations. Additionally, we want to maximize the number of preserved and correctly oriented (bottom-level) external adjacencies. We consider two scenarios: first we assume that the global layout is part of the input, and then we study the case in which we optimize over all global layouts. The former situation is particularly interesting for GIS applications, in which the user specifies a certain global layout that needs to be filled with the bottom-level cells. If, however, the bottom-level adjacencies are more important, then optimizing over global layouts allows to preserve more external adjacencies.

### 3.1 Given the global layout

In this section we are given, in addition to the initial two-level subdivision $\mathcal{L}$, a global target layout $\mathcal{L}'$. The goal is to find a two-level treemap that preserves all oriented

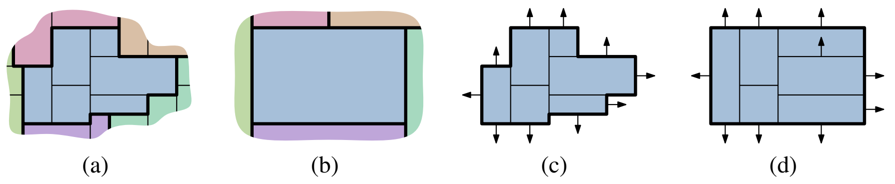


**Fig. 5.** (a) A region in the input. (b) The same region in the given global layout. (c) Edges of rectangles that want to become part of a boundary have been marked with arrows. Note that one rectangle wants to become part of the top boundary but can't, because it is not extensible in that direction. (d) All arrows that aren't blocked can be made happy.

bottom-level internal adjacencies and that maximizes the number of preserved oriented bottom-level external adjacencies in the output.

First observe that in the rectangular output layout any two neighboring global regions have a single orientation for their adjacency. Hence we can only keep those bottom-level external adjacencies that have the same orientation in the input as their corresponding global regions have in the output layout. Secondly, consider a rectangle $R$ in a complex $\mathcal{R}$, and a rectangle $B$ in another complex $\mathcal{B}$. Observe that if $R$ and $B$ are adjacent in the input, for example with $R$ to the left of $B$, then their adjacency can be preserved only if $R$ is right-extensible in $\mathcal{R}$ and $B$ is left-extensible in $\mathcal{B}$.

The main result in this section is that the previous two conditions are enough to describe all adjacencies that cannot be preserved, whereas all the other ones can be kept. Furthermore, we will show how to decide extensibility for convex complexes, and how to construct a final solution that preserves all possible adjacencies, leading to an algorithm for the optimal solution.

Recall that we assume all regions are orthogonally convex. Consider each rectangle complex of $\mathcal{L}$ separately. Since we know the required global layout and since all cells externally adjacent to our region are consecutive along its boundary, we can immediately determine the cells on each of the four sides of the output region (see Figure 5). The reason is that for a rectangle $R$ that is exterior to its region $\mathcal{R}$, and that is adjacent to another rectangle $B \in \mathcal{B}$, their adjacency is relevant only if $\mathcal{R}$ and $\mathcal{B}$ are adjacent with the same orientation in the global layout. We can easily categorize the extensible rectangles of a convex rectangle complex. For the proof of the following lemma and other proofs in this section we refer to the full version [3].

**Lemma 2.** *Let $\mathcal{R}$ be a convex rectangle complex, let $R \in \mathcal{R}$ be a rectangle, and $d \in \mathbb{D}$ a direction. $R$ is $d$-extensible if and only if there is no rectangle $R' \in \mathcal{R}$ directly adjacent to $R$ on the $d$-side of $R$.*

Unfortunately, though, we cannot extend all extensible rectangles at the same time. However, we show that we can actually extend all those rectangles that we want to extend for an optimal solution.

We call a rectangle of a certain complex belonging to a global region *engaged* if it wants to be adjacent to a rectangle of another global region, and the direction of their desired adjacency is the same as the direction of the adjacency between these two regions in the global layout. We say it is $d$-engaged if this direction is $d \in \mathbb{D}$.

