Fitting Planar Graphs on Planar Maps

Md. J. Alam[1], M. Kaufmann[2], S. G. Kobourov[1], T. Mchedlidze[3]

[1] Department of Computer Science, University of Arizona, USA
[2] Institute for Informatics, University of Tübingen, Germany
[3] Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Germany

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. We generalize our techniques to nonconvex rectilinear polygons, where we also address the problem of effective distribution of the vertices inside the map regions.

1 Introduction

Geographic maps often contain relational information between entities in the map, such as roads connecting cities in a country; see Fig. 1. Recently, there has been increased interest in visualizing non-geographic data with the help of the map metaphor. In this setting, fitting the data and showing various connections between the data points becomes a non-trivial problem. With this in mind, we study the problem of *fitting planar graphs on planar maps*, subject to natural requirements, such as avoiding edge crossings and guaranteeing that edges between points in the same region remain in that region.

Fitting planar graphs on planar maps is related to cluster planarity [6]. In cluster-planar drawing we are given the graph along with a clustering and 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 a given cluster are placed in their corresponding region, and there are no edge-crossings or edge-region crossings (i.e., edges that cross a region more than once).

In our setting (fitting graphs on maps), we are given both the graph and the regions in the plane, and must draw the clusters in their corresponding regions. The regions form a proper partition of the plane, such that the adjacency between two clusters is represented by a common border between their regions.

Feng *et al.* define c-planarity as planarity for clustered graphs [7]. For clustered graphs in which every cluster induces a connected subgraph, c-planarity can be tested in quadratic time. Algorithms for creating regions in the plane in which to draw c-planar graphs have also been studied. Eades *et al.* [5] present 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 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 [12] represent hierarchical information by means of space-filling tilings, allocating area in proportion to some metric. GMap [9] uses the geographic map metaphor

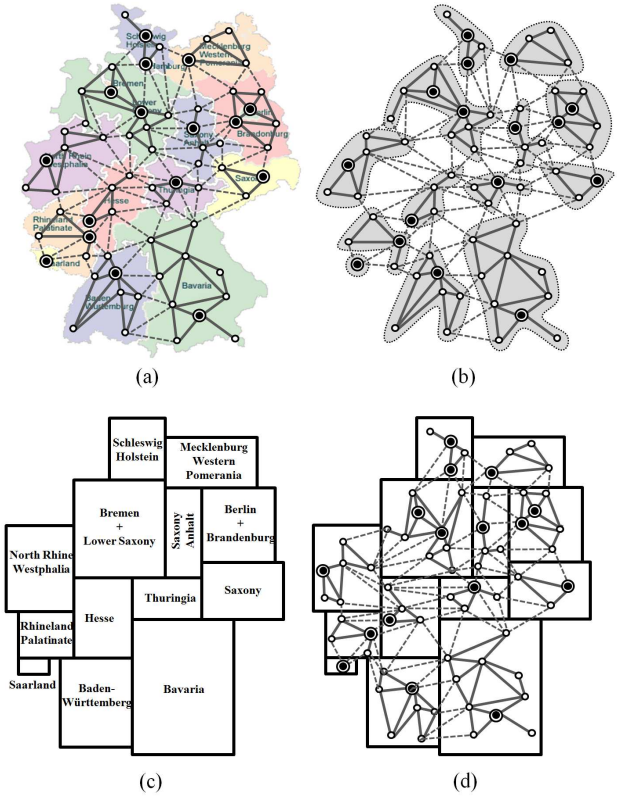

Fig. 1. (a) A map of Germany; (b) a state-based clustering of cities; (c) a rectangular map of the cluster-graph; (d) a straight-line plane fitting of the graph on the map.

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 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 [4]. 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 [8]. If the rectilinear polygons are restricted to rectangles, the class of planar graphs that allows such *rectangular duals* is completely characterized [2].

Our Contributions: We first consider the question of testing whether a given planar clustered graph fits in 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 is always possible, and show how to generalize to rectilinear (not just rectangular) maps.

In particular, we describe an efficient algorithm for distributing vertices appropriately in the case of maps with non-convex polygons.

