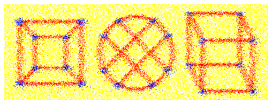

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 19, no. 1, pp. 413–440 (2015)
DOI: 10.7155/jgaa.00367

Fitting Planar Graphs on Planar Maps

Md. Jawaherul Alam[1] *Michael Kaufmann*[2]
Stephen G. Kobourov[1] *Tamara Mchedlidze*[3]

[1]Department of Computer Science, University of Arizona
[2]Institut für Informatik, Universität Tübingen
[3]Institute of Theoretical Informatics, Karlsruhe Institute of Technology

Abstract

Graph and cartographic visualization have the common objective to provide intuitive understanding of some underlying data. We consider a problem that combines aspects of both by studying the problem of fitting planar graphs on planar maps. After providing an NP-hardness result for the general decision problem, we identify sufficient conditions so that a fit is possible on a map with rectangular regions. We generalize our techniques to non-convex rectilinear polygons, where we also address the problem of efficient distribution of the vertices inside the map regions.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
November 2013	May 2014	October 2014	November 2014	August 2014

Published: September 2014

Article type: Regular paper

Communicated by: H. Meijer

A preliminary version of this paper will appear in SOFSEM 2014. In addition to providing all proofs, this version includes new results, such as those in section 5

E-mail addresses: mjalam@cs.arizona.edu (Md. Jawaherul Alam) mk@informatik.uni-tuebingen.de (Michael Kaufmann) kobourov@cs.arizona.edu (Stephen G. Kobourov) mched@iti.uka.de (Tamara Mchedlidze)

1 Introduction

Visualizing geographic maps may require showing relational information between entities within and between the map regions. We study the problem of fitting such relational data on a given map. In particular, we consider the problem of *fitting planar graphs on planar maps*, subject to natural requirements, such as avoiding edge crossings and ensuring that edges between points in the same region remain in that region.

Fitting planar graphs on planar maps is related to cluster planarity [3, 4, 14]. In a *cluster-planar drawing*, (also called *c-planar drawing* in the literature), a *plane graph* (i.e., a planar graph with a fixed planar embedding) along with a *clustering* (i.e., a partition of the vertices) is given. The goal is to find disjoint regions in the plane for the clusters for a valid plane realization of the given graph. The realization is *valid* if all the vertices in each given cluster and the edges between them are drawn in their corresponding region, and the drawing is planar.

In our setting (fitting graphs on maps), we are given both the graph and the regions embedded in the plane, and must draw the clusters in their corresponding regions (rather than compute suitable regions). The regions form a proper partition of the plane (except for a single connected unbounded outer region), such that the *adjacency between two clusters* (i.e., edges between vertices from the two clusters) is represented by a common border between their corresponding regions. We want to find a drawing of the graph such that the vertices in each cluster are placed in its corresponding region and there are no edge-crossings. or edge-region crossings (i.e., edges crossing a region in the map, including the outer region, more than once).

1.1 Related Work

The concept of clustering involves grouping a set of objects so that objects in the same group are more similar to each other than to those in other groups. In graph theory, this notion is captured by a clustered graph. Clustering of graphs is used in information visualization [20], VLSI design [17], knowledge representation [21], and many other areas.

For the analysis and visualization of clustered graphs in these application areas, it is desirable to draw a clustered graph in such a way that a visual separation of its clusters is evident. This is usually obtained by a drawing of the graph where the vertices and edges of each cluster can be surrounded by a simple closed curve defining disjoint regions in the plane. Feng *et al.* thus defined *c-planarity* as planarity for plane clustered graphs [15], where the drawing is planar (i.e., there is no edge-crossings) and the drawing for each cluster remains in its corresponding region (.i.e., there is no edge-region crossings); also see Section 2 for related definitions. For clustered graphs in which every cluster induces a connected subgraph, c-planarity can be tested in quadratic time. Without the connectivity condition, the complexity of testing c-planarity is still an open problem. Algorithms for creating regions in the plane in which to draw

c-planar graphs have also been studied. Eades *et al.* [12] presented an algorithm for constructing c-planar straight-line drawings of c-planar clustered graphs in which each cluster is drawn as a convex region, while Angelini *et al.* [1] show that every such c-planar clustered graph has a c-planar straight-line drawing where each cluster is drawn inside an axis-aligned rectangle.

Many visualizations take advantage of our familiarity with maps by producing map-like representations that show relations among abstract concepts. For example, treemaps [27], squarified treemaps [6] and news maps represent hierarchical information by means of space-filling tilings, allocating area in proportion to some metric. Concept maps [10] are diagrams showing relationships among concepts. Somewhat similar are cognitive maps and mind-maps that represent words or ideas linked to and arranged around a central keyword. GMap [20] uses the geographic map metaphor to visualize relational data by combining graph layout and graph clustering, together with the creation and coloring of regions/countries.

Also related is work on contact graphs, where vertices are represented by simple interior-disjoint polygons and adjacencies are represented by a shared boundary between the corresponding polygons. For example, every maximally planar graph has a contact representation with convex polygons with at most six sides, and six sides are also necessary [11]. Of particular interest are *rectilinear duals*, where the vertices are represented by simple (axis-aligned) rectilinear polygons. It is known that 8 sides are sometimes necessary and always sufficient [18, 25, 29]. If the rectilinear polygons are restricted to rectangles, the class of planar graphs that allows such *rectangular duals* is completely characterized [24, 28] and can be obtained via bipolar orientation of the graph [16]; see Buchsbaum *et al.* [7] and Felsner [13] for excellent surveys.

1.2 Our Contributions

We first consider the question of testing whether a given planar clustered graph fits on a given planar map and show that the decision problem is NP-hard, even in the case where the map is made of only rectangular regions and each region contains only one vertex. Then we provide sufficient conditions that ensure such a fit on a rectangular map. Finally, we generalize the fitting techniques to rectilinear maps with rectangles, L-shaped and T-shaped polygons. In particular, we describe an efficient algorithm for distributing vertices appropriately in the case of maps with L-shaped polygons.

The rest of the paper is organized as follows. Section 2 contains some definitions and terminology that we used throughout the paper. In Section 3 we show that the problem of straight-line planar fitting of a planar graph on a map is NP-hard even when all the regions in the map are rectangles. We give two sufficient conditions for fitting in this case in Section 4. In Section 5 we address the problem of fitting on a more general case of rectilinear maps consisting of rectangles, L-shaped and T-shaped polygons. Section 6 concludes the paper.

2 Preliminaries

In this section we introduce definitions used throughout the paper and then describe the properties of clustered graphs considered in the paper.

Let $G = (V, E)$ be a planar graph, with vertex set V partitioned into disjoint sets $\mathcal{V} = \{V_1, \ldots, V_k\}$. We call the pair $C = (G, \mathcal{V})$ a *planar clustered graph*. We consider the following partition of the edges of G that corresponds to the given partition of vertices $\mathcal{V} = \{V_1, \ldots, V_k\}$. Let E_i, for each i, $1 \leq i \leq k$ be the set of edges in E between two vertices of V_i and let E_{inter} be the set of all the remaining edges in E. Note that $E = E_1 \cup E_2 \cup \ldots \cup E_k \cup E_{inter}$. We call $G_i = (V_i, E_i)$, $1 \leq i \leq k$, a *cluster* of G, the edges of E_i, $1 \leq i \leq k$, the *intra-cluster edges* and the edges of E_{inter} the *inter-cluster edges*.

The *cluster-graph* of a clustered graph $C = (G, \mathcal{V})$ is the graph $G_C = (\mathcal{V}, \mathcal{E})$, where the edge $(V_i, V_j) \in \mathcal{E}$, $1 \leq i, j \leq k$, $i \neq j$ if there exists an edge (u, w) in G so that $u \in V_i$ and $w \in V_j$. A clustered graph $C = (G, \mathcal{V})$ is said to be *connected* (resp. *biconnected*) if each of G_i, $1 \leq i \leq k$, is a connected (resp. biconnected) graph.

A *drawing of a planar clustered graph* $C = (G, \mathcal{V})$ is a planar straight-line drawing of G where each cluster G_i is represented by a simply-connected closed region R_i such that R_i contains only the vertices of G_i and the drawing of each edge e in E_i is completely contained in R_i. An edge e and a region R (i.e., either a region for a cluster or the single connected unbounded outside region) have an *edge-region crossing* if the drawing of e crosses the boundary of R more than once. A drawing of a planar clustered graph C is *c-planar* if there is no edge crossing or edge-region crossing. If C has a c-planar drawing then we say that it is *c-planar*.

A *polygonal map* M is a set of interior-disjoint polygons in a plane. A *dual graph* G_M of M is a graph that contains one vertex for each polygon of M. Two vertices of G_M are connected by an edge if the corresponding polygons have a non-trivial common boundary. Given a planar graph G_M, a polygonal map M is called a *contact map* of G_M if G_M represents the dual graph of M. Let $C = (G, \mathcal{V})$ be a planar clustered graph. A polygonal map M which represents a contact map of the cluster-graph G_C is said to be *compatible* with C. Notice that this definition yields a correspondence between the clusters of C and polygons of M. In this paper we are interested in determining whether each cluster G_i of C can be drawn with straight-line edges inside its corresponding polygon in M, so that there is no edge crossing and no edge-region crossing. In case such a drawing exists we say that planar clustered graph C has a *straight-line planar fitting*, or just *planar fitting* on map M.