Therefore, the rectangles that we want to extend are exactly those that are $d$-extensible and $d$-engaged, since they are the only ones that help preserve bottom-level exterior adjacencies. It turns out that extending all these rectangles is possible, because the engaged rectangles of $\mathcal{R}$ have a special property:

**Lemma 3.** *If we walk around the boundary of a region $\mathcal{R}$, we encounter all $d$-engaged rectangles consecutively.*

This property of $d$-engaged rectangles is useful due to the following fact.

**Lemma 4.** *Let $\mathcal{R}$ be a convex rectangle complex composed of $r$ rectangles, and let $S$ be a subset of the extensible and engaged rectangles in $\mathcal{R}$ with the property that if we order them according to a clockwise walk along the boundary of $\mathcal{R}$, all $d$-extensible rectangles in $S$ are encountered consecutively for each $d \in \mathbb{D}$ and in the correct clockwise order. We can compute, in $O(r)$ time, a rectangular extension $\mathcal{R}'$ of $\mathcal{R}$ in which all $d$-extensible rectangles in $S$ are extreme in direction $d$, for all $d \in \mathbb{D}$.*

Therefore, the engaged and extensible rectangles form a subset of rectangles for which Lemma 4 holds, thus by using the lemma we can find a rectangular extension where all extensible and engaged rectangles are extreme in the appropriate direction.

Then we can apply this idea to each region. Now we still have to match up the adjacencies in an optimal way, that is, preserving as many adjacencies from the input as possible. This can be done by matching horizontal and vertical adjacencies independently. It is always possible to get all the external bottom-level adjacencies that need to be preserved. This can be seen as follows. We process first all horizontal adjacencies. Consider a complete stretch of horizontal boundary in the global layout. Then the position and length of the boundary of each region adjacent to that boundary are fixed, from the global layout. The only freedom left is in the $x$-coordinates of the vertical edges of the rectangles that form part of that boundary (except for the leftmost and rightmost borders of each region, which are also fixed). Since the adjacencies that want to be preserved are part of the input, it is always possible to set the $x$-coordinates in order to fulfill them all. The same can be done with all horizontal boundaries. The vertical boundaries are independent, thus can be processed in exactly the same way. This yields the main theorem in this subsection.

**Theorem 1.** *Let $\mathcal{T}$ be a 2-level treemap, where $n$ is the number of cells in the bottom level, and where all global regions are orthogonally convex. For a given global target layout $\mathcal{L}$, we can find, in $O(n)$ time, a rectangular layout of $\mathcal{T}$ that respects $\mathcal{L}$, preserves all oriented internal bottom-level adjacencies, and preserves as many oriented external bottom-level adjacencies as possible.*

## 3.2 Unconstrained global layout

In this section the global target layout of the rectangle complexes is not given, i.e., we are given a rectilinear input layout and need to find a rectangular output layout preserving all adjacencies of the rectangle complexes and preserving a maximum number of adjacencies of the cells of different complexes.

We can represent a particular rectangular global layout $\mathcal{L}$ as a *regular edge labeling* [10] of the dual graph $G(\mathcal{L})$ of the global layout. Let $G(\mathcal{L})$ be the extended dual graph of $\mathcal{L}$. Then $\mathcal{L}$ induces an edge labeling as follows: an edge corresponding to a joint vertical (horizontal) boundary of two rectangular complexes is colored blue (red). Furthermore, blue edges are directed from left to right and red edges from bottom to top. Clearly, the edge labeling obtained from $\mathcal{L}$ in this way satisfies that around each inner vertex $v$ of $G(\mathcal{L})$ the incident edges with the same color and the same direction form contiguous blocks around $v$. The edges incident to one of the external vertices $\{l,t,r,b\}$ all have the same label. Such an edge labeling is called *regular* [10]. Each regular edge labeling of the extended dual graph $G(\mathcal{L})$ defines an equivalence class of global layouts.

