Optimal Polygonal Representation of Planar Graphs

E. R. Gansner[2], Y. F. Hu[2], M. Kaufmann[3], and S. G. Kobourov[4]

[1] AT&T Research Labs, Florham Park, NJ
{erg, yifanhu}@research.att.com
[2] Wilhelm-Schickhard-Institut for Computer Science, Tübingen University
mk@informatik.uni-tuebingen.de
[3] Dept. of Computer Science, University of Arizona
kobourov@cs.arizona.edu

Abstract. In this paper, we consider the problem of representing graphs by polygons whose sides touch. We show that at least six sides per polygon are necessary by constructing a class of planar graphs that cannot be represented by pentagons. We also show that the lower bound of six sides is matched by an upper bound of six sides with a linear time algorithm for representing any planar graph by touching hexagons. Moreover, our algorithm produces convex polygons with edges with slopes 0, 1, -1.

1 Introduction

For both theoretical and practical reasons, there is a large body of work considering how to represent planar graphs as *contact graphs*, i.e., graphs whose vertices are represented by geometrical objects with edges corresponding to two objects touching in some specified fashion. Typical classes of objects might be curves, line segments or isothetic rectangles, and an early result is Koebe's theorem [20], which shows that all planar graphs can be represented by touching disks.

In this paper, we consider contact graphs whose objects are simple polygons, with an edge occurring whenever two polygons have non-trivially overlapping sides. As with treemaps [3], such representations are preferred in some contexts [4] over the standard node-link representations for displaying relational information. Using adjacency to represent a connection can be much more compelling, and cleaner, than drawing a line segment between two nodes. For ordinary users, this representation suggests the familiar metaphor of a geographical map.

It is clear that any graph represented this way must be planar. As noted by de Fraysseix *et al.* [7], it is also easy to see that all planar graphs have such representations for sufficiently general polygons. Starting with a straight-line planar drawing of a graph, we can create a polygon for each vertex by taking the midpoints of all adjacent edges and the centers of all neighboring faces. Note that the number of sides in each such polygon is proportional to the degree of its vertex. Moreover, these polygons are not necessarily convex; see Figure 1.

It is desirable, for aesthetic, practical and cognitive reasons, to limit the complexity of the polygons involved, where "complexity" here means the number of sides in the polygon. Fewer sides, as well as wider angles in the polygons, make for simpler and cleaner drawings. In related applications such as floor-planning [24], physical constraints make undesirable polygons with very small angles or many sides. One is then led to consider how simple can such representations be. How many sides do we really need? Can we insist that the polygons be convex, perhaps with a lower bound on the size of the angles or the edges? If limiting some of these parameters prevents the drawings of all planar graphs, which ones can be drawn?

1.1 Our Contribution

This paper provides answers to some of these questions. Previously, it was known [12, 24] that all planar graphs can be represented using non-convex octagons. On the other hand, it is not hard to see that one cannot use triangles (e.g., K_5 minus one edge cannot be represented with triangles).

Our main result is showing that hexagons are necessary and sufficient for representing all planar graphs. For necessity we construct a class of graphs that cannot be represented using five or fewer sides. For sufficiency, we

Fig. 1. Given a drawing of a planar graph(a), we apportion the edges to the endpoints by cutting each edge in half (b), and then apportion the faces to form polygons (c).

describe a linear-time algorithm that produces a representation using convex hexagons all of whose sides have slopes 1, 0, or -1. Finally, we describe an alternative algorithm for generating convex hexagonal representations for general planar graphs that leads to $O(n) \times O(n)$ drawing area.

1.2 Related Work

As remarked above, there is a rich literature related to various types of contact graphs. There are many results considering curves and line segments as objects (cf. [13, 14]). For closed shapes such as polygons, results are rarer, except for axis-aligned (or *isothetic*) rectangles. In a sense, results on representing planar graphs as "contact systems" can be dated back to Koebe's 1936 theorem [20] which states that any planar graph can be represented as a contact graph of disks in the plane.

The focus of this paper is side-to-side contact of polygons. The algorithms of He [12] and Liao *et al.* [24] produce contact graphs of this type for any planar graph, with nodes represented by the union of at most two isothetic rectangles, thus giving a polygonal representation by non-convex octagons. By relaxing the isothetic constraint to allow angles of 45° we are able to reduce the number of sides to six, while enforcing convexity.