It is natural to consider all planar graphs, regardless of the clustering they come with. We preview the construction of a straight-line planar fitting and isolate the problem we are interested in. Recall that, by the definition of a planar fitting, each cluster has to be drawn inside a polygon, and there should be no edge crossings and no edge-region crossings. This implies that a clustered graph that has a planar fitting is also c-planar, so we consider only c-planar graphs. Unfortunately, the characterization of c-planar graphs is still an open

problem. Thus we restrict ourselves to clustered graphs for which we know that c-planarity can be efficiently tested. We use the results of Feng et al. [15] who provide a polynomial-time algorithm to test whether a *connected* clustered graph is c-planar. Thus, in the rest of the paper we consider only *connected c-planar graphs*.

3 Fitting on a Rectangular Map

In this section we consider the problem of deciding whether a connected c-planar graph G has a straight-line planar fitting on a given compatible rectangular map M. We first show that such a fitting does not always exist. To construct the counterexample we use a *wheel map*, which contains a center rectangle and four congruent "thin rectangles" that surround the center rectangle; see Fig. 1(a)–(b).

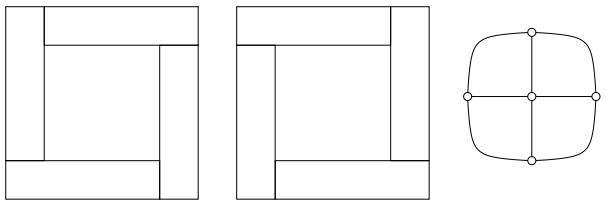


Figure 1: Wheel maps cw (a) and ccw (b), and their dual graph (c), which is also a trivial clustered graph on these maps (each vertex constitutes a cluster).

Intuitively, the notion of a thin rectangle will be clear in the following constructions from the way it is used, but to be more precise, we formally define it. A *thin rectangle* is one whose larger side is at least 4 times its smaller side, i.e., it has aspect ratio at least 4. A thin rectangle is *horizontal* if its smaller side is its height; otherwise it is *vertical*. We assume all four thin rectangles in a wheel map have the same size (same length of larger sides, same length of smaller sides).

Let $\{V_1, \ldots, V_k\}$ be the set of clusters of G and let (v_i, v_j) be an edge of G such that $v_i \in V_i$, $v_j \in V_j$, $1 \leq i, j \leq k$. Then there exists a common boundary between the polygons representing V_i and V_j in M. Call the common boundary the *door* for the edge (v_i, v_j). Consider a wheel map W and its dual G which has a simple clustering: each vertex constitutes a cluster; see Fig. 1(c). For the rest of the section we often assume that a wheel map is associated with this clustered graph. With this consideration in mind, each thin rectangle of W contains a door for each of its two incident thin rectangles. We define the *entry door* to be the one which contains a complete side of the rectangle itself, and the *exit door* to be the one that contains a complete side of an incident thin rectangle. We call a wheel map a *clockwise (cw) wheel* when going from the entry door to the exit door in each rectangle requires a clockwise walk through

the wheel; see Fig. 1(a). A *counterclockwise (ccw) wheel* is defined analogously; see Fig. 1(b).

We now define the notion of "proximity region" for the entry and exit doors inside each thin rectangle of a wheel map. Given that the thin rectangles of a wheel map are of the same size, using basic geometry we can define inside each thin rectangle a triangular region of maximum area where these two conditions hold: (i) the triangular region inside each thin rectangle contains the entry door of this rectangle; and (ii) for each point inside the triangle of one of the four thin rectangles, there exist three other points, one inside the triangle of each other rectangle, such that each pair in adjacent rectangles can be joined by a line segment within the rectangles. We call these triangular regions the *proximity regions* of the corresponding entry doors; see Fig. 2(a). For each exit door we can analogously define a quadrangular *proximity region*; see Fig. 2(b). Next we state a simple observation that follows from these definitions:

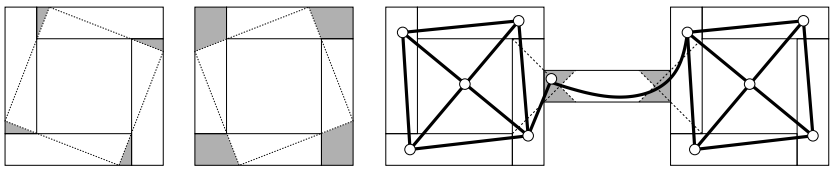


Figure 2: (a)–(b) Proximity regions for the doors in a wheel map, (c) a clustered graph and a map with no fit (the visibility regions of the bridge are highlighted).

Observation 3.1 *Let W be a wheel map and let G be its dual graph. In a straight-line planar fitting of G the vertices in the thin rectangles either all lie inside the proximity region of the entry doors, or they all lie in the proximity region of the exit doors. There exists a straight-line planar fitting in each case.*

Proof: The sufficiency follows from the definition of proximity regions. The necessity follows from the fact that the proximity regions for the entry and the exit doors inside each thin rectangle are disjoint since the aspect ratio of the thin rectangles is ≥ 4. □

The next lemma shows that fitting a planar clustered graph on a compatible map is not always possible.

Lemma 1 *There exist a planar clustered graph $C = (G, \mathcal{V})$ and a compatible rectangular map M, so that there is no straight-line planar fitting of C on M.*

Proof: Consider a rectangular map M made of two wheel maps (of the same size) joined together by a thin horizontal rectangle, called a *bridge*; see Fig. 2(c). We choose the height of the bridge to be at most the length of the smaller sides in the thin rectangles of the wheels and we attach the bridge in the middle of the neighboring two vertical rectangles. Call these two vertical rectangles R_1

and R_2, respectively and call the bridge B. A point p of B is *visible* from a point q in R_i, $i = 1, 2$ if the visibility line between p and q passes through the door between B and R_i. For $i = 1, 2$, the *visibility region* of B for R_i is the set of all points in B that are visible from at least one point in the proximity region of either the entry or the exit door of R_i. We choose the length of the bridge to be long enough, such that the two visibility regions for the two rectangles do not overlap.

Let G be the dual of M: two 5-vertex wheels connected by a path of length two. Assume a clustering of G where each vertex constitutes a separate cluster. Then G has no straight-line planar fitting on M. If G had a straight-line planar fitting Γ on M, by Observation 3.1, all the vertices inside the thin rectangles of both the wheels must be placed in the proximity regions of the doors in Γ. But then, there is no feasible position for the vertex that represents the bridge since the two visibility regions of the bridge do not overlap. □

3.1 Fitting is NP-Hard

We show that deciding if a given plane clustered graph has a planar fitting inside a given map is NP-hard, even for rectangular maps, with a reduction from Planar-3-SAT which is known to be NP-complete [26]. *Planar-3-SAT* is defined analogously to 3-SAT with the additional restriction that the *variable-clause bipartite graph* G_F for a given formula F is planar. There is an edge (x_i, A_j) in G_F if and only if x_i or $\overline{x_i}$ appears in A_j. Knuth and Raghunathan [23] showed that one can always find a crossing-free drawing of the graph G_F for a Planar-3-SAT instance, where the variables and clauses are represented by rectangles, with all the variable-rectangles on a horizontal line, and with vertical edge segments representing the edges connecting the variables to the clauses. The problem remains NP-complete when such a drawing is given.

Theorem 1 *Let $C = (G, \mathcal{V})$ be a planar clustered graph with a rectangular map M, compatible with C. Deciding if C admits a straight-line planar fitting on M is NP-hard.*

Proof: We reduce an instance of Planar-3-SAT to an instance (C, M) of our problem. Let $F := A_1 \wedge \ldots \wedge A_m$ be an instance of a Planar-3-SAT, where each literal in each clause A_i is a variable (possibly negated) from $U = \{x_1, \ldots, x_n\}$. Let Γ_F be the given planar rectilinear drawing for this instance, as defined in [23]. From Γ_F we first construct the rectangular map M, then take G as the dual of M, where each vertex constitutes a separate cluster. We represent each literal by a wheel map (of the same size) in M: a positive (negative) literal is a cw (ccw) wheel. From the two possible vertex configurations inside each wheel we take the one in which the corresponding literal assumes a true value when the vertices inside the thin rectangles of the wheel lie in the proximity region of the exit doors and the literal assumes a false value when they lie in the proximity region of the entry doors. Unlike in Γ_F, we use a distinct wheel for each literal in each clause. For each variable x, we draw the wheels for all the (positive

and negative) literals for x appearing in different clauses in a left-to-right order, according to the ordering of the edges incident to the corresponding vertices in Γ_F. To maintain consistency, we ensure that a true (false) value for a literal x implies a true (false) value for every other instance of x and a false (true) value for each instance of $\overline{x}$. This is done by means of thin rectangular bridges between two consecutive literals; see Fig. 3. The size of the bridges is chosen equal to the thin rectangles in the wheels.

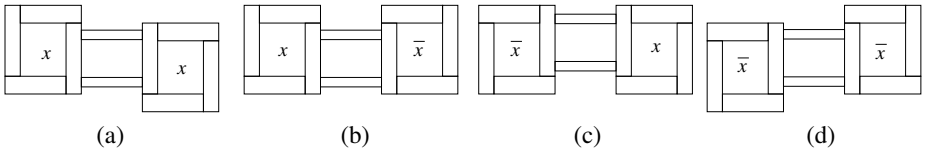


Figure 3: Representation of variables.