2 Preliminaries

Let $G = (V, E)$ be a planar clustered graph, with vertex set V partitioned into disjoint sets $\mathcal{V} = \{V_1, \ldots, V_k\}$. Let E_i, for each i, $1 \leq i \leq k$ be the set of edges of G between two vertices of V_i and let E_{inter} be the set of all the remaining edges of E. 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*. A *cluster-graph* of G is the graph $G_C = (V_C, E_C)$ where V_C contains a vertex v_i for each cluster G_i of G, $1 \leq i \leq k$ and the edge $(v_i, v_j) \in E_C$, $1 \leq i, j \leq k$ if there exists an edge (u, w) in G, so that vertex u belongs to cluster G_i and vertex w belongs to cluster G_j. Set $\mathcal{V}$ is referred to as *clustering* of G, which is said to be *connected* if each of G_i, $1 \leq i \leq k$, is a connected graph. A *drawing of a 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 that contains only the vertices of G_i and such that if there is an edge e between two vertices of G_i then the drawing of e is completely contained in R_i. An edge e and a region R have an *edge-region crossing* if the drawing of e crosses the boundary of R more than once. A drawing of a clustered graph is *c-planar* if there are no edge crossings or edge-region crossings. If a clustered graph 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 on 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 its corresponding polygons have a non-trivial common boundary. Given a planar graph G_M, a polygonal map M, such that G_M represents the dual graph of M, is called a *contact map* of G_M. Assume that we are given a map M, a planar clustered graph $C = (G, \mathcal{V})$ so that M represents a contact map of the cluster-graph G_C. In this paper we are interested in determining whether each cluster G_i of C can be drawn inside its corresponding polygon in M, so that there are no edge crossings and there are no edge-region crossings. In case such a drawing exists we say that clustered graph C has a *planar fitting* on map M. If the resulting drawing is straight-line we talk about *straight-line planar fitting*.

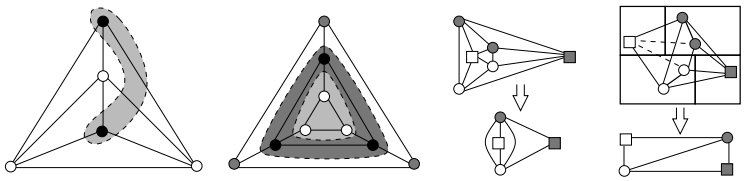

Fig. 2. (a) A graph with disconnected clusters that has no straight-line planar fitting; (b) a non c-planar graph; (c) a clustered graph and a rectangular map with incompatible embeddings, and without straight-line planar fitting.

Necessary Conditions: It is natural to consider all planar graphs, regardless of the clustering they come with, but there are several necessary conditions, in addition to planarity, without which there can be no planar fitting. The first necessary condition is the connectivity of the clusters. This necessity can be illustrated with the graph K_5 minus an edge partitioned into two clusters so that one of them is disconnected. It is easy to see that this graph does not have a straight-line planar fitting on any map representing the cluster-graph (e.g., two adjacent rectangles); see Fig. 2(a).

The second necessary condition is c-planarity. Even if the graph is planar and its clusters are connected, the resulting cluster graph need not be c-planar; see Fig. 2(b). Fortunately, there is a simple characterization of c-planarity for the case when the clusters are connected which can be tested in quadratic time [7]. The characterization states that a graph $G = (V, E)$ with a connected clustering $\mathcal{V} = \{V_1, \ldots, V_k\}$ is c-planar if and only if G is planar and there exists a planar drawing of G, such that for each V_i, all the vertices and edges of $G \setminus G(V_i)$ are in the outerface of the drawing of $G(V_i)$.

The third necessary condition is compatibility of the graph and map embeddings. Specifically, the embedding of G and hence the embedding of the cluster-graph G_C are given, and they are *compatible* with the given planar map M, that is, the dual of M must represent the same graph as G_C and it must have the same embedding as G_C. Otherwise there is no straight-line planar fitting; see Fig. 2(c).

Given these necessary conditions, in the rest of the paper we consider only connected c-planar graphs that have an embedding compatible with the given map.

3 Fitting on a Rectangular Map

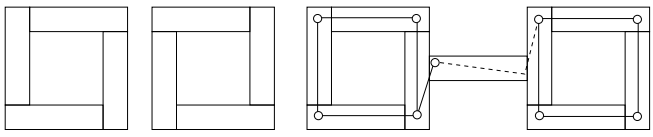

Fig. 3. Wheel maps cw (a) and ccw (b). A clustered graph and map with no fit (c).

In this section we address the problem of deciding whether a connected c-planar graph G has a straight-line planar fitting on a given rectangular map M, assuming compatible embeddings of G and M. We first show that this is not always possible. To construct the example we use a *wheel map*, which contains a rectangle surrounded by four "thin rectangles" (with aspect ratio of at least 4); see Fig. 3(a)–(b). A thin rectangle is *horizontal* if its smaller dimension is its height and it is *vertical* otherwise.

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 $v_i \in V_i$, $v_j \in V_j$, $1 \leq i, j \leq k$. Call the common boundary between the polygons representing V_i and V_j in M, the *door* for the edge (v_i, v_j). In wheel maps for each thin rectangle, we can distinguish an *entry door* (one that contains a complete side of the rectangle) and an *exit door* (one that contains a complete side of a neighboring thin rectangle). We call a wheel map *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. 3(a). A *counterclockwise (ccw) wheel* is defined analogously; see Fig. 3(b). Let $v_i \in V_i$ be a vertex of G and let R_i be the rectangle representing V_i in M. In a straight-line planar fitting of G, we say that v_i is placed *near a door* of R_i when the distance between v_i and its closest point on the door is less than the smaller side of R_i.