Although not considered by the authors, an upper bound of six for the minimum number of sides in a touching polygon representation of planar graphs might be obtained from the vertex-to-side triangle contact graphs of de Fraysseix *et al.* [7]. The top edge of each triangle can be converted into a raised 3-segment polyline, clipping the tips of the triangles touching it from above, thereby turning the triangles into side-touching hexagons. It is likely to be difficult to use this approach for generating hexagonal representations as it involves computing the amounts by which each triangle may be raised so as to become a hexagon without changing any of the adjacencies. Moreover, by the nature of such an algorithm, there would be many "holes," potentially making such drawings less appealing, or requiring further modifications to remove them.

We now turn to contact graphs using isothetic rectangles, which are often referred to as *rectangular layouts*. This is the most extensively studied class of contact graphs, due, in part, to the relation to application areas such as VLSI floor-planning [22, 31], architectural design [28] and geographic information systems [10], but also due to the mathematical ramifications and connections to other areas such as rectangle-of-influence drawings [25] and proximity drawings [1, 16].

Graphs allowing rectangular layouts have been fully characterized [26, 30] with linear algorithms for deciding if a rectangular layout is possible and, if so, constructing one. The simplest formulation [4] notes that a graph has a rectangular layout if and only if it has a planar embedding with no filled triangles. Thus, K_4 has no rectangular layout. Buchsbaum *et al.* [4] also show, using results of Biedl *et al.* [2], that graphs that admit rectangular layouts are precisely those that admit a weaker variation of planar rectangle-of-influence drawings.

Rectangular layouts required to form a partition of a rectangle are known as *rectangular duals*. In a sense, these are "maximal" rectangular layouts; many of the results concerning rectangular layouts are built on results concerning rectangular duals. Graphs admitting rectangular duals have been characterized [11, 21, 23] and there are linear-time algorithms [11, 19] for constructing them.

Another view of rectangular layouts arises in VLSI floorplanning, where a rectangle is partitioned into rectilinear regions so that region adjacencies correspond to a given planar graph. It is natural to try to minimize the complexities of the resulting regions. The best known results are due to He [12] and Liao *et al.* [24] who show that regions need not have more than 8 sides. Both of these algorithms run in $O(n)$ time and produce layouts on an integer grid of size $O(n) \times O(n)$, where n is the number of vertices.

Rectilinear cartograms can be defined as rectilinear contact graphs for vertex-weighted planar graphs, where the area of a rectilinear region must be proportional to the weight of its corresponding node. Even with this extra condition, de Berg *et al.* [6] show that rectilinear cartograms can always be constructed in $O(n \log n)$ time, using regions having at most 40 sides.

1.3 Preliminaries

Touching Hexagons Graph Representation: Throughout this paper, we assume we are dealing with a connected planar graph $G = (V, E)$. We would like to construct a set of closed simple polygons R whose interiors are pairwise disjoint, along with an isomorphism $\mathcal{R} : V \to R$, such that for any two vertices $u, v \in V$, the boundaries of $\mathcal{R}(u)$ and $\mathcal{R}(v)$ overlap non-trivially if and only if $\{u, v\} \in E$. For simplicity, we adopt a convention of the cartogram community and define the *complexity* of a polygonal region as the number of sides it has. We call the set of all graphs having such a representation where each polygon in R has complexity 6 *touching hexagons graphs*.

Canonical Labeling: Our algorithms begin by first computing a *planar embedding* of the input graph $G = (V, E)$ and using that to obtain a *canonical labeling* of the vertices. A planar embedding of a graph is simply a clockwise order of the neighbors of each vertex in the graph. Obtaining a planar embedding can be done in linear time using the algorithm by Hopcroft and Tarjan [15]. The canonical labeling or order of the vertices of a planar graph was defined by de Fraysseix *et al.* [9] in the context of straight-line drawings of planar graphs on an integer grid of size $O(n) \times O(n)$. While the first algorithm for computing canonical orders required $O(n \log n)$ time [8], Chrobak and Payne [5] have shown that this can be done in $O(n)$ time.