For each clause $A = (x \vee y \vee z)$ of F, with the corresponding vertex lying above the variables in Γ_F, we draw vertical rectangles l_x^A, l_y^A and l_z^A from the topmost rectangles $T_x,\ T_y, T_z$ of the wheels for x, y and z, respectively, attached at the end that is not shared by other thin rectangles. (The case when the vertex lies below the variables in Γ_F is analogous.) We place l_x^A, l_y^A, l_z^A so that they are completely visible from all the points in the proximity of the exit doors of $T_x,\ T_y, T_z$, respectively. We choose the length of the rectangles l_x^A, l_y^A, l_z^A so that not all points inside them are visible from any point of the proximity region of the entry doors of $T_x,\ T_y, T_z$, respectively; see Fig. 5.

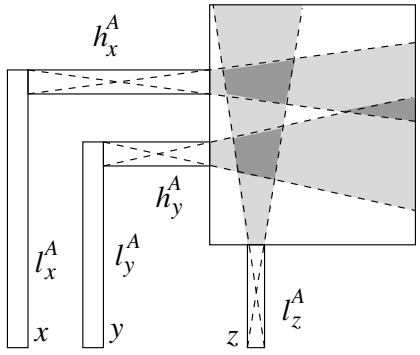


Figure 4: Clause representation.

We then draw a rectangle R for the clause and attach these three thin rectangles l_x^A, l_y^A and l_z^A to R. For z we attach the vertical rectangle l_z^A to the bottom of R, while for each of x and y, we attach horizontal rectangles h_x^A, h_y^A to R that also touch the vertical rectangles l_x^A and l_y^A coming from x or y, respectively. A point p in rectangle h_x^A (h_y^A) is *reachable* from a point q inside T_x (T_y) if there exists a point r inside l_x^A or l_y^A such that the two lines pr and rq pass through

the doors between the corresponding rectangles. The *reachable region* of h_x^A (h_y^A) is the set of all points that are reachable from a point inside the proximity region of the entry door of T_x (T_y). Similarly a point p inside l_z^A is reachable from a point q inside rectangle T_z of the wheel for z if the line pq passes through the door between the two rectangles. The *reachable region* of l_z^A is the set of all points reachable from a point inside the proximity region of the entry door of T_z. Choose the lengths for the horizontal rectangles h_x^A, h_y^A and the vertical rectangles l_x^A, l_y^A, l_z^A so that the reachable regions inside them do not coincide with the entire inside of these rectangles. For this purpose it is sufficient that the sizes of these rectangles are comparable to the sizes of the rectangles inside the wheels. Thus the sizes of all the wheels and other rectangles are polynomial in the size of the Planar-3-SAT instance.

Next we attach the thin rectangles h_x^A, h_y^A, l_z^A to R in such a way that the areas visible from the reachable regions of h_x^A, h_y^A, l_z^A do not have a common intersection, while every pair of them does have a common intersection; see Fig 4. We now observe that the size of R does not have to be too big to ensure this. Specifically, we attach l_z^A to the left half of the bottom line of R and choose the height of R small enough so that the visible area from its reachable region is only in the left half of R. We also adjust the vertical distance between the horizontal rectangles h_x^A and h_y^A and adjust the width of R so that the areas visible from the reachable regions of h_x^A and h_y^A do not intersect in the left half of R but they do intersect in the right half. This can be achieved if for example we take the vertical distance between h_x^A and h_y^A to be a constant multiple of their height, while we take the width of R to be a constant multiple of the length of h_x^A, h_y^A. Finally we fill all the unused regions in the map with additional rectangles to get the final map M. Since the sizes of all the rectangles are constant multiples of each other and the total size is polynomial in the size of the Planar-3-SAT instance, the coordinates for the map can be chosen to be polynomial in the size of the Planar-3-SAT instance.

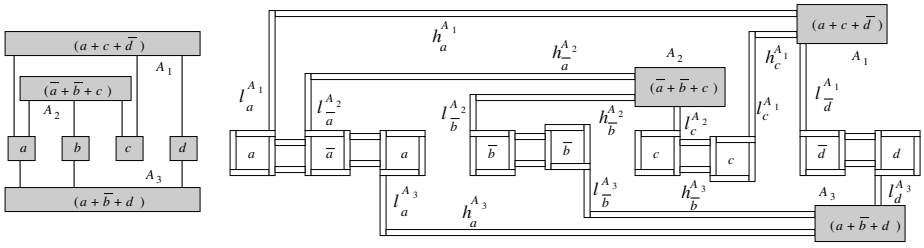


Figure 5: Planar 3-SAT instance and corresponding map fitting instance.

Lemma 2 *F is satisfiable if and only if G has a straight-line planar fitting on M.*

Proof: Assume first that there exists a straight-line planar fitting Γ of G on M. We show that F is satisfiable, i.e., there is a truth assignment for all the

variables of F such that for each clause $A = (x, y, z)$ of F, at least one of x, y and z is true. Let W_x be the wheel for x. If the vertices in the thin rectangles of W_x are placed inside the proximity regions of the entry doors, then by construction of M, the vertex in the horizontal rectangle h_x^A is placed inside the reachable region. Thus this vertex can see only the highlighted visible area in Fig. 4 inside the rectangle R for A. However if the vertices in the thin rectangles of W_x are placed in the proximity regions of the exit doors, then the vertex in the horizontal rectangle h_x^A can be placed outside the reachable region so that it can see the entire interior of R. This is true for each of the three literals. Since the visible areas of the three literals have no common intersection, the vertices in the wheel for at least one of x, y and z must be placed in the proximity region of the exit door. We make each such literal true. This assignment has no conflict, because of the way the wheels for a particular variable are attached to each other. Furthermore, this assignment satisfies F.

Conversely if F is satisfiable, for each clause $A = (x, y, z)$ of F, at least one of x, y, z is true. Without loss of generality, assume that x is true. Place the vertices in the wheel of x in the proximity regions of the exit doors. Then the vertex in the h_x^A can be placed outside the reachable region and it can see the entire interior of R. Place the vertex for R in the intersection of the areas visible from reachable regions of h_y^A and l_z^A. This ensures that we can place the vertices in the wheel for y and z in the proximity regions of either the entry doors or the exit doors and we are still able to place the vertices in rectangles h_y^A, l_y^A, l_z^A so that all the straight-line edges create no area-region crossings. This yields the desired straight-line planar fitting of G on M. □

The proof of Lemma 2 completes the NP-hardness proof. Fig. 5 illustrates a 3-SAT formula, its Planar-3-SAT realization with the conditions of [23], and the corresponding instance for the map fitting problem (rectangles filling up the holes are not shown). □

Note that Bern and Gilbert [5] and recently Kerber [22] obtained NP-completeness results using similar techniques. In particular, Bern and Gilbert [5] consider the problem of drawing the dual on top of the drawing of a plane graph G, such that each dual vertex lies in the corresponding face of G, while each dual edge is drawn as a straight-line segment that crosses only its corresponding primal edge. They show that this problem is NP-complete and the techniques used are similar to ours, as this problem can be thought of as a special case of fitting a clustered graph on a map, where each cluster consists of a single vertex. However, we consider the more restricted class of rectangular maps instead of the generic drawing of a planar graph, and hence the NP-completeness of our problem is not implied by [5]. Kerber [22] considers the problem of embedding the dual on top of a primal partition of the d-dimensional cube into axis-aligned simplices and proved that this problem is NP-complete. In 2D, this problem is also a special case of our problem, with the exception that in Kerber's setting edge-region crossings are allowed. Thus the result in [22] also does not imply our results.

4 Sufficient Conditions for Fitting

We showed in the previous section (Lemma 1) that not every c-planar connected graph admits a planar straight-line fitting on a compatible map even if each cluster is a single vertex. The counterexample relies on two facts: (1) there exists a vertex in some cluster (the bridge) that is connected to vertices in two different clusters (the wheels); (2) its cluster-graph contains two cycles. By considering graphs that do not have at least one of the above characteristics we show a planar straight-line fitting is always possible. In this sense the following two lemmas present tight sufficient conditions for graphs to admit planar straight-line fittings.

Lemma 3 *Let $C = (G, \mathcal{V})$ be a biconnected c-planar graph. Let M be a rectangular map compatible with C. If for each vertex v of G, all the vertices adjacent to v through an inter-cluster edge lie in the same cluster, C has a straight-line planar fitting on M.*

Proof: Let Γ be a c-planar drawing of C. Let $G_1, G_2, \ldots, G_k$ be the clusters of C and let $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ be the corresponding vertex partition. For each rectangle R_i, $1 \leq i \leq k$, of M representing the cluster G_i, denote by O_i the ellipse inscribed in R_i. We first place the vertices on the outer boundary of G_i in Γ on O_i as follows. Consider two adjacent rectangles R_i and R_j in M. Let $v_{i_1}, \ldots, v_{i_r} \in V_i$ and $v_{j_1}, \ldots, v_{j_s} \in V_j$ be the vertices of V_i and V_j, incident to the inter-cluster edges between these two clusters, taken in the order they appear on the outer boundary of G_i and G_j, respectively. Define p_i, p'_i and p_j, p'_j to be points of O_i and O_j, respectively, such that the straight-line segments $\overline{p_i p_j}$ and $\overline{p'_i p'_j}$ cross the common border of R_i and R_j, without crossing each other. Place the vertices $v_{i_1}, \ldots, v_{i_r}$ of V_i and $v_{j_1}, \ldots, v_{j_s}$ of V_j on O_i and O_j, between points p_i, p'_i and p_j, p'_j, respectively, so that all the inter-cluster edges between these vertices cross the common border of R_i and R_j. Repeat the above procedure for each pair of adjacent rectangles in M. Since each vertex thus placed is adjacent to a unique cluster, its position is uniquely defined. Then for each cluster G_i, $1 \leq i \leq k$, we have placed some vertices on the outer boundary of G_i in Γ on the ellipse O_i. Distribute the remaining vertices of the boundary of G_i on O_i, so that the order of the vertices is the same as in the boundary of G_i. Since the resulting drawing of the outer boundary of G_i is convex and G_i is biconnected, apply the algorithm for drawing a graph with a prescribed convex outer face [9] to complete the drawing of each cluster. □

Lemma 4 *Let $C = (G, \mathcal{V})$ be a biconnected c-planar graph. Let M be a rectangular map compatible with C. If each connected component of cluster-graph G_C contains at most one cycle, then C has a straight-line planar fitting on M.*

Proof: Assume that each connected component of G_C contains at most one cycle. Let $v_1, \ldots, v_k$ be the vertices of G_C that represent clusters $G_1, \ldots, G_k$ respectively. To complete the proof we go through the following steps:

(1) We show that G_C has a planar fitting on M.