Lemma 1. *Let W be a wheel map and G its dual graph. In a straight-line planar fitting of G, all vertices in the thin rectangles lie near the entry doors, or all lie near the exit doors. There exists a straight-line planar fitting in each case.*

Proof: Consider first an arbitrary straight-line planar fitting Γ of G on W. We first show that each vertex inside a thin rectangle must be placed near one of the two doors. Consider the rightmost vertical rectangle R of W. The vertex inside it has two edges incident to the vertices in the two horizontal rectangles. Thus if the vertex inside it is not near neither of its doors, then the vertices in both the horizontal rectangles must be placed near the door adjacent to R. In this case, there is no feasible placement of the vertex in the leftmost vertical rectangle from where both the vertices in the two horizontal rectangles are visible. Similar arguments show that in each of the thin rectangles, the vertex must be placed near one of the two doors. If in one of the thin rectangles, the vertex is placed near the entry door (resp. exit door), then the previous argument confirms that this placement will force the placement of all the vertices inside the thin rectangles near the corresponding entry doors (resp. exit doors).

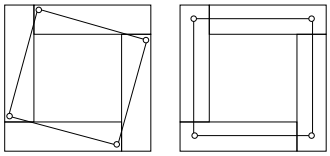

Fig. 4. Straight-line planar fittings on wheel maps.

On the other hand, it is easy to find a valid fit where the vertices inside thin rectangles are all near entry doors (or all near exit doors); see Fig. 4. □

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

Lemma 2. *There exist planar clustered graph G and compatible rectangular map M, so that there is no straight-line planar fitting of G on M.*

Proof: Consider a rectangular map M made of two wheel maps joined together by a thin horizontal rectangle, called a *bridge*; see Fig. 3(c). Let G be the dual of M: two 4-wheels connected by a path of length two. We show that G does not have a straight-line planar fitting on M. Let us assume that there exists a straight-line planar fitting Γ of G on M. Then by Lemma 1, all the vertices inside the thin rectangles of both the wheels must be placed near the doors. But then, there is no feasible position for the vertex that represents the bridge. In particular, if placed near one of the doors of the

bridge it cannot "see" its neighbors in the other wheel and vice versa. As the bridge is thin by construction, there is no place for the vertex representing the bridge so that it sees its neighbors in both wheels. □

3.1 Fitting is NP-Hard

We show that deciding if a given planar graph fits 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 [11]. *Planar-3-SAT* is defined analogously to 3-SAT with the additional restriction that the vertex-clause bipartite graph G_F for a given formula F (there is an edge (x_i, C_j) in G_F if and only if x_i or $\overline{x_i}$ appears in C_j) is planar. Knuth and Raghunatan [10] 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 G be a planar clustered graph and let M be a rectangular map, compatible with G. Deciding if G admits a straight-line planar fitting on M is NP-hard.*

Proof: We reduce an instance of Planar 3-SAT to an instance (G, M) of our problem. Let $F := C_1 \dots C_m$ be an instance of a planar 3-SAT where each literal in each clause is a variable (possibly negated) from $U = \{x_1, \dots, x_n\}$. Let Γ_F be the given planar rectilinear drawing for this instance, as defined in [10]. 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 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 near the exit doors and the literal assumes a false value when they lie near 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. In order to maintain consistency, we ensure that a true (false) value to an instance of each literal x would imply a true (false) value for each 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. 5.

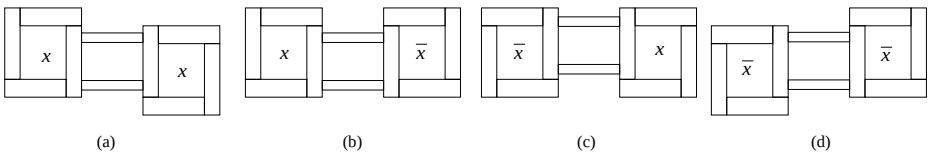

Fig. 5. Representation of variables.

For each clause $C = (x+y+z)$ of F with the corresponding vertex lying above the variables in Γ_F we draw vertical rectangles l_x^C, l_y^C and l_z^C from the topmost rectangles of the wheels for x, y and z, respectively, attached near the exit door. (The case when

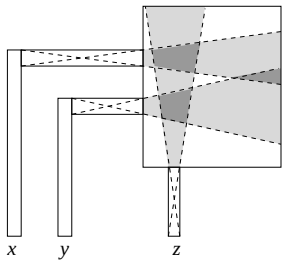

Fig. 6. Clause representation.

the vertex lies below the variables in Γ_F is analogous.) Then we draw a rectangle R for the clause and attach these three thin rectangles l_x^C, l_y^C and l_z^C to R. For z we attach the vertical rectangle l_z^C to the bottom of R, while for each of x and y, we attach horizontal rectangles to R that also touch the vertical rectangles l_x^C and l_y^C coming from x or y, respectively. We attach these thin rectangles to R in such a way that the three visibility regions do not have a common intersection, while each two of them have a common intersection; see Fig 6. Specifically, we adjust the width of l_z^C and attach it in a position of R such that its visibility region is only in the left half of R. We also adjust the heights of the horizontal rectangles adjacent to l_x^C and l_y^C and adjust the vertical distance between them, so that their visibility regions do not intersect in the left half of R and they do intersect in the right half. Finally we fill up all the unused regions in the map with the appropriate number of rectangles.