In order to represent the family of all possible rectangular global layouts we apply a technique described by Eppstein et al. [5,6]. Let $\mathcal{L}$ be the rectilinear global input layout and let $G(\mathcal{L})$ be its extended dual graph. The first step is to decompose $G(\mathcal{L})$ by its separating 4-cycles into minors called *separation components* with the property that they do not have non-trivial separating 4-cycles any more, i.e., 4-cycles with more than a single vertex in the inner part of the cycle. If $C$ is a separating 4-cycle the interior separation component consists of $C$ and the subgraph induced by the vertices interior to $C$. The outer separation component is obtained by replacing all vertices in the interior of $C$ by a single vertex connected to each vertex of $C$. This decomposition can be obtained in linear time [6]. We can then treat each component in the decomposition independently and finally construct an optimal rectangular global layout from the optimal solutions of its descendants in the decomposition tree. So let's consider a single component of the decomposition, which by construction has no non-trivial separating 4-cycles.

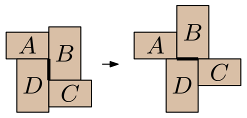
**Preprocessing of the bottom level** We start with a preprocessing step to compute the number of realizable external bottom-level adjacencies for pairs of adjacent global regions. This allows us to ignore the bottom-level cells in later steps and to focus on the global layout and orientations of global adjacencies.

Let $\mathcal{L}$ be a global layout, let $\mathcal{R}$ and $\mathcal{S}$ be two adjacent rectangle complexes in $\mathcal{L}$, and let $d \in \mathbb{D}$ be an orientation. Then we define $\omega(\mathcal{R},\mathcal{S},d)$ to be the total number of adjacencies between $d$-engaged and $d$-extensible rectangles in $\mathcal{R}$ and $-d$-engaged and $-d$-extensible rectangles in $\mathcal{S}$. By Lemma 4 there is a rectangular layout of $\mathcal{R}$ and $\mathcal{S}$ with exactly $\omega(\mathcal{R},\mathcal{S},d)$ external bottom-level adjacencies between $\mathcal{R}$ and $\mathcal{S}$.

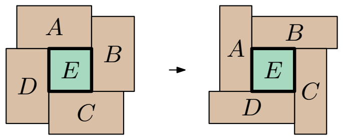
We show the following (perhaps surprising) lemma:

**Lemma 5.** *For any pair $\mathcal{L}$ and $\mathcal{L}'$ of global layouts and any pair $\mathcal{R}$ and $\mathcal{S}$ of rectangular rectangle complexes, whose adjacency direction with respect to $\mathcal{R}$ is $d$ in $\mathcal{L}$ and $d'$ in $\mathcal{L}'$ the number of external bottom level adjacencies between $\mathcal{R}$ and $\mathcal{S}$ in any optimal solution for $\mathcal{L}'$ differs by $\omega(\mathcal{R},\mathcal{S},d') - \omega(\mathcal{R},\mathcal{S},d)$ from $\mathcal{L}$. For adjacent rectangle complexes whose adjacency direction is the same in both global layouts the number of adjacencies in any optimal solution remains the same.*

This basically means we can consider changes of adjacency directions locally and independent from the rest of the layout. Furthermore, since the values $\omega(\mathcal{R},\mathcal{S},d)$ are



(a) edge flip *BD*



(b) vertex flip *E*

**Fig. 6.** Flip operations

directly obtained from counting the numbers of $d$-extensible and $d$-engaged rectangles in $\mathcal{R}$ (or $-d$-extensible and $-d$-engaged rectangles in $\mathcal{S}$) we get the next lemma.

**Lemma 6.** *We can compute all values $\omega(\mathcal{R},\mathcal{S},d)$ in $O(n)$ total time.*

**Optimizing in a graph without separating 4-cycles** Here we will prove the following:

**Theorem 2.** *Let $G$ be an embedded triangulated planar graph with $k'$ vertices without separating 3-cycles and without non-trivial separating 4-cycles, except for the outer face which consists of exactly four vertices. Furthermore, let a weight $\omega(e,d)$ be assigned to every edge $e$ in $G$ and every orientation $d$ in $\mathbb{D}$. Then we can find a rectangular subdivision of which $G$ is the extended dual that maximizes the total weight of the directed adjacencies in $O(k'^4 \log k')$ time.*

In order to optimize over all rectangular subdivisions with the same extended dual graph we make use of the representation of these subdivisions as elements in a distributive lattice or, equivalently, as closures in a partial order induced by this lattice [5,6]. There are two *moves* or *flips* by which we can transform one rectangular layout (or its regular edge labeling) into another one, *edge flips* and *vertex flips* (Figure 6). They form a graph where each equivalence class of rectangular layouts is a vertex and two vertices are connected by an edge if they are transformable into each other by a single move, with the edge directed toward the more counterclockwise layout with respect to this move. This graph is acyclic and its reachability ordering is a distributive lattice [7]. It has a minimal (maximal) element that is obtained by repeatedly performing clockwise (counterclockwise) moves.