(2) We blow up the drawing of G_C, so that the edges of G_C are represented by strips of width $\varepsilon > 0$, without creating edge-region crossings; see Fig. 6(a). For each vertex v_i of G_C, we draw a small circle $circ(G_i)$ centered at the intersection of the strip-edges that are adjacent to v_i.

(3) We draw the boundary of G_i on the circle $circ(G_i)$, $i = 1, \ldots, k$, so that the inter-cluster edges, when drawn with straight-line segments, intersect neither the boundaries of the clusters, nor each other; see Fig. 6(b).

(4) Since the boundary of each G_i is a convex polygon and G_i is biconnected, we can apply the algorithm for drawing a graph with a prescribed convex outer face [9] to complete the drawing of the clusters; see Fig. 6(c).

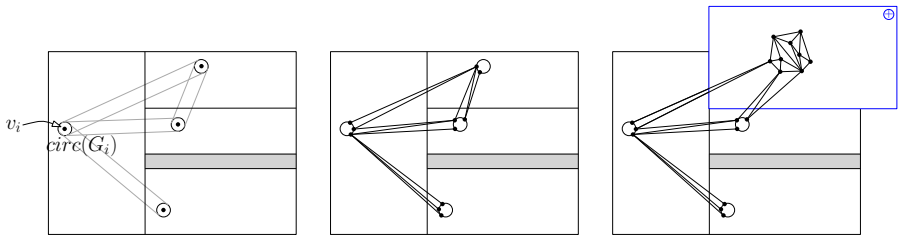


Figure 6: (a) Drawing of G_C, each edge is represented by a strip of width $\varepsilon > 0$. (b) Placing the boundary vertices of the clusters on the corresponding circles. (c) Step 4 of the proof of Lemma 4.

While steps (2) and (4) are straight-forward, steps (1) and (3) need to be proved. We provide a detailed proof for these two steps below.

Step (1). We show that $G_C = (V_C, E_C)$ has a straight-line planar fitting on M. Consider first the case when G_C is a tree and let $v_1 \in V_C$ be the root of G_C. We prove that even if the position of v_1 is fixed in its corresponding rectangle R_1, we can place the remaining vertices of G_C in their corresponding rectangles so that the resulting straight-line drawing is a planar fitting of G_C on M. Let $v_2, \ldots, v_f$ be the children of v_1 and let $R_2, \ldots, R_f$ be the corresponding rectangles of M. Place $v_2, \ldots, v_f$ inside $R_2, \ldots, R_f$, respectively so that the straight-line edges (v_1, v_i), $2 \leq i \leq f$ cross the common boundary of R_1 and R_i. Continue with the children of $v_2, \ldots, v_f$, recursively.

Assume now that each connected component of $G_C = (V_C, E_C)$ contains at most one cycle. We show how to draw a single connected component of G_C. Let $v_0, \ldots, v_m \in V_C$ induce the unique cycle of G_C and let $R_0, \ldots, R_m$ be the rectangles that correspond to them, so that R_i and $R_{(i+1) \mod (m+1)}$, $0 \leq i \leq m$, are adjacent. Place v_i, $0 \leq i \leq m$ inside R_i such that for any point $p \in R_{(i+1) \mod (m+1)}$, segment $\overline{pv_i}$ crosses the common boundary of R_i and

$R_{(i+1) \bmod (m+1)}$. Since this was the only cycle of G_C, the removal of the edges of this cycle results in several trees. Root the trees at the vertices $v_0, \ldots, v_m$, to which they are adjacent and apply the procedure described in the first part of the proof. This completes the construction of planar fitting of G_C on M.

Step (3)**.** Since graph G is c-planar there exists a drawing $\Gamma(G)$ of G, where all the vertices of G_i that are adjacent to the vertices of clusters $G_1, \ldots, G_{i-1}$, $G_{i+1}, \ldots, G_k$ appear on the outer face of G_i [14]. Let u_1^i be a vertex of G_i that lies on G_i's boundary; see Fig. 7(a). Let $v_1^i, \ldots, v_{p_i}^i$ be all the neighbors

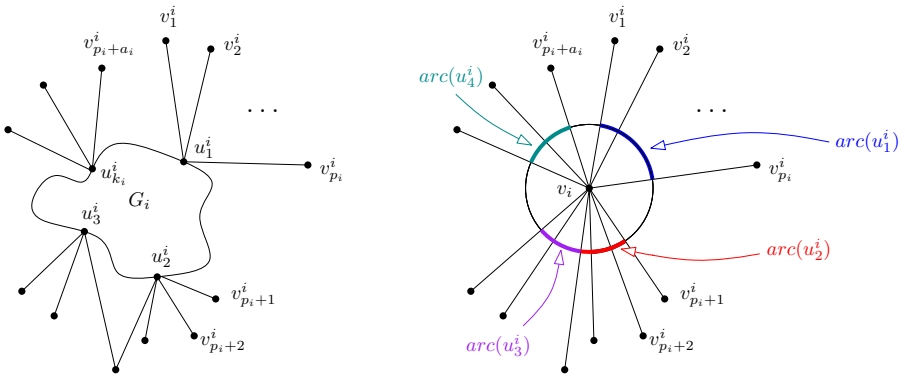


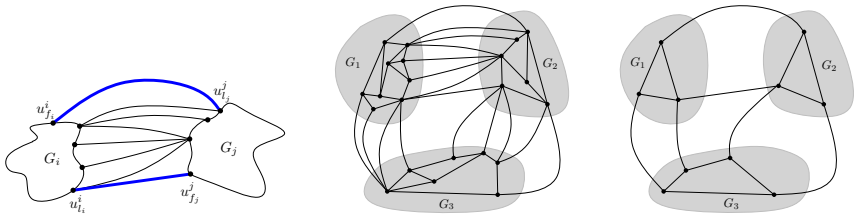
Figure 7: Illustration for the proof of Lemma 3.

of v_i that represent the clusters to which vertex u_1^i is adjacent. Assume that $v_1^i, \ldots, v_{p_i}^i$ are given in the clockwise order they appear around v_i in $\Gamma(G_C)$. We denote by $arc(u_1^i)$ a circular arc of $circ(G_i)$ that is included between the straight-line segments $\overline{v_i, v_1^i}$ and $\overline{v_i, v_{p_i}^i}$, as one travels from v_1^i to $v_{p_i}^i$ in the clockwise direction; see Fig. 7(b). Since $\Gamma(G_C)$ is a straight-line planar drawing, we have the following observation.

Observation 4.1 *Let v_i be a vertex of G_C and G_i be a cluster of G that corresponds to v_i. Let $u_1^i, \ldots, u_{k_i}^i$ be the vertices of the boundary of G_i traversed in the clockwise direction. The circular arcs $arc(u_1^i), \ldots, arc(u_{k_i}^i)$ appear clockwise around the $circ(G_i)$ in this specific order and are internally disjoint; see Fig. 7(a)-(b).*

Let now G_i and G_j be two clusters of G. Assume that they are connected by multiple inter-cluster edges. Let $u_{f_i}^i, \ldots, u_{l_i}^i$ (resp. $u_{f_j}^j, \ldots, u_{l_j}^j$) be the vertices of the boundary G_i (resp. G_j) in the clockwise direction that are involved in the inter-cluster edges between G_i and G_j; see Fig. 8(a).

The edges $(u_{f_i}^i, u_{l_j}^j), (u_{l_i}^i, u_{f_j}^j)$ are called the *bounding* inter-cluster edges. In order to accomplish a drawing of G we first construct a drawing of its *skeleton*. The *skeleton* $\mathcal{S}(G)$ of G is a graph that is constructed as following: (1) consider



(a) (b) (c)

Figure 8: (a) The bounding edges between clusters G_i and G_j are drawn with fat blue ink. (b) A clustered graph G with clusters identified by gray. (c) The skeleton $\mathcal{S}(G)$ of graph G depicted in figure (b).

the graph containing the boundary of each cluster G_i and all the bounding inter-cluster edges, (2) in the resulting graph contract all the vertices of the boundaries of the clusters which have degree two; see Fig. 8(b)–(c). We assume that $\mathcal{S}(G)$ is an embedded graph, with the embedding which preserves the embedding of G. Consider a planar drawing $\Gamma(\mathcal{S}(G))$ so that the vertices of each cluster $G_i \cap \mathcal{S}(G)$ are drawn on the circle $circ(G_i)$ in the order they appear on the boundary of G_i. Such a drawing exists, since G is c-planar. But for an arbitrary placement of the vertices, it is not true that the edges of $\mathcal{S}(G)$ can be drawn straight-line without creating crossings. Next we show how to place the vertices of $\mathcal{S}(G) \cap G_i$ on the circle $circ(G_i)$ so that the edges of $\mathcal{S}(G)$ do not cross each other, when drawn straight-line. For each v of $\mathcal{S}(G) \cap G_i$, we place v in the middle of circular arc $arc(v)$. We next show that this results in no crossings between the inter-cluster edges. Let $u \in G_a, v \in G_b, w \in G_c$ and $s \in G_d$, so that (u, v) and (w, s) are two inter-cluster edges; see Fig. 9(a). Next we consider several cases based on whether the clusters G_a, G_b, G_c and G_d are distinct or not.