Lemma 3. *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 now show that F is satisfiable, i.e., there exists a truth assignment for all the variables of F such that for each clause $C = (x, y, z)$ of F, at least one of x, y and z is true. Let W_x be the wheel for x. If the vertices in W_x are placed near the entry doors, then by the arguments in Lemma 1, we can show that the vertex in l_x^C is placed far from the door adjacent to the rectangle R for C. Thus this vertex can see only the highlighted visibility region inside R in Fig. 6. On the other hand, if the vertices in W_x are placed near the exit doors, then the vertex in l_x^C can be placed near the door adjacent to the rectangle R for C and can see the entire interior of R. This is true for each of the three literals. Since the visibility regions of the three literals have no common intersection, it must be the case that the vertices in the wheel for at lease one of x, y and z are placed near the exit door. We make each such literal true. Note that this assignment will not have conflicts, because of the way all the wheels for a particular variable are attached with each other during the construction. Furthermore, this assignment will satisfy F.

Conversely if F is satisfiable, then for each clause $C = (x, y, z)$ of F, at least one of x, y and z is true. Without loss of generality, assume that x is true. Then we place the vertices in the wheel of x near the corresponding exit doors. With this placement, the vertex in l_x^C can be placed near its door adjacent to the rectangle R for C so that it can see the entire interior of R. Then we place the vertex for R in the intersection of the visibility regions of y and z. This placement will ensure that we can place the vertices in the wheel for y and z either near entry doors or exit doors and will still be able to place all the vertices in all the rectangles from this wheel to R without violating linearity of the edges. This yields the desired straight-line planar fitting of G on M. □

The proof of Lemma 3 completes the NP-hardness proof. Fig. 7 illustrates a 3-SAT formula, its planar 3-SAT realization using the additional conditions of Knuth and Raghunatan [10] and the corresponding instance for the map fitting problem (the rectangles to fill up the holes are not shown).

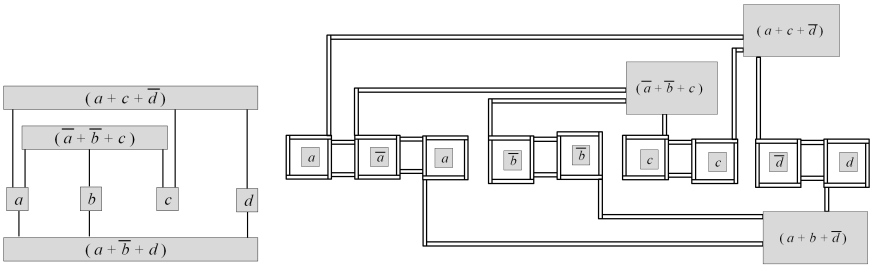

Fig. 7. Planar 3-SAT instance and corresponding map fitting instance.

□

3.2 Sufficient Conditions for Fitting

The counterexample of Lemma 2 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. In contrast to this we show the following:

Lemma 4. *Let G be a planar clustered graph where V_1, V_2, ..., V_k induce the clusters of G. Suppose each cluster of G is biconnected. Let M be a rectangular map compatible with G. 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 G has a straight-line planar fitting on M.*

Proof:

(a) We first assume that for each vertex v of G, all the vertices adjacent to v through an inter-cluster edge lie on the same cluster. Let R_i and R_j be two incident rectangles of M and let G_i and G_j be the clusters corresponding to them, $1 \leq i, j \leq k$. Let $v_{i_1}, \ldots, v_{i_k} \in V_i$ and $v_{j_1}, \ldots, v_{j_k} \in V_j$ be the incident to each other vertices of V_i and V_j, respectively, taken in the order they appear in the outer boundary of

G_i and G_j, respectively. For each R_i, $1 \leq i \leq k$, we define O_i to be the oval inscribed in R_i. We 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. Next we place vertices $v_{i_1}, \ldots, v_{i_k}$ of V_i and $v_{j_1}, \ldots, v_{j_k}$ 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 induced by these vertices, cross the common border of R_i and R_j. As a result of the above procedure we have placed some of the vertices of the outer boundary of G_i on the oval O_i, $1 \leq i \leq k$. Since each placed vertex is adjacent to a unique cluster, its position is uniquely defined. We distribute the rest vertices of the boundary of G_i on O_i, so that the order of the vertices in total is the same as in the boundary of G_i. Since the resulting drawing of the outer boundary of G_i, $1 \leq i \leq k$ is convex, we can apply the algorithm of "drawing graph with a prescribed outer face" [3] to complete the drawing of each cluster.