In this section we review the canonical labeling of a planar graph as defined by de Fraysseix *et al.* [8]. Let $G = (V, E)$ be a fully triangulated planar graph embedded in the plane with exterior face u, v, w. A canonical labeling of the vertices $v_0 = u, v_1 = v, v_2, \ldots, v_{n-1} = w$ is one that meets the following criteria for every $2 < i < n$:

1. The subgraph $G_{i-1} \subseteq G$ induced by $v_0, v_1, \ldots, v_{i-1}$ is 2-connected, and the boundary of its outer face is a cycle C_{i-1} containing the edge (u, v);
2. The vertex v_i is in the exterior face of G_{i-1}, and its neighbors in G_{i-1} form an (at least 2-element) subinterval of the path $C_{i-1} - (u, v)$.

The canonical labeling of a planar graph G allows for the incremental placement of the vertices of G on a grid of size $O(n) \times O(n)$ so that when the edges are drawn as straight-line segments there are no crossings in the drawing. The two criteria that define a canonical labeling are crucial for the region creation step of our algorithm.

Kant generalized the definition for triconnected graphs. In this case, the vertices are partitioned into sets V_1 to V_K which can be either singleton vertices or chains of vertices [18].

2 Lower Bound of Six Sides

Here we show that at least six sides per polygon are needed in touching polygon representations of planar graphs. We begin by constructing a class of planar graphs that cannot be represented by four-sided polygons and then extend the argument to show that there exists a class of planar graphs that cannot be represented by five-sided regions.

2.1 Four Sides Are Not Enough

Consider the fully triangulated graph G in Figure 2(a). G has three nodes on the outer face A, B and C, and contains a chain of nodes $1, ..., k$ which are all adjacent to A and B. Consecutive nodes in the chain, i and $i + 1$, are also adjacent. The remaining nodes of G are degree-3 nodes l_i and r_i inside the triangles $\Delta(A, i, i + 1)$ and $\Delta(B, i, i + 1)$.

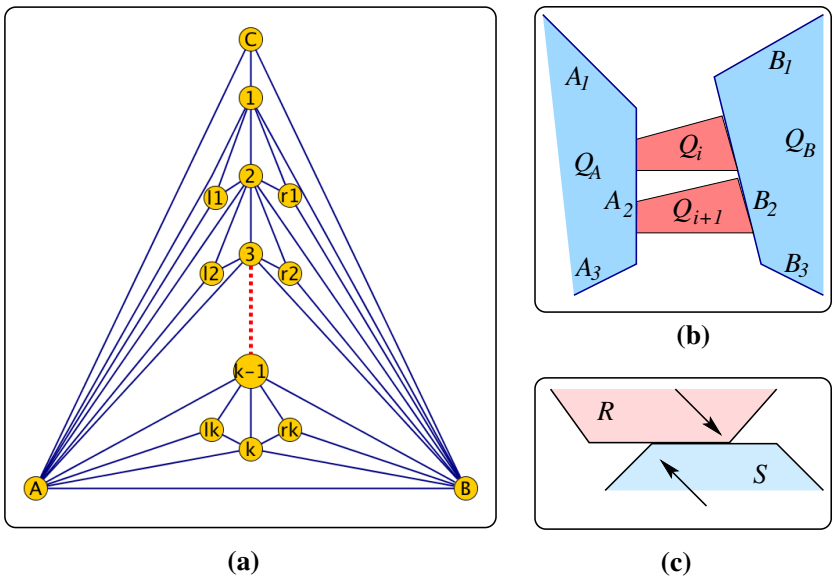

Fig. 2. (a) The graph that provides the counterexample. (b) A pair of subsequent fair quadrilaterals adjacent to the same sides of Q_A and Q_B. (c) Illustration for Lemma 2.

Theorem 1. *For k sufficiently large, there does not exist a touching polygon representation for G in which all regions have complexity 4 or less.*

Proof: Assume, for the sake of contradiction, that we are given a touching polygon drawing for G in which all regions have complexity 4 or less. Without loss of generality, we assume that the drawing has an embedding that corresponds to the one shown in Figure 2(a). Let Q_A and Q_B denote the quadrilaterals representing nodes A and B, and Q_i denotes the quadrilateral representing node i. Once again, without loss of generality, let Q_A lie in the left corner, Q_B in the right corner and Q_C at the top of the drawing.

We start with a couple of observations:

Observation 1: Since the three quadrilaterals Q_A, Q_B, Q_C are adjacent to the outer face, a complete side of each quadrilateral is adjacent to the outer face.