By Birkhoff's representation theorem [2] each element in this lattice is in one-to-one correspondence to a partition of a partial order $\mathcal{P}$ into an upward-closed set $U$ and a downward-closed set $L$. The elements in $\mathcal{P}$ are pairs $(x,i)$, where $x$ is a flippable item, i.e., either the edge of an edge flip or the vertex of a vertex flip [5,6]. The integer $i$ is the so-called flipping number $f_x(\mathcal{L})$ of $x$ in a particular layout $\mathcal{L}$, i.e., the well-defined number of times flip $x$ is performed counterclockwise on any path from the minimal element $\mathcal{L}_{\min}$ to $\mathcal{L}$ in the distributive lattice. An element $(x,i)$ is smaller than another element $(y,j)$ in this order if $y$ cannot be flipped for the $j$-th time before $x$ is flipped for the $i$-th time. For each upward- and downward-closed partition $U$ and $L$, the corresponding layout can be reconstructed by performing all flips in the lower set $L$. $\mathcal{P}$ has $O(k'^2)$ vertices and edges and can be constructed in $O(k'^2)$ time [5,6]. The construction starts with an arbitrary layout, performs a sequence of clockwise moves until we reach $\mathcal{L}_{\min}$, and

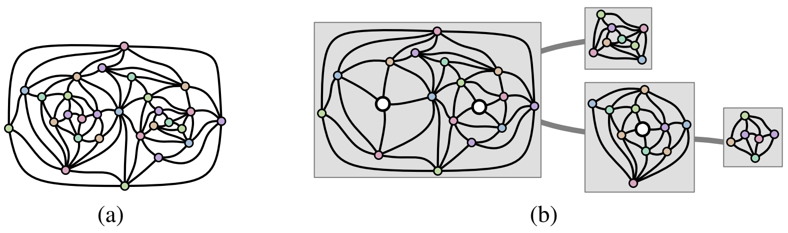


**Fig. 7.** (a) A graph with non-trivial separating 4-cycles. Note that some 4-cycles intersect each other. (b) A possible decomposition tree of 4-cycle-free graphs (root on the left).

from there performs a sequence of counterclockwise moves until we reach the maximal element. During this last process we count how often each element is flipped, which determines all pairs $(x,i)$ of $\mathcal{P}$. Since each flip $(x,i)$ affects only those flippable items that belong to the same triangle as $x$, we can initialize a queue of possible flips, and iteratively extract the next flip and add the new flips to the queue in total time $O(k'^2)$. In order to create the edges in $\mathcal{P}$ we again use the fact that a flip $(x,i)$ depends only on flips $(x',i')$, where $x'$ belongs to the same triangle as $x$ and $i'$ differs by at most 1 from $i$. The actual dependencies can be obtained from their states in $\mathcal{L}_{\min}$.