(b) We now 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.

Intuitively the proof is based on the achievement of the following goals:

(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. 8(a). For each vertex v_i of G_C, we draw a small circle $circ(G_i)$ centered at it in 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 straight-line, do not cross neither the boundaries of the clusters, nor each other; see Fig. 8(b).

(4) Since the boundary of each G_i is a convex polygon we can apply the algorithm for "drawing graph with a prescribed convex outer face" [3] to complete the drawings of the clusters; see Fig. 8(c).

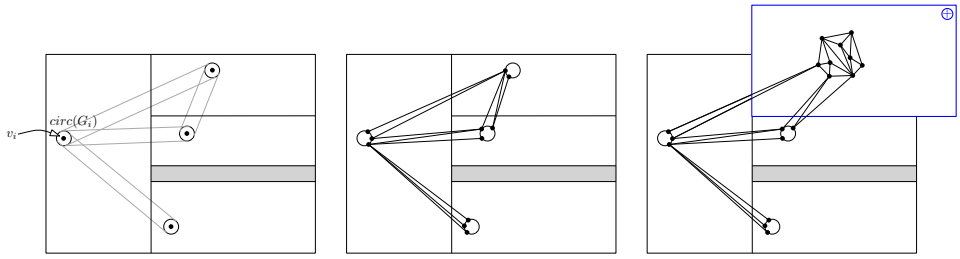

Fig. 8. (a) Drawing of G_C where each edge is represented by a strip of width $\varepsilon > 0$. (b) Placement of the boundary vertices of the clusters on corresponding circles. (c) Step 4 of the proof of Lemma 4.

For the first step of the proof, we show that $G_C = (V_C, E_C)$ has a 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. We 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. We continue with 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 components of G_C. Let $v_0, \ldots, v_m \in V_C$ induce the unique cycle of G_C and let $R_0, \ldots, R_m$ denote the corresponding to them rectangles, so that R_0 and R_m are adjacent. We place v_i, $0 \leq i \leq m$ inside R_i such that for any point $p \in R_{(i+1) \bmod m}$, segment $\overline{pv_i}$ crosses the common boundary of R_i and $R_{(i+1) \bmod m}$. Since this was the unique cycle of G_C, the removal of vertices $v_0, \ldots, v_m$ give several trees. We root these trees at the vertices $v_0, \ldots, v_m$, to which they are adjacent and apply the procedure descried in the first part of the proof. This completes the construction of planar fitting of G_C on M.

In the rest of the proof we show how to accomplish step (3). Since the graph G is c-planar, by [6], there exists a drawing $\Gamma(G)$ of G, where all the vertices of clusters $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots, G_k$ appear on the outer face of G_i. Let v be a vertex of G_i that lie on G_i's boundary; see Fig. 9(a). Let $v_1^i, \ldots, v_{p_i}^i$ be all the neighbors of

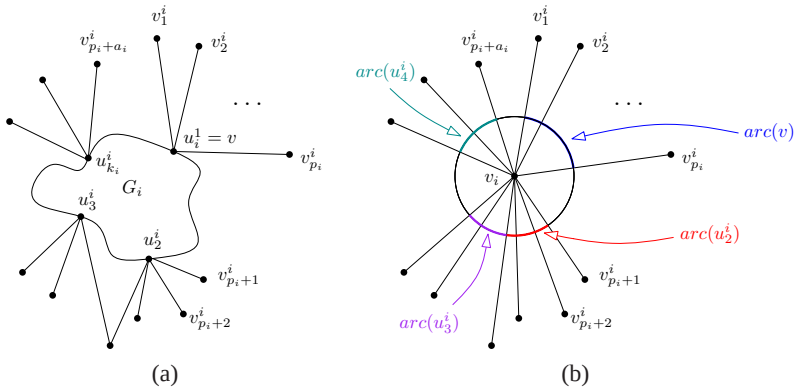

Fig. 9. Illustration for the proof of Lemma 4.

v_i that represent the clusters to which vertex v is adjacent. As, we have mentioned, $v_1^i, \ldots, v_{p_i}^i$ appear on the outer face of G_i. Assume that $v_1^i, \ldots, v_{p_i}^i$ are given in the clockwise order they appear in $\Gamma(G_C)$. We denote by $arc(v)$ 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. 9(b). Since $\Gamma(G_C)$ is a straight-line planar drawing, we have the following observation.