From this observation, we conclude that at most three sides of each of the outer quadrilaterals are inside of the drawing. We consider the three sides A_1, A_2, A_3 and B_1, B_2, B_3 of Q_A and Q_B, respectively, numbered from top to bottom; see Figure 2(b). The quadrilaterals of the chain are adjacent to the three sides in this order, such that if Q_i is adjacent to A_j (resp. B_j), then Q_{i+1} is adjacent to A_k (resp. B_k) with $k \geq j$. The adjacency of each Q_i defines two intervals, one on the polygonal chain A_1, A_2, A_3 and another one on B_1, B_2, B_3.

Observation 2: Consider the $c(= 4)$ corners of Q_A and Q_B, where the sides A_1 and A_2, A_2 and A_3, B_1 and B_2, B_2 and B_3 coincide. Clearly, at most 2 of the intervals that are defined by the adjacencies of the Q_i's are adjacent to each of the c corners. In total, this makes at most $2c = 8$ intervals, that are adjacent to any of the corners of Q_A or Q_B. Hence, at most 8 quadrilaterals of the chain $Q_1, ..., Q_k$ are adjacent to corners of Q_A and/or Q_B.

We now consider the quadrilaterals that do *not* define any of those intervals.

Let Q_i be a quadrilateral that is not adjacent to any of the corners of the polygonal chains A_1, A_2, A_3 and B_1, B_2, B_3. Two of its corners are adjacent to the same side A_k and to the same side $B_l, 1 \leq k, l \leq 3$ of Q_B. We call such a quadrilateral a *fair quadrilateral*.

Lemma 1. *If we choose k large enough, there exists a pair of fair quadrilaterals Q_i and Q_{i+1} that are adjacent to the same sides of Q_A and Q_B.*

Proof: We use a counting argument. We know that at most 8 quadrangles are not fair. Hence, for $k \geq 2 \cdot 2c + 2 = 18$, there must be a pair of subsequent fair quadrilaterals. The worst case happens for $k = 17$ if

$Q_2, Q_4, Q_6, \ldots Q_{16}$ are not fair. We can state even more precisely that there are at least $k-17$ pairs of subsequent fair quadrilaterals. Note that the pair (Q_i, Q_{i+1}) of fair quadrilaterals where Q_i is adjacent to the sides A_1 and B_1, but Q_{i+1} is not adjacent to A_1 and B_1 does not have the property claimed in the lemma. We call such a pair transition pair.

We can partition the set of fair quadrilaterals into at most 5 equivalence classes $C_1, ..., C_5$ that denote the sets of fair quadrilaterals, which are adjacent to the same sides of Q_A and Q_B. When we sweep through the chain of middle quadrilaterals, we simultaneously proceed through the equivalence classes. Hence there exist at most $t = 4$ transition pairs, namely pairs of subsequent fair quadrilaterals that are in different equivalence classes.

These equivalence classes denote the pairs of sides (A_i, B_j) that are used, beginning from the top with, say, (A_1, B_1), then (A_1, B_2), (A_2, B_2), (A_3, B_2) and finally (A_3, B_3). Note that this is not the only possible set of equivalence classes, but by planarity, it is not possible to have (A_2, B_3) and (A_3, B_1) simultaneously. Hence, there are at most 5 classes.

We repeat our counting argument from above and argue that for $k \geq 23$ there are at least 5 or more pairs of subsequent fair quadrilaterals, so at least one has the property claimed in the lemma. □

Before we continue with the proof of the theorem, we include the following Lemma, illustrated in Figure 2(c):

Lemma 2. *If there are two regions R, S touching in some nontrivial interval $I = (a, b)$ then at a, there is a corner of R or S. The same holds for corner b.*

Now, let (Q_i, Q_{i+1}) be a pair of fair same-sided quadrilaterals, touching sides A_p and B_q. Since Q_i and Q_{i+1} have to be adjacent, the two sides next to each other touch. We can use the above Lemma 2 to show that each interval that is shared by two polygons ends at two of the corners of the two polygons. Since there exist the polygonal regions representing r_i and l_i, it is clear that the interval where Q_i and Q_{i+1} touch is disjoint from the regions Q_A and Q_B. Hence the corners derived from Lemma 2 are not the corners of Q_i or Q_{i+1} that are incident to sides A_p and B_q. This is a contradiction, since then both Q_i and Q_{i+1} must have at least 5 corners, or one of them has even 6 corners. □

2.2 Five Sides Are Not Enough

If we allow the regions to be pentagons, we have to sharpen the argument a little more.