Case 1: The clusters G_a, G_b, G_c, G_d are pairwise distinct. A crossing between the edges (u, v) and (w, s) is impossible, since each of the G_a, G_b, G_c, G_d lie in a distinct rectangle of map M and an inter-cluster edge is drawn in the union of the two rectangles associated with the two clusters it bridges.

Case 2: Two of the non adjacent clusters G_a, G_b, G_c, G_d coincide. Assume that $G_a = G_c$. Recall that w is placed on the middle of circular segment $arc(w)$ and u in the middle of circular segment $arc(u)$. By Observation 4.1, the circular segments $arc(u)$ and $arc(v)$ are internally disjoint; see Fig. 9(b). Therefore the edges (u, v) and (w, s) do not cross each other.

Case 3: $G_a = G_c$ and $G_b = G_d$. Let v_a and v_b be the vertices of G_C that correspond to G_a and G_b, respectively. First, note that (u, v) and (w, s) are bounding edges of G and form a cycle u, w, s, v in $\Gamma(\mathcal{S}(G))$. Without loss of generality assume that we traverse this cycle in this specific order; see Fig. 9(c). Then u appears before w and s before v on the boundary of G_a and G_b, respectively. By Observation 4.1, the circular segment $arc(u)$

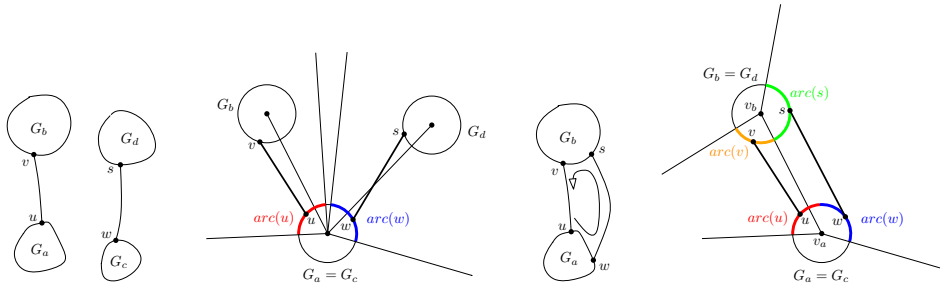


Figure 9: (a) Two edges of G, with the end vertices belonging to distinct clusters G_a, G_b, G_c, G_d. (b) Case 2 of the proof, $G_a = G_c$. (c-d) Case 3 of the proof, $G_a = G_c$ and $G_b = G_d$.

appears before $arc(w)$ in the clockwise order around $circ(G_a)$. Since both u and w are adjacent to G_b, $arc(u)$ and $arc(w)$ meet at a point lying on the line through v_a and v_b; see Fig. 9(d). Similarly, $arc(s)$ appears before $arc(v)$ in the clockwise order around $circ(G_b)$ and meet at the line through v_a and v_b. Thus, edges (u, v) and (w, s) are separated by the horizontal line through v_a, v_b and hence do not cross.

We have constructed a planar straight-line drawing of $\mathcal{S}(G)$. We complete the proof by explaining how to draw the remaining vertices of the boundaries of the clusters of G and the inter-cluster edges of G. For each cluster G_i, $i = 1, \ldots, k$, connect the vertices of $G_i \cap \mathcal{S}(G)$ that have already been placed with straight-lines in the order they appear on the boundary of G_i. They form a convex polygon. Place the remaining vertices of the boundary of G_i on the respective sides of this convex polygon. Draw the remaining inter-cluster edges straight-line. It is easy to see that they do not create crossings, since they lie in the convex polygons created by the bounding edges of G. □

Putting together the results in this section we obtain the following theorem:

Theorem 2 *Let $C = (G, \mathcal{V})$ be a biconnected c-planar graph. Let M be a rectangular map compatible with C. If (a) for each vertex v of G, all the vertices adjacent to v through an inter-cluster edge lie on the same cluster, or (b) each connected component of cluster-graph G_C contains at most one cycle, then C has a straight-line planar fitting on M. Moreover, there exist a c-planar graph C and a compatible map M which do not fulfill condition (a) and (b) and do not admit a planar straight-line fitting.*

5 Fitting Graphs on Rectilinear Maps

In this section we give a description of our results for more general rectilinear maps.

It is known that only a restricted class of planar graphs can be realized by rectangular maps. For general planar graphs, 8-sided polygons (T-shapes) are necessary and sufficient for contact maps [18] (some T-shapes degenerate to L-shapes or rectangles). In this section, we assume that the input is a rectilinear map, with rectangles, L- and T-shaped polygons, together with a c-planar graph G with a cluster-graph G_C. The first condition that we require for our input graph G is that each vertex of G is incident to at most one inter-cluster edge. From Lemma 3 this condition is sufficient for rectangular maps. Now, we extend this to L-shaped and T-shaped polygons (maps). We impose several conditions under which we prove the existence of a fitting. Because of the presence of concave corners, we impose our second condition: none of the clusters contains a *boundary chord*, i.e., a non-boundary edge between two boundary vertices.

The idea is to apply the algorithm for drawing a graph with a prescribed convex outer face [9]. We partition the polygons into convex pieces. Since the polygons form a contact map, for each common boundary of adjacent polygons there is at least one edge between the corresponding clusters. Our last condition restricts this further: between any two adjacent clusters there exist at least two independent inter-cluster edges. We call a graph in which every pair of adjacent clusters has this property *doubly-interconnected*.

We now have the following theorem.

Theorem 3 *Let G be a doubly-interconnected and biconnected c-planar graph such that the inter-cluster edges of G form a matching and there is no boundary chord in any cluster. Then there exists a straight-line planar fitting of G on any compatible map M with rectangular, L-shaped, or T-shaped polygons.*

Proof: We prove this theorem by constructing a straight-line planar fitting of G on M. We now show how we place the vertices of each cluster inside the corresponding polygon. First we place the vertices on the outer boundary of each cluster.

Note that the common boundary of two adjacent polygons contains at most two concave corners, one in each polygon. Since G is doubly-interconnected, there are at least two independent inter-cluster edges corresponding to this common boundary. We place the vertices incident to these inter-cluster edges next to the common boundary and in particular on both sides of these concave corners; see, for example, the placement of the vertices in the two concave corners on the common boundary between A and B in Fig. 10. This ensures that the cycle spanned by the boundary vertices of the cluster can be placed completely within the corresponding polygon with at most two concave corners along the cycle. Note that the placement of the boundary vertices around each polygon (other than those placed at the concave corners) is quite arbitrary provided that their placement follow the circular order along the outer-boundary of the cluster for that polygon. Let G_i be a particular cluster of G and let P_i be its corresponding polygon. If P_i is a rectangle, then we have already placed some vertices on the outerface of G_i, we distribute the remaining vertices on the outerface so that the outer face is in convex position and then we can directly

apply the algorithm for drawing a graph with a prescribed convex outer face [9] to complete the drawing of G_i.

We thus assume that P_i is either an L-shaped or a T-shaped polygon. If P_i is an L-shaped polygon, then we have already placed a vertex, say a, next to the concave corner of P_i. We choose a second boundary vertex, say b, to be placed at the convex corner of P_i opposite to its only concave corner. Since G is doubly-interconnected and since the polygons form a contact map, the vertex b exists in all such L-shaped polygons. A straight-line segment between a and b defines two convex regions inside the polygon. We now compute an $(a \rightsquigarrow b)$-path and place the vertices on this path on the straight-line segment between a and b; see Fig 10(a). Thus this path should not have shortcuts, where a *shortcut* of a path P is an edge between vertices nonadjacent in P. Furthermore this path should not contain any other boundary vertex, already placed elsewhere. We now show how we compute an $(a \rightsquigarrow b)$-path with the above properties. Assume that G_i contains at least one internal vertex, otherwise G_i can be trivially drawn by drawing its boundary. Since G_i has no boundary chord, we can triangulate its interior so that the resulting graph has no boundary chord as well. Therefore assume that G_i is internally triangulated without any boundary chord and hence is 3-connected [2]. Then there exists some $a \rightsquigarrow b$-path that does not contain any boundary vertex. In particular by Menger's theorem, there are at least three vertex disjoint $a \rightsquigarrow b$-paths in G_i. The path which is topologically the middle of these three does not contain any vertex on the boundary. Take the shortest such $a \rightsquigarrow b$-path P_{ab} not containing any boundary vertex. Then P_{ab} has no shortcut edge. This path placed on the straight-line cut between a and b divides the L-shaped polygon P into two convex regions. We can distribute the remaining vertices on the boundary so that they are in convex position inside the two convex pieces. Hence we are able to apply the algorithm in [9] directly to complete the drawing of G_i.