Observation 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^i_1,...,$u^i_{k_i}$ be the vertices of the boundary of G_i traversed in the clockwise direction. The circular arcs $arc(u^i_1), \ldots, arc(u^i_{k_i})$ appear clockwise around the $circ(G_i)$ in this specific order and are internally disjoint; see Fig. 9(b).*

Let now G_i and G_j be two clusters of G. Assume that they are connected by multiple inter-cluster edges. Let $u^i_{f_i}, \ldots, u^i_{l_i}$ (resp. $u^j_{f_j}, \ldots, u^j_{l_j}$) 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. 10(a). The edges $(u^i_{f_i}, u^j_{l_j}), (u^i_{l_i}, u^j_{f_j})$

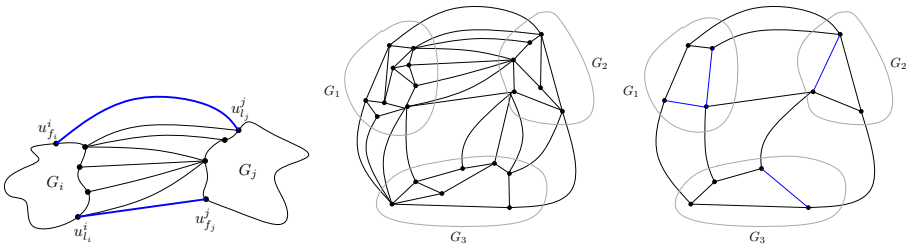

Fig. 10. (a) The bounding edges between clusters G_i and G_j are drawn with fat blue ink. (b) A clustered graph G with clusters identified by gray curves. (c) A skeleton $\mathcal{S}(G)$ of graph G depicted in figure (b).

are called the *bounding* intra-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) take the boundaries of each cluster G_i and connect them by all the bounding inter-cluster edges, (2) substitute the paths of the boundaries by edges; see Fig. 10(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. 11(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.

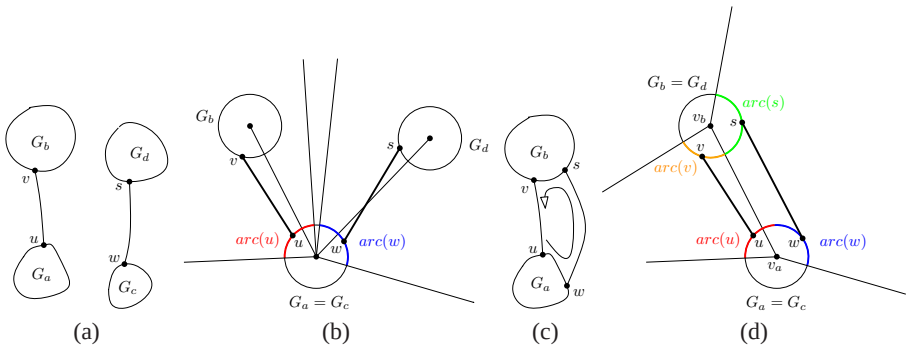

Fig. 11. (a) Two edges of G, with the end vertices belonging to distinct clusters G_a, G_b, G_c, G_d. (b) Case 2 of the proof, $G_a = G_c$. (c-d) Case 3 of the proof, $G_a = G_c$ and $G_b = G_d$.

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 1, the circular segments $arc(u)$ and $arc(v)$ are internally disjoint; see Fig. 11(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 lost of generality assume that if we traverse this cycle in this specific order then u appear before w and s before v on the boundary of G_a and G_b, respectively. By Observation 1, the circular segment $arc(u)$ 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. 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, any edges (u, v) and (w, s) are separated by the horizontal line through v_a and v_b and therefore do not cross.

We have constructed a planar straight-line drawing of $\mathcal{S}(G)$. We complete the proof explaining how to draw the rest 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 straight-line the already placed vertices of $G_i \cap \mathcal{S}(G)$ in the order they appear on the boundary of G_i. They form a convex polygon. Place the rest vertices of the boundary of G_i on the respective sides of this convex polygon. Draw the rest inter-cluster edges straight-line. It is easy to see that they do not create crossings, since they lie either in triangles or convex quadrilaterals created by the bounding edges of G.

□

4 Fitting Graphs on Rectilinear Maps

It is known that only a restricted class of planar graphs can be realized by rectangular maps. For general maximal planar graphs, 8-sided polygons (T-shapes) are necessary and sufficient for contact maps [8]. In this section, we assume that the input is a recti-

linear map, together with a c-planar graph G with planar embedding and cluster-graph G_C. The first condition that we require is that the subgraph induced by the inter-cluster edges is a matching. From Lemma 4 it follows that this condition is sufficient for rectangular maps. Now, we will extend this to L-shapes and T-shapes. We cannot directly apply the strategy of Section 3 because now we have to deal with concave corners of the regions in the map. We therefore impose our second condition: none of the clusters contains a *boundary chord*, i.e. a non-boundary edge between two boundary vertices.

To be able to apply the algorithms for drawing graph with a prescribed convex outer face [3], 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 two corresponding clusters. We now impose our last condition: there are at least two inter-cluster edges between adjacent clusters. We call these *doubly-interconnected clusters*. Note that the common boundary of two adjacent polygons consists of at most two concave corners. We place the vertices next to the common boundary on both sides of the concave corners. This ensures that the cycle spanned by the boundary vertices of the cluster is completely within the corresponding polygon and there are at most two concave corners along the cycle. Let a and b be the vertices at these corners; see Fig. 12. We choose a third boundary vertex c lying opposite a and b. Straight-line cuts between a, c and b, c define 3 convex regions. Now we compute an a, c-path and a b, c-path without any shortcut[4] so that we can place the vertices on these two paths on the two cuts between a, c and b, c. Note that such a path should not contain any other boundary vertex, already placed elsewhere. We therefore find these paths in such a way that they do not contain any boundary vertex.