Lemma 3. *If we choose k large enough, there exists a triple of fair pentagons P_i, P_{i+1}, P_{i+2} that is adjacent to the same sides of P_A and P_B.*

Proof: We prove this along the same lines as before. Now we have four sides with $c = 6$ inner corners of the pentagons P_A and P_B. As before, we can see that at most 12 pentagons of the inner chain are not fair. Since we aim now for triples and not just for pairs, we get a worst case where every third pentagon is not fair. Hence for $k \geq 3 \cdot 2c + 3$, we get at least $k - 38$ fair subsequent pentagons. Next, we estimate the number of transition triples. The number of equivalence classes of pentagons with sides solely on the same side of P_A and P_B is seven. As we deal with triples, this makes a bound of at most 14 transition triples, since we can differentiate transition points between the first two and the last two pentagons of the triple.

Hence, we have to grow k to 38 + 14 = 52 to ensure that a triple of fair same-sided pentagons exists. □

Theorem 2. *For k sufficiently large, there does not exist a touching polygon representation for G in which all regions have complexity five or less.*

Proof: We choose k to be at least 52. Now, let (P_i, P_{i+1}, P_{i+2}) be a triple of fair same-sided pentagons, touching sides A_p and B_q. Since P_i and P_{i+1} have to be adjacent, the two sides next to each other touch. We can use Lemma 2 that each interval that is shared by two polygons ends at two of the corners of the two polygons. Since there exist the polygonal regions representing r_i and l_i, it is clear that the interval where Q_i and Q_{i+1} touch is disjoint from the regions P_A and P_B. Hence the corners derived from Lemma 2 are not the corners of P_i or

P_{i+1} that are incident to sides A_p and B_q. This is a contradiction, since both P_i and P_{i+1} have at least 5 corners, or one of them has even 6 corners. In the case, that P_i and P_{i+1} have exactly 5 corners, we repeat the same argument for P_{i+1} and P_{i+2}. From the second application, we prove the existence of a second additional corner at P_{i+1} or that P_{i+2} has two additional corners at the side opposite to P_{i+1}. In both cases, we get a contradiction. There exists a region with at least 6 corners. □

Note that six-sided polygons are indeed sufficient to represent the graph in Figure 2(a). In particular, for subsequent fair polygons P_i and P_{i+1}, we can use three segments on the lower side of P_i, while the upper side of P_{i+1} consists of only one segment which completely overlaps the middle of the three segments from the lower side of P_i.

3 Touching Hexagons Representation

In this section, we present a linear time algorithm that takes as input a planar graph $G = (V, E)$ and which produces a representation of G in which all regions are convex hexagons, thus proving that planar graphs belong to the class of touching hexagons graphs.

3.1 Algorithm Overview

We assume that the input graph $G = (V, E)$ is a fully triangulated planar graph with $|V| = n$ vertices. If the graph is planar but not fully triangulated, we can augment it to a fully triangulated graph with the help of dummy vertices and edges, run the algorithm below and remove the polygons that correspond to dummy vertices. Traditionally, planar graphs are augmented to fully triangulated graphs by adding edges to each non-triangular face. Were we to take this approach, however, when we remove the dummy edges we have to perturb the resulting space partition to remove polygonal adjacencies. As this is difficult to do, we convert our input graph to a fully triangulated one by adding one additional vertex to each face and connecting it to all vertices in that face. The above approach works if the input graph is biconnected. Singly-connected graphs must first be augmented to biconnected graphs as follows. Consider any articulation vertex v, and let u and w be consecutive neighbors of v in separate biconnected components. Add new vertex z and edges (z, u) and (z, w). Iterating for every articulation point biconnects G and results in an embedding in which each face is bounded by a simple cycle.