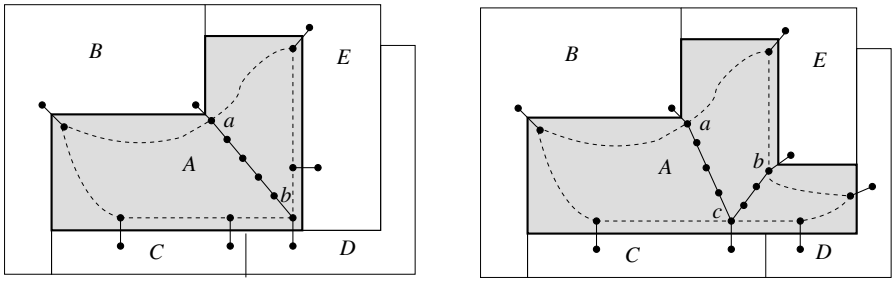


Figure 10: Illustrating how vertices on the boundary of an L-shaped and a T-shaped polygon are placed. The cluster for the L-shaped polygon is partitioned by an $(a \rightsquigarrow b)$-path into two convex polygons with the path as common boundaries. Similarly the cluster for the T-shaped polygon is partitioned by an $(a \rightsquigarrow c)$-path and a $(b \rightsquigarrow c)$-path into three convex polygons with the paths as common boundaries.

Finally assume that P_i is a T-shape polygon. Let a and b be the vertices placed next to the two concave corners of P_i. We choose a third boundary vertex c lying on the segment of P_i opposite to a and b; see Fig. 10(b). We choose c so that the straight-line segments between a, c and b, c are completely within P_i. Again since G is doubly-interconnected, for every pair of adjacent vertices we have at least two independent edges that connect the two clusters. Since the polygons form a contact map, this condition implies that the vertex c exists in all the T-shaped polygons that make a contact with its edge opposite to its two concave corners. We now compute an $(a \rightsquigarrow c)$-path and a $(b \rightsquigarrow c)$-path such that neither contains a boundary vertex and neither has a shortcut and place the vertices on these two paths on the two straight-line cuts between a, c and b, c, respectively. We again assume that G_i is internally triangulated and hence internally 3-connected. Then by Menger's theorem, there exist some $a \rightsquigarrow c$-path and some $b \rightsquigarrow c$-path that do not contain any boundary vertex. Take the shortest such $a \rightsquigarrow c$-path P_{ac} and shortest such $b \rightsquigarrow c$-path P_{bc}. Then P_{ac} and P_{bc} contain no boundary vertex and they have no shortcut edge. If these two paths are internally vertex disjoint, then we place the vertices on them on the two straight-line cuts between a, c and b, c. These two cuts define three convex regions in P_i and we can apply the algorithm in [9] directly to complete the drawing of G_i as before. In case the two paths P_{ac} and P_{bc} are not internally vertex disjoint we may assume that they have a common subpath towards c, say a $(c \rightsquigarrow d)$-path; since they are shortest paths. We then draw the $(a \rightsquigarrow d)$-path, $(b \rightsquigarrow d)$-path and $(c \rightsquigarrow d)$-path on three segments with a common endpoint at d, such that each pair of these segments are less than 180 degree apart from each other. This again forms three convex polygons and we can again apply the algorithm in [9] directly to complete the drawing of G_i. □

Note that there is another approach different than that in the proof of Theorem 3 to draw a cluster G_i inside its corresponding L-shaped or T-shaped polygon P_i. After we place the boundary vertices of G_i participating in a inter-cluster edge near some concave corner, we can distribute the remaining boundary vertices of G_i on segments parallel to the edges of P_i. Then the resulting polygon representing the boundary of G_i is a star-shape with a non-empty kernel. Assuming again that G_i is internally triangulated and hence internally 3-connected, we can then complete its drawing inside this polygon due to [19]]. However we use the approach described in Theorem 3 since in this approach we have a better vertex distribution since we do not need to place all the internal vertices in the kernel of the non-convex polygon.

However any algorithm that distributes vertices inside non-convex regions while preserving cluster-planarity must distribute the vertices among the convex components of the regions. It is only natural to try to make such a distribution balanced. In the following we consider a map containing only rectangles and L-shaped polygons and we aim to find a balanced distribution of vertices and faces inside the polygons of a map. We define a measure called "imbalance" which captures the difference between the geometric partition of the non-convex regions into convex regions (e.g., region area) and the partition of the clusters

into subclusters (e.g., subcluster size) corresponding to the convex regions. First we consider the distribution inside one L-shaped polygon partitioned into two pieces by a straight-line cut; then we use this result to simultaneously minimize the maximum imbalance in all L-shaped polygons in the map. We prove that the global imbalance minimization problem can be solved optimally, using dynamic programming and min-max paths. These techniques are interesting by themselves.

5.1 Shortest Path Separators

Here we consider the balanced distribution for a particular L-shaped polygon (the local problem). Given an L-shaped polygon with a straight-line cut partitioning it into parts of area A_1 and A_2, and the corresponding cluster C, we first compute an extended graph G by placing a dummy vertex in each face and adding edges from this dummy vertex to all the vertices in the face. Let P be a shortest path in the extended graph G and denote by $L(P)$ and $R(P)$ the subgraphs of G induced by the vertices in the two parts of G created by P, not including the vertices on P. Then the sizes of $L(P)$ and $R(P)$ give the summations of the numbers of vertices and faces in the two parts of the original graph partitioned by P. We want to find a shortest path P between two given boundary vertices s and t of G that contains no other boundary vertex and minimizes the *imbalance* $|A_1/A_2 - |L(P)|/|R(P)||$.

Lemma 5 *The values of $|L(P)|$ for all shortest paths P between fixed boundary vertices s and t of the extended plane graph G can be enumerated in $O(n^2)$ time.*

Proof: We prove this lemma constructively. We first remove all the boundary vertices from G except s and t. We then start by computing the shortest paths from s to t by a standard method like Dijkstra's algorithm, that can slightly be extended to compute all shortest paths between s and t. The output of this algorithm is a subgraph G_s of G consisting of vertices and edges that lie on at least one of the paths. The shortest paths on this graph imply an orientation for the edges of G_s, where each path between s and t is oriented towards t; see Fig. 11(a). Since G_s consists of directed shortest paths from s to t, G_s is a directed acyclic graph (In fact it is an st-digraph with s as the unique source and t as the unique sink). Furthermore, any directed path in G_s is a shortest path between its two end-points since it is a subpath of a shortest path. We consider an embedding of G where G_s is drawn *upward*, i.e. each edge of G_s is drawn upward. In such an embedding for a vertex v of G_s, we can define a *leftmost shortest path* from v to t in G_s, denoted by $left(v)$, where each edge (u, w) of the path is the leftmost outgoing edge incident to u. We now give a dynamic programming algorithm that finds and enumerates, for each vertex v of G_s, the value of $|L(P)|$ for possible paths P formed by any shortest path from s to v, followed by the path $left(v)$, where one of these subpaths might be empty if v is one of s or t. We call each such path a *feasible path* for v. We keep these values for a vertex v in a list denoted by **values**(v). Fig. 11(a) shows a feasible path P for a vertex v and the highlighted region defines $L(P)$.

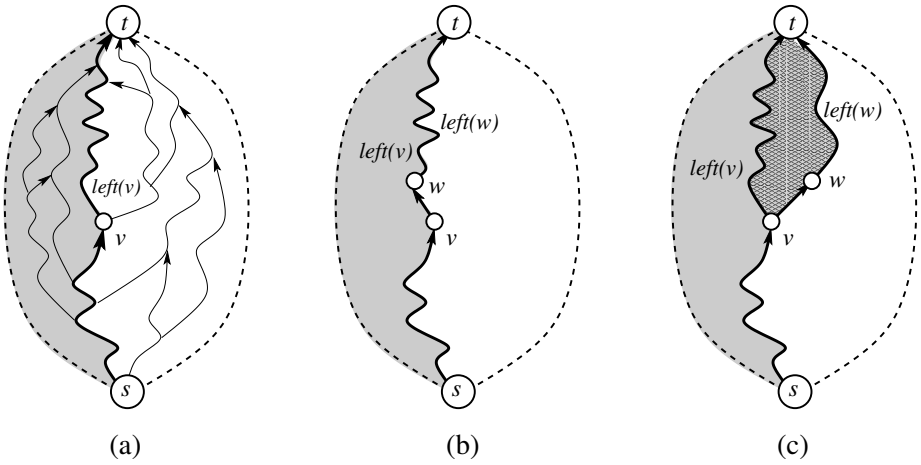


Figure 11: (a) The directed acyclic graph G_s, (b)–(c) illustration for the algorithm for finding a suitable shortest path separator.

We consider the vertices of G_s in a topological order. Initially, we set **values**(s) to be a singleton set, consisting of the value of $|L(left(s))|$. Consider now the case when we address a vertex w of G_s, other than s. We construct **values**(w) by starting with an empty set and for each incoming edge (v, w) of w, inserting an integer corresponding to each integer in **values**(v). For each incoming edge (v, w) of w, we compute these entries to **values**(w) in one of the following two cases.

Case 1: w **is on** $left(v)$**.** In this case each feasible path for v is also a feasible path for w. Furthermore each feasible path for w passing through v is also a feasible path for v; see Fig. 11(b). For each integer $x \in$ **values**(v), we thus insert x to **values**(w).