Consider the shortest path l in the dual graph of a cluster between the two inner faces containing b and c. We will now find an ordered set of vertices $b = v_0, v_1, \ldots, v_f = c$ in the input graph "following" this dual path l. Starting from b, we find the next vertices on this set one after another such that all the internal vertices lie on the common boundary between two consecutive faces on l. On such a common boundary there are at least two vertices and both of them cannot be a boundary vertex since otherwise that would induce a chord edge. We will then choose a non-boundary vertex on the common boundary. Note that consecutive vertices on this set might be non-adjacent in the input graph. However, we may assume that all consecutive pairs of vertices on this set are adjacent by means of an original or a dummy edge since from the construction the assumption of such a dummy edge would not violate planarity. Thus they can be thought of forming a b, c-path. Furthermore note that this path contains no shortcut or transitive edge. The reason is that with the successive choice of vertices we are making a jump of one face at a time along l and a shortcut edge would also induce a shortcut face on l, which is a contradiction since l is a shortest path on the dual graph. In the same way we can find another such path between a and c. Together with these two paths we now have three cycles that describe convex areas, assuming that they are disjoint. (In case they are not, they have a common subpath towards c, say a c, d-path. We then draw the a, d-path, b, d-path and c, d-path on three segments 120 degree apart from each other with a common endpoint at d, again forming three convex polygons). Hence we are able to apply the algorithm in [3] directly. The same idea can be applied for an L-shaped

[4] *Shortcut* of path P is an edge between vertices nonadjacent in P.

polygon where the straight-line segment splitting this polygon into two convex parts are between the concave corner and its opposite corner of the polygon. This yields the following theorem.

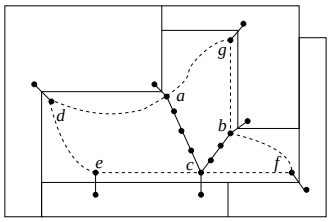

Fig. 12. Illustrating how vertices on the boundary of a T-shape are placed and how the cluster is partitioned by two paths such that it fits into three adjacent convex polygons with the paths as common boundaries.

Theorem 2. *For doubly-interconnected clustered c-planar graphs with matching inter-cluster edges, such that there is no boundary chord in any cluster, there exists a straight-line planar fitting on any compatible map with rectangular, L- or T-shaped polygons.*

Fitting a planar graph G on a rectilinear map with L- or T-shaped polygons may force a very irregular distribution of the vertices and faces of the graph, which in turn, may affect the resolution. Next we want to find a balanced distribution of vertices and faces inside the L-shaped polygons of a map. First, we consider the distribution inside only one polygon partitioned in two pieces by a straight-line cut; then we use this result to find a balanced distribution inside all the L-shaped polygons of a map.

Shortest Path Separators We first 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 placing a dummy vertex in each face. Let P be a path in the extended graph and denote by $L(P)$ and $R(P)$ the two parts induced by P in the extended graph. 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 induced by P. We want to find a shortest path P between two given vertices s and t of P containing no boundary vertex that 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 a 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 the graph. 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. 13(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. 13(a) shows a feasible path P for a vertex v and the highlighted region defines $L(P)$.

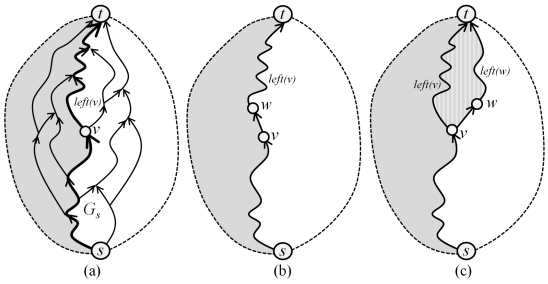

Fig. 13. (a) The directed acyclic graph G_s, (b)–(c) illustration for the algorithm for finding a suitable shortest path separator.

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. 13(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. 13(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 $left(w)$. For each integer $x \in VALUES(v)$, we thus insert $x + y$ to $VALUES(w)$.

We thus compute the sets $VALUES(v)$ for each vertex v of G_s. We now have the following lemma.