Next, we assign weights to the nodes in $\mathcal{P}$. Let $\mathcal{L}_{\min}$ be the layout that is minimal in the distributive lattice, i.e., the layout where no more clockwise flips are possible. For an edge-flip node $(e,i)$ let $\mathcal{R}$ and $\mathcal{S}$ be the two rectangle complexes adjacent across $e$. Then the weight $\omega(e,i)$ is obtained as follows. Starting with the adjacency direction between $\mathcal{R}$ and $\mathcal{S}$ in $\mathcal{L}_{\min}$ we cycle $i$ times through the set $\mathbb{D}$ in counterclockwise fashion. Let $d$ be the $i$-th direction and $d'$ the $(i+1)$-th direction. Then $\omega(e,i) = \omega(e,d') = \omega(\mathcal{R},\mathcal{S},d') - \omega(\mathcal{R},\mathcal{S},d)$. For a vertex-flip node $(v,i)$ let $\mathcal{R}$ be the degree-4 rectangle complex surrounded by the four complexes $\mathcal{S}_1,\ldots,\mathcal{S}_4$. We again determine the adjacency directions between $\mathcal{R}$ and $\mathcal{S}_1,\ldots,\mathcal{S}_4$ in $\mathcal{L}_{\min}$ and cycle $i$ times through $\mathbb{D}$ to obtain the $i$-th directions $d_1,\ldots,d_4$ as well as the $(i+1)$-th directions $d'_1,\ldots,d'_4$. Then $\omega(v,i) = \sum_{j=1}^{4} \omega(\mathcal{R},\mathcal{S}_j,d'_j) - \omega(\mathcal{R},\mathcal{S}_j,d_j)$. Equivalently, if the four edges incident to $v$ are $e_1,\ldots,e_4$, we have $\omega(v,i) = \sum_{j=1}^{4} \omega(e_j,d'_j)$.

Finally, we compute a maximum-weight closure of $\mathcal{P}$ using a max-flow algorithm [1, Chapter 19.2], which will take $O(k'^4 \log k')$ time for a graph with $O(k'^2)$ nodes.

**Optimizing in General Graphs** In this section, we show how to remove the restriction that the graph should have no separating 4-cycles. We do this by decomposing the graph $G$ by its separating 4-cycles and solving the subproblems in a bottom-up fashion.

**Lemma 7 (Eppstein et al. [6]).** *Given a plane graph $G$ with $k$ vertices, there exists a collection $\mathcal{C}$ of separating 4-cycles in $G$ that decomposes $G$ into separation components that do not contain separating 4-cycles any more. Such a collection $\mathcal{C}$ and the decomposition can be computed in $O(k)$ time.*

These cycles naturally subdivide $G$ into a tree of subgraphs, which we will denote as $T_G$. Still following [6], we add an extra artificial vertex inside each 4-cycle, which corresponds to filling the void in the subdivision after removing all rectangles inside by a single rectangle. Figure 7 shows an example of a graph $G$ and a corresponding tree $T_G$.

Now, all nodes of $T_G$ have an associated graph without separating 4-cycles on which we can apply Theorem 2. The only thing left to do is assign the correct weights to the edges of these graphs. For a given node $\nu$ of $T_G$, let $G_\nu$ be the subgraph of $G$ associated to $\nu$ (with potentially extra vertices inside its 4-cycles).

For every leave $\nu$ of $T_G$, we assign weights to the internal edges of $G_\nu$ by simply setting $\omega(e,d) = \omega(\mathcal{R},\mathcal{S},d)$ if $e$ separates $\mathcal{R}$ and $\mathcal{S}$ in the global layout $\mathcal{L}$. For the external edges of $G_\nu$ (the edges that are incident to one of the "corner" vertices of the outer face), we fix the orientations in the four possible ways, leading to four different problems. We apply Theorem 2 four times, once for each orientation. We store the resulting solution values as well as the corresponding optimal layouts at $\nu$ in $T_G$.

Now, in bottom-up order, for each internal node $\nu$ in $T_G$, we proceed in a similar way with one important change: for each child $\mu$ of $\nu$, we first look up the four optimal layouts of $\mu$ and incorporate them in the weights of the four edges incident to the single extra vertex that replaced $G_\mu$ in $G_\nu$. Since these four edges must necessarily have four different orientations, their states are linked, and it does not matter how we distribute the weight over them; we can simply set the weight of three of these edges to 0 and the remaining one to the solution of the appropriately oriented subproblem. The weights of the remaining edges are derived from $\mathcal{L}$ as before, and again we fix the orientations of the external edges of $G_\nu$ in four different ways and apply Theorem 2 to each of them. We again store the resulting four optimal values and the corresponding layouts at $\nu$, in which we insert the correctly oriented subsolutions for all children $\mu$ of $\nu$.

This whole process takes $O(k^4 \log k)$ time in the worst case. Finally, since weights are expressed as differences with respect to the minimal layout $\mathcal{L}_{\min}$ we compute the value of $\mathcal{L}_{\min}$ and add the offset computed as the optimal solution to get the actual value of the globally optimal solution. This takes $O(n)$ time.