Case 2: w **is not on** $left(v)$**.** In this case, the edge (v, w) is to the right of $left(v)$. We find a feasible path for w from a feasible path for v, followed by the edge (v, w), followed by the path $left(w)$. Moreover, each feasible path for w passing through v can be found in this way; see Fig. 11(c). Let y be the number of vertices between the paths $left(v)$ and $(v, w).left(w)$, including those on $left(v)$, but not on $left(w)$. Here $(v, w).left(w)$ denote the path consisting of the edge (v, w) followed by the path $left(w)$. For each integer $x \in$ **values**(v), we thus insert $x + y$ to **values**(w). This completes the description of the procedure to compute **values**(v).

Statement 1 *For each vertex w of G_s,* ***values****(w) enumerates the values of $|L(P)|$ for all feasible paths P for w.*

Proof: The claim is true for s since the only feasible path for s is $left(s)$ and $|L(left(s))| \in$ **values**(s). For a vertex w other than s, each feasible path for w is formed by a path from s to v, followed by the edge (v, w), followed by $left(w)$

for a vertex v with an outgoing edge to w. All such paths are addressed in the dynamic programming algorithm that computes **values**(w) for each vertex w of G_s. □

Since each shortest path in G is a directed path in G_s and by definition, is a feasible path for t, **values**(t) enumerates the values of $|L(P)|$ for each shortest path P in G. For each integer $x \in$ **values**(v) for a vertex v of G_s, we can store a pointer to the predecessor of v for the corresponding feasible path for v. Using these pointers we can easily reconstruct the shortest path P for which a particular value of $|L(P)|$ is achieved. The algorithm can be implemented to run in $O(n^2)$ time as follows. The entries in **values**(v) for a vertex v are integers between 1 and n, so it is sufficient to maintain a table of size $n \times n$. In the first step, we perform Dijkstra's algorithm in time $O(n \log n)$. Computation of $|L(left(s))|$ can be done in $O(n)$ time with a suitable data structure storing the embedding of G. Afterwards for each vertex v we need to insert at most n numbers in **values**(v). Furthermore for each vertex v, the leftmost path $left(v)$ from v to t can be computed in $O(n)$ time. Finally for each vertex v and for each incoming edge (w, v), the computation of the number of vertices between the two paths $left(v)$ and $(v, w).left(w)$, takes $O(n)$ time in total for a vertex v with the given embedding. The total running time is thus $O(n^2)$. □

5.2 Vertex Distribution in Maps with L-Shaped Polygons

Here we address the global problem: finding shortest-path cuts in all the clusters simultaneously, so that we minimize the maximum imbalance over all clusters. Note that it is not sufficient to select pairs of end vertices s and t of the cuts separately for each polygon to minimize the imbalance in each of them since the choice of end-vertices in one polygon may influence the choice of end-vertices in others. However we show next that this influence is limited. Let P_i be an L-shaped polygon that contains a cluster G_i. Recall from the previous sections that we identify two corners of P_i: the only concave corner and the convex corner opposite to the concave corner. For the remaining part of the section we will call these two corners *left-corner* and *right-corner* of P_i, where the left corner has smaller x-coordinate than the right corner. Then we find a shortest path cut in G_i between two end-vertices s and t and place this shortest path along the straight-line between the two corners of P_i. Again for convenience we call the two end-vertices *left* and *right* end-vertices where the left (resp. right) end-vertex is placed near the left (resp. right) corner of P_i. The shared boundary between a particular polygon Q and its adjacent polygons in a map induces a set of intervals as we circularly walk across the boundary of Q. Since the dual of M preserves the embedding of the input graph, these intervals on the boundary of Q naturally define a set of intervals on the outer vertices of the cluster corresponding to Q. Each inter-cluster edge incident to an outer vertex of the cluster must pass through the corresponding interval of the boundary of Q avoiding edge-region crossings. We now have the following lemma.

Lemma 6 *Let L and L' be two adjacent L-shaped polygons in the map and let C and C' be the two corresponding clusters. Then the choice of the right end-vertex of a cut in C and the choice of the left end-vertex of a cut in C' depend on each other if and only if the right-corner of L and the left-corner of L' are internal points of the same interval in the boundary of L and L'.*

Proof: Suppose s and t denote the two end-vertices of a cut of C such that s lies near the left-corner of L and t lies near the right-corner of L. Similarly define s' and t' for L'. Assume without loss of generality that the concave corner of L is its right-corner. Then the only polygon that can influence the choice of t is the one that shares an interval of the boundary with L containing the concave corner (right-corner) of L. Thus in order for s' to influence the choice of t, it is necessary that L' shares an interval of its boundary that contains the right-corner of L. Now, depending on whether the concave corner of L' is its left-corner or right-corner, the choice of s' can be influenced by the choice of t in one of the following two ways.

(a) The concave corner of L' is its right-corner and the left-corner of L' coincides with the right-corner of L. In this case, the left-corner of L' and the right-corner of L is a common point, which is an internal point of the common boundary of L and L'; see Fig. 12(a).

(b) The concave corner of L' is its left-corner and the common boundary between L and L' contains this point. Thus the right-corner of L and the left-corner of L' are internal points of this common boundary in this case too; see Fig. 12(b).

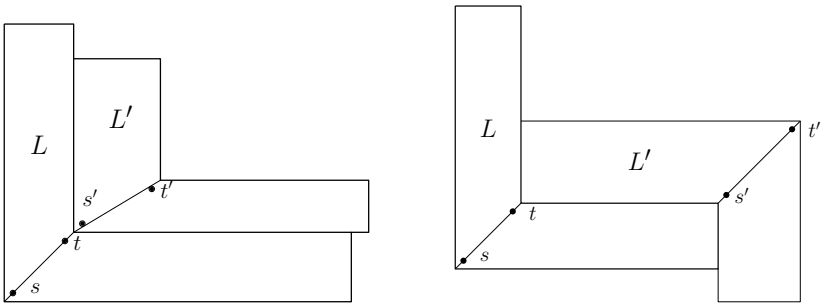


Figure 12: The two cases for the dependency between the choice of end-points of a cut

Thus in both cases, for the choice of t in C and the choice of s in C' depends on each other only if the right-corner of L and the left-corner of L' are internal points of the same interval in the boundary of L and L'. We will now show that if this is indeed the case, the choices of the two end-points of the cuts in C and C' are in fact dependent. We again assume that the concave corner of L is

its right-corner. In case it is the left-corner of L, we can show the dependency between the choices in a similar way.
Case 1: The concave corner of L' is its right-corner. In this case, the choices of t and s' must be such that either t and s' are adjacent or if there is no such edge, one can insert (t, s') without introducing any crossings.
Case 2: The concave corner of L' is its left-corner. In this case, the choice of t and s' must be such that s' lies to the right of all the neighbors of t in C' and t lies to the left of all the neighbors of s' in C. □

Lemma 6 gives a necessary and sufficient condition for two L-shaped polygons to influence the choice of the end-points of the cuts of each other. This dependency can be expressed in the directed *dependency graph* $D = (P, E_P)$, where P is the set of all L-shaped polygons and for $L, L' \in P$, there is a directed edge from L to L' in E_P when the choice of the right end-point of a cut in L influences the choice of the left end-point of a cut in L'. We can represent each edge (L, L') of graph D by drawing a directed line from the right-corner of L to the left-corner of L', with all edges pointing to the right; hence D is acyclic. Since the choice of left-corner and right-corner of a polygon may affect the choice in at most one polygon each, the maximum degree of a vertex in D is two and each component of D is either a single vertex or a path. Here for a given pair s and t in a cluster, we define $w(s,t) = |A_1/A_2 - |L(P)|/|R(P)||$ to be the smallest imbalance imposed by shortest paths from s to t. The following theorem shows that we can minimize the maximum imbalance over all the clusters.

Theorem 4 *Let G be a connected c-planar graph, G_C be the cluster-graph of G and let M be a rectilinear map of G_C with six-sided polygons such that M represents the contact map of G_C. Then one can split the regions of M in $O(n^4)$ time into convex shapes and distribute the vertices and faces of the clusters within the regions, such that the maximum imbalance imposed by a shortest path is minimized.*

Proof: We use the dependency graph to find cuts in all the clusters simultaneously such that the maximum smallest imbalance for the clusters is minimized. For this purpose, we refine this dependency graph D to capture all possible cuts. Before that we need to define the notion of compatible pairs of vertices in two adjacent clusters. Informally, compatible pairs of vertices are those that influence dependency on two clusters. More formally, let C and C' be two clusters with some inter-cluster edge between them such that the choice of one end-point of the cut in C depends on the choice of one end-point of a cut in C' and vice versa. Let L and L' be the two L-shaped polygons corresponding to C and C' respectively. Without loss of generality, assume that the concave corner of L is its right-corner. Then the compatible pairs between C and C' are defined in two cases.
Case 1: The concave corner of L' is its right-corner. In this case, the compatible pairs of vertices between C and C' are all pairs (t, s') such that t is a vertex in C, s' is a vertex in C' and either t and s' are adjacent or if there is no such edge, one can insert (t, s') without introducing any crossings.

Case 2: The concave corner of L' is its left-corner. In this case, the compatible pairs between C and C' are all pairs (t, s') where t is a vertex in C, s' a vertex in C', s' lies to the right of all neighbors of t in C' and t lies to the left of all neighbors of s' in C.