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 value of $|L(P)|$ for each shortest path P in G. The implementation can be done in time $O(n^2)$, maintaining for each vertex the $VALUES(v)$ in the range between 1 and n. □

Distributing Vertices Nicely in More General Maps Next, we address the global problem: finding cuts in all the clusters simultaneously, so that we minimize the maximum imbalance over all clusters. 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. Clearly, the choice of s and t in one polygon may influence the choice of s and t in others, but we show next that this influence is limited. 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-point of a cut in C and the choice of the left end-point of a cut in C' depend on each other if and only if the t-corner of L and the s-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-points of a cut of C such that s lies near the s-corner of L and t lies near the t-corner of L. Similarly define s' and t' for L'. Assume without loss of generality that the concave corner of L is its t-corner. Then one can see that 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 (t-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 t-corner of L. Now, depending on whether the concave corner of L' is its s-corner of t-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 t-corner and the s-corner of L' coincides with the t-corner of L. In this case, the s-corner of L' and the t-corner of L is a common point, which is an internal point of the common boundary of L and L'; see Fig. 14(a).

(b) The concave corner of L' is its s-corner and the common boundary between L and L' contains this point. Thus the t-corner of L and the s-corner of L' are internal points of this common boundary in this case too; see Fig. 14(b).

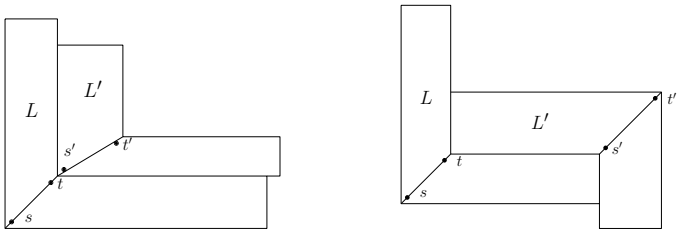

Fig. 14. The two cases for the dependency between the choice of end-points of a cuts in two L-shaped polygons.

Thus in both the case, for the choice of t in C and the choice of s in C' depends on each other only if the t-corner of L and the s-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 t-corner. In case it is the s-corner of L, we can show the dependency between the choices in a similar way.

Case 1: The concave corner of L' is its t-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 s-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 t-corner of L to the s-corner of L', with all edges pointing to the right; hence D is acyclic. Since the choice of s-corner and t-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. The following theorem shows that we can minimize the cluster imbalance.

Theorem 3. *Let G be a connected c-planar graph, G_C be the cluster-graph of G and 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 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 is as small as possible.*

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 pair 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 t-corner. Then the compatible parts between C and C' are defined in the following two cases.

Case 1: The concave corner of L' is its t-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 s-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', 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.

We will 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 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. The choice of possible candidates for s and t is done as described in the previous section. Then we insert edges (s, t) for the different pairs s and t with weights $w(s, t)$ which has been defined above. A bottleneck shortest path computation looking for the path with the smallest maximal weight (imbalance) 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 the 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 bottleneck shortest path computation from S to T once again delivers the desired result, namely the path with the minimal largest weight on an edge.

The bottleneck shortest path computation in both the cases takes $O(n \log n)$ time using a variant of Dijkstra algorithm. Thus, determining the imbalance for each pair s, t in each cluster dominates the running time. Since there can be at most $O(n^2)$ possible

s, t pairs and computing $w(s, t)$ for each of them take $O(n^2)$ time, the total running time is $O(n^4)$. □

5 Conclusions and Future Work

We showed that fitting planar graphs on planar maps is NP-hard. The proof involves skinny regions; it is natural to ask whether the problem becomes easier if all regions are "fat". We presented necessary and sufficient conditions for the construction of planar straight-line fitting on rectangular map, for a c-planar graphs with biconnected clusters. The presented sufficient conditions are tight, meaning that violating them makes it possible to construct counterexamples. It is natural to study the case where the clusters are not necessarily biconnected. Finally, we gave a rather restricted set of sufficient conditions for fitting planar graphs on maps with non-convex regions. It would be interesting to investigate whether these conditions can be relaxed. One of the most interesting questions is to study the vertex resolution of the constructed fittings. To find a bound on the vertex resolution remains open.

References

1. P. Angelini, F. Frati, and M. Kaufmann. Straight-line rectangular drawings of clustered graphs. In *Symposium on Algorithms & Data Structures (WADS)*, pages 25–36, 2009.
2. A. Buchsbaum, E. Gansner, C. Procopiuc, and S. Venkatasubramanian. Rectangular layouts and contact graphs. *ACM Transactions on Algorithms*, 4(1), 2008.
3. 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 (JGAA)*, 16(2):243–259, 2012.
4. C. Duncan, E. Gansner, Y. Hu, M. Kaufmann, and S. G. Kobourov. Optimal polygonal representation of planar graphs. *Algorithmica*, 63(3):672–691, 2012.
5. 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.
6. Q.-W. Feng, R. F. Cohen, and P. Eades. How to draw a planar clustered graph. In *Conference on Computing and Combinatorics (COCOON)*, pages 21–30, 1995.
7. Q.-W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. In *European Symposium on Algorithms (ESA)*, pages 213–226, 1995.
8. X. He. On floor-plan of plane graphs. *SIAM Journal of Computing*, 28(6):2150–2167, 1999.
9. 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.
10. D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.
11. D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.
12. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.