**Theorem 3.** *Let $\mathcal{T}$ be a 2-level treemap, such that the extended dual graph $G$ of the global layout has no separating 3-cycles. Let $n$ be the number of cells in the bottom level and $k$ the number of regions in the top level. Then we can find a rectangular subdivision that preserves all oriented internal bottom-level adjacencies, and preserves as many oriented external bottom-level adjacencies as possible in $O(k^4 \log k + n)$ time.*

## 4 Without preserving orientations

In this section we do not need to preserve orientations of internal adjacencies. The global regions are convex and we assume that the global layout is given. However, maximizing the number of preserved external adjacencies in this case is NP-hard even if we only have two top-level regions. For two top-level regions we give a $1/3$-approximation algorithm for this problem. Furthermore, this algorithm preserves at least $1/9$ of the external adjacencies. We also show that we sometimes cannot keep more than $1/4$ of the adjacencies. The algorithm extends to more than two regions. In this case it is a $1/6$-approximation and at least $1/18$ of the adjacencies are kept. Due to space restrictions, we defer all details of these results to the full version of this paper [3].

## Acknowledgements

This research was initiated at MARC 2009. We would like to thank Jo Wood for proposing this problem, and all participants for sharing their thoughts on this subject.

D. E. is supported by the National Science Foundation under grant 0830403. D. E. and M. L. are supported by the U.S. Office of Naval Research under grant N00014-08-1-1015. M. N. is supported by the German Research Foundation (DFG) under grant NO 899/1-1. R. I. S. is supported by the Netherlands Organisation for Scientific Research (NWO).

## References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.
2. G. Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, 1937.
3. K. Buchin, D. Eppstein, M. Löffler, M. Nöllenburg, and R. I. Silveira. Adjacency-preserving spatial treemaps. Arxiv report, arXiv:1105.0398 [cs.CG], May 2011.
4. K. Buchin, B. Speckmann, and S. Verdonschot. Optimizing regular edge labelings. In *Proc. GD*, volume 6502 of *LNCS*, pages 117–128. Springer, 2011.
5. D. Eppstein and E. Mumford. Orientation-constrained rectangular layouts. In *Proc. WADS*, pages 266–277, 2009.
6. D. Eppstein, E. Mumford, B. Speckmann, and K. Verbeek. Area-universal rectangular layouts. In *Proc. SoCG*, pages 267–276, 2009.
7. É. Fusy. Transversal structures on triangulations: A combinatorial study and straight-line drawings. *Discrete Mathematics*, 309(8):1870–1894, 2009.
8. R. Heilmann, D. A. Keim, C. Panse, and M. Sips. Recmap: Rectangular map approximations. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 33–40, Washington, DC, USA, 2004. IEEE Computer Society.
9. G. Kant and X. He. Two algorithms for finding rectangular duals of planar graphs. In *Proc. 19th Internat. Workshop Graph-Theoret. Concepts Comput. Sci.*, volume 790 of *Lecture Notes Comput. Sci.*, pages 396–410. Springer-Verlag, 1993.
10. G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theoretical Computer Science*, 172(1–2):175–193, 1997.
11. K. Kozminski and E. Kinnen. Rectangular duals of planar graphs. *Networks*, 5(2), 1985.
12. F. Mansmann, D. A. Keim, S. C. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE Transactions on Visualization and Computer Graphics*, 13:1105–1112, November 2007.
13. E. Raisz. The rectangular statistical cartogram. *Geographical Review*, 24(2):292–296, 1934.
14. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.
15. A. Slingsby, J. Dykes, and J. Wood. Configuring hierarchical layouts to address research questions. *IEEE Trans. Vis. Comput. Graph.*, 15(6):977–984, 2009.
16. A. Slingsby, J. Dykes, and J. Wood. Rectangular hierarchical cartograms for socio-economic data. *Journal of Maps*, pages v2010, 330–345. 10.4113/jom.2010.1090, 2010.
17. M. van Kreveld and B. Speckmann. On rectangular cartograms. *CGTA*, 37(3), 2007.
18. J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1348–1355, 2008.