We refine the dependency graph as follows. We consider each component of D independently and find an optimum path for that component. We start with the case, where the component is a single vertex corresponding to polygon L. In that case, L is partitioned into two convex pieces and we have to choose the best pair (s, t). We create an artificial source S and sink T, and add an edge from S to each possible candidate s, and an edge from each possible candidate t to T. Then we insert edges (s, t) for the different pairs s and t with weights $w(s, t)$, where $w(s, t)$ denotes the smallest imbalance imposed by shortest paths from s to t (defined in the previous section). A min-max path computation looking for the path with the smallest maximal weight on it gives the best cut for this single component.

In the case where the component consists of a path of length one or more, the direction of edges of the path gives an order of the vertices (representing L-shaped polygons) on this path. Specifically, there is exactly one vertex L_0 in the path with no incoming edge and exactly one vertex L_f with no outgoing edge. Once again we create an artificial source S and sink T, and add an edge from S to each possible candidate s of the cluster for L_0, and analogously from each possible candidate t of the cluster for L_f to T. For each edge (L, L') of the path, we also insert an edge between each compatible pair (t, s') where t is a vertex in L and s' is a vertex in L'. Set the weight of each such edge (t, s') to be zero. Finally for each vertex L on the path, let C be the corresponding cluster. We insert edges (s, t) with weight $w(s, t)$ between all possible candidates for s and t found in a similar way as described in the previous section. The min-max path computation from S to T once again delivers the desired result, namely the path with the minimal largest weight on an edge.

The min-max path can be computed in $O(n)$ time using the algorithm in [8]. Thus, determining the imbalance for each pair s, t in each cluster dominates the running time. Since there are at most $O(n^2)$ possible s, t pairs and computing $w(s, t)$ for each takes $O(n^2)$ time, the total running time is $O(n^4)$. □

6 Conclusion and Future Work

We showed that fitting planar graphs on planar maps is NP-hard, even when the map consists of only rectangles and each region contains a single vertex. We then presented two sufficient conditions for the construction of planar straight-line fitting on rectangular maps, for c-planar graphs with biconnected clusters. These conditions are tight, in the sense that violating them makes it possible to construct counterexamples. Finally we gave a rather restricted set of sufficient conditions for fitting planar graphs on maps with non-convex regions, where we also gave an algorithm for finding a fitting with a "balanced distribution" of the vertices.

For fitting planar graphs on maps with non-convex rectilinear regions, we considered only polygons with at most 8 corners, in particular, T-shaped and L-shaped polygons along with the rectangles. The reason is that any planar graph can be represented as contact maps consisting of these three types of polygons. We therefore did not consider Z-shaped polygons (although it also contains 8 corners) or polygons with even higher number of corners on them. However it is worth-mentioning that an approach similar to the one in Theorem 3 can compute a fitting of doubly-interconnected and biconnected c-planar graphs on maps containing Z-shaped polygons as well.

One interesting variant of the problem is when we allow bends on the inter-cluster edges. A fitting of any c-planar graph with at most 2 bends per inter-cluster edges can be computed as a simple corollary of Lemma 3. Indeed we can subdivide each inter-cluster edge twice and include the two degree-2 vertices obtained from these subdivision in the two incident clusters. Then these degree-2 vertices provide the bend points in the drawing. However one can also find a fitting with at most 1-bend per inter-cluster edges in the following simple approach: allocate a small circle $circ_i$ in the center of each rectangle R_i to place the boundary vertices for each cluster G_i. We then draw the inter-cluster edges by placing the single bend on the door between the two corresponding rectangles and finally complete the drawing of each cluster G_i inside the circle $circ_i$ by applying the algorithm for drawing a graph with a prescribed convex outer face [9].

The work in this paper raises several open problems; a few are summarized below:

(i) Our proof for the NP-hardness of fitting involves the use of skinny rectangles; it is natural to ask whether the problem becomes easier if all regions are "fat".

(ii) An interesting variant of the problem of fitting on rectangular maps is when only the map topology is given but one can change the size and the aspect ratio of the rectangular regions so that a fitting is possible.

(iii) Our sufficient conditions for fitting on a rectangular map addresses only c-planar 2-connected graphs. For fitting on more general rectilinear maps, our sufficient conditions are even more restricted. It would be worthwhile to investigate whether these conditions can be relaxed.

(iv) Another interesting question is whether an exact bound on vertex resolution for fitting on maps with non-convex rectilinear polygons can be guaranteed.

Acknowledgments

We thank the two anonymous referees for useful suggestion and for providing us with sketch about fitting graphs with a bend per each inter-cluster edge.

References

[1] P. Angelini, F. Frati, and M. Kaufmann. Straight-line rectangular drawings of clustered graphs. *Discrete & Computational Geometry*, 45(1):88–140, 2011. doi:10.1007/s00454-010-9302-z.

[2] D. Avis. Generating rooted triangulations without repetitions. *Algorithmica*, 16(6):618–632, 1996. doi:10.1007/BF01944353.

[3] G. D. Battista, G. Drovandi, and F. Frati. How to draw a clustered tree. *Journal of Discrete Algorithms*, 7(4):479–499, 2009. doi:10.1016/j.jda.2008.09.015.

[4] G. D. Battista and F. Frati. Efficient c-planarity testing for embedded flat clustered graphs with small faces. *Journal of Graph Algorithms and Applications*, 13(3):349–378, 2009. doi:10.7155/jgaa.00191.

[5] M. W. Bern and J. R. Gilbert. Drawing the planar dual. *Information Processing Letters*, 43(1):7–13, 1992. doi:10.1016/0020-0190(92)90022-N.

[6] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In *Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42, 2000. doi:10.1007/978-3-7091-6783-0_4.

[7] A. L. Buchsbaum, E. R. Gansner, C. M. Procopiuc, and S. Venkatasubramanian. Rectangular layouts and contact graphs. *ACM Transactions on Algorithms*, 4(1), 2008. doi:10.1145/1328911.1328919.

[8] P. M. Camerini. The min-max spanning tree problem and some extensions. *Information Processing Letters*, 7(1):10–14, 1978. doi:10.1016/0020-0190(78)90030-3.

[9] E. Chambers, D. Eppstein, M. Goodrich, and M. Löffler. Drawing graphs in the plane with a prescribed outer face and polynomial area. *Journal of Graph Algorithms and Applications*, 16(2):243–259, 2012. doi:10.7155/jgaa.00257.

[10] J. W. Coffey, R. R. Hoffman, and A. J. Cañas. Concept map-based knowledge modeling: perspectives from information and knowledge visualization. *Information Visualization*, 5(3):192–201, 2006. doi:10.1057/palgrave.ivs.9500129.

[11] C. Duncan, E. Gansner, Y. Hu, M. Kaufmann, and S. G. Kobourov. Optimal polygonal representation of planar graphs. *Algorithmica*, 63(3):672–691, 2012. doi:10.1007/s00453-011-9525-2.

[12] P. Eades, Q.-W. Feng, X. Lin, and H. Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006. doi:10.1007/s00453-004-1144-8.

[13] S. Felsner. Rectangle and square representations of planar graphs. In J. Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 213–248. Springer, 2012. doi:10.1007/978-1-4614-0110-0_12.

[14] Q.-W. Feng, R. F. Cohen, and P. Eades. How to draw a planar clustered graph. In D.-Z. Du and M. Li, editors, *1st Annual International Conference on Computing and Combinatorics (COCOON)*, volume 959 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 1995. doi:10.1007/BFb0030816.

[15] Q.-W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. In P. G. Spirakis, editor, *3rd Annual European Symposium on Algorithms (ESA)*, volume 979 of *Lecture Notes in Computer Science*, pages 213–226. Springer, 1995. doi:10.1007/3-540-60313-1_145.

[16] É. Fusy. Transversal structures on triangulations: A combinatorial study and straight-line drawings. *Discrete Mathematics*, 309(7):1870–1894, 2009. doi:10.1016/j.disc.2007.12.093.

[17] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988. doi:10.1145/42411.42414.

[18] X. He. On floor-plan of plane graphs. *SIAM Journal of Computing*, 28(6):2150–2167, 1999. doi:10.1137/S0097539796308874.

[19] S.-H. Hong and H. Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discrete Applied Mathematics*, 156(12):2368–2380, 2008. doi:10.1016/j.dam.2007.10.012.

[20] Y. Hu, E. R. Gansner, and S. G. Kobourov. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications*, 30(6):54–66, 2010. doi:10.1109/MCG.2010.101.

[21] T. Kamada. *Visualizing Abstract Objects and Relations*, volume 5 of *World Scientific Series in Computer Science*. 1989.

[22] M. Kerber. Embedding the dual complex of hyper-rectangular partitions. *Journal of Computational Geometry*, 4(1):13–37, 2013.

[23] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992. doi:10.1137/0405033.

[24] K. Koźmiński and E. Kinnen. Rectangular duals of planar graphs. *Networks*, 15(2):145–157, 1985. doi:10.1002/net.3230150202.

[25] C.-C. Liao, H.-I. Lu, and H.-C. Yen. Compact floor-planning via orderly spanning trees. *Journal of Algorithms*, 48(2):441–451, 2003. doi:10.1016/S0196-6774(03)00057-9.

[26] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

[27] B. Shneiderman. Tree visualization with tree-maps: 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992. doi:10.1145/102377.115768.

[28] P. Ungar. On diagrams representing maps. *Journal of London Mathematical Sociery*, 28:336–342, 1953. doi:10.1112/jlms/s1-28.3.336.

[29] K.-H. Yeap and M. Sarrafzadeh. Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM Journal on Computing*, 22(3):500–526, 1993. doi:10.1137/0222035.