The algorithm has two main phases. The first phase, computes the canonical labeling. In the second phase we create regions with slopes 0, 1, -1 out of an initial isosceles right-angle triangle, by processing vertices in the canonical order. Each time a new vertex is processed, a new region is carved out of one or more already existing regions. At the end of the second phase of the algorithm we have a right-angle isosceles triangle which has been partitioned into exactly $n = |V|$ convex regions, each with at most 6 sides. We will show that creating and maintaining the regions requires linear time in the size of the input graph. We illustrate the algorithm with an example; see Figure 3.

3.2 Region Creation

In this section we describe the n-step incremental process of inserting new regions in the order given by the canonical labeling, where $n = |V|$. The regions will be carved out of an initial triangle with coordinates $(0, 0), (-1, 1), (1, 1)$. The process begins by the creation of R_0, R_1, and R_2, which correspond to the first three vertices, v_0, v_1, v_2; see Figure 3(a). Note that the first three vertices in the canonical order form a triangular face in G and hence must be represented as mutually touching regions.

At step i of this process, where $2 < i < n$, region R_i will be carved out from the current set of regions. Define a region as "active" at step i if it corresponds to a vertex that has not yet been connected to all its neighbors. An invariant of the algorithm is that all active regions are non-trivially tangent to the top side of the initial triangle, which we refer to as the "active front."

New vertices are created in one of two ways, depending on the degree of the current node, v_i, in the graph induced by the first i vertices, G_i. By the property of the canonical ordering and the active regions invariant, v_i is connected to 2 or more consecutive vertices on the outer face of G_{i-1}:

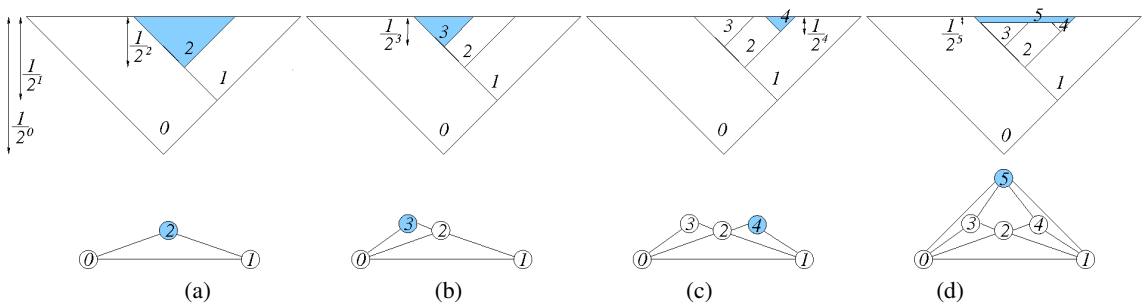

Fig. 3. Incremental construction of the touching hexagons representation of a graph. Shaded vertices on the bottom row and shaded regions on the top row are processed at this step. In general, the region defined at step i is carved at distance $1/2^i$ from the active front on the top. Note that the top row forms a horizontal line at all times.

1. If $d_{G_i}(v_i) > 2$ then R_i, the region corresponding to v_i, is a quadrilateral carved out of all but the leftmost and rightmost regions, by a horizontal line segment that is at distance $1/2^i$ from the active front; see Figure 3(d). Note that all but the leftmost and rightmost neighbors of v_i are removed from the set of active regions as their corresponding vertices have been connected to all their neighbors. Region R_i is added to the new set of active regions. Call this a "type 1 carving."
2. If $d_{G_i}(v_i) = 2$, let R_a and R_b be its neighbors on the frontier. Region R_i is then carved out as a triangle from either R_a or R_b

Lemma 4. *The regions produced by the above algorithm are convex and have at most 6 sides.*

Proof: First note that the above algorithm leads to the creation of at most fifteen different types of regions; see Figure 4. Each region has a horizontal top segment, a horizontal bottom segment (possibly of length 0), and sides with slopes -1 or 1. Moreover, each region can be characterized as either opening (the first two), static (the next six), or closing (the last 7), depending on the angles of the two sides connecting it to the top horizontal segment. Opening and static regions give rise to new regions via type 1 carvings (dashed arrows) and type 2 carvings (solid arrows). Closing regions only give rise to type 1 carvings.

We show that the regions produced as a result of type 1 and type 2 carvings from the initial triangle are convex polygons with at most 6 sides with slopes 0, 1, -1 by induction on the number of steps. Assume that the claim is true until right before step i; we will show that the claim is true after step i.

If $d_{G_i}(v_i) > 2$ then the new region R_i is created by a type 1 carving. Recall that R_i is created by the addition of the horizontal line segment at distance $1/2^i$ from the top of the triangle, cutting through all but the leftmost and rightmost neighbors of v_i. It remains to show that the resulting region R_i has exactly four sides and that the complexity of the all other regions is unchanged. By construction, R_i has a top and bottom horizontal segments and exactly one line segment on the left and one line segment on the right. The construction of R_i resulted in modifications in the regions representing all but the leftmost and rightmost neighbors of v_i in G_i, and there is at least one such neighbor. The changes in these regions are the same: each such region had its top carved off by the bottom horizontal side of the new region R_i. These changes do not affect the number of sides defining the regions. Regions corresponding to nodes that are not adjacent to v_i in G_i are unchanged.

Otherwise, if $d(v_i) = 2$ we must create a new region R_i between two adjacent regions R_a and R_b. By construction, the complexity of the new region R_i is 3, as we carve off a new triangle between regions R_a and R_b with a horizontal top side and apex at distance $1/2^i$ from the active front. As a result of this operation either the R_a or R_b was modified and all other regions remain unchanged. Specifically, the complexity of either R_a or R_b must increase by exactly one. Without loss of generality, let R_a be the region from which R_i will be carved; see Figure 5. It is easy to see that if R_a had complexity 6 then it must have been a "closing" region (one of the rightmost two in the last row on Fig. 4. Then the new region R_i would have been carved out of R_b which must have complexity 5 or less as it is impossible to have R_a and R_b both "closing" and adjacent. Therefore, at the end of step i the complexity of R_a has increased by one but is still no greater than 6. □

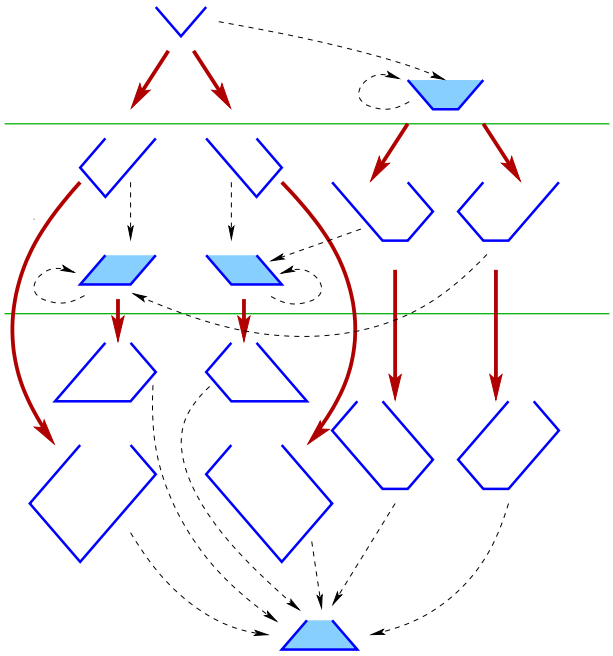

Fig. 4. There are a fifteen possible region shapes, falling into three categories: 2 opening, 6 static, and 7 closing. Solid arrows indicate type 2 (triangular) carving and dashed arrows indicate type 1 carving (a horizontal strip from the top of the current region). The four filled quadrilateral regions are the only types created due to type 1 carving.

3.3 Running Time

The above algorithm can be implemented in linear time. The linear time algorithm for computing a canonical labeling of a planar graph [5] requires a planar embedding as an input. Recall that planar embedding of a graph is simply a clockwise order of the neighbors of each vertex in the graph. Obtaining a planar embedding can be done in linear time using the algorithm by Hopcroft and Tarjan [15].

Creating and maintaining the regions in the second phase of our algorithm can also be done in linear time. We next prove this by showing that each region requires $O(1)$ time to create and requires $O(1)$ number of modifications.

Consider the creation of new regions. By the properties of canonical labeling, when we process the current vertex v_i, it is adjacent to at least two consecutive vertices on the outer face of G_{i-1}. By construction of our algorithm the vertices in the outer face of G_{i-1} correspond to active regions and so have a common horizontal tangent. If $d_{G_i}(v_i) = 2$, then a new region R_i is carved out of one of the neighboring regions R_a or R_b. Determining the coordinates of R_i takes constant time, given the coordinates of R_a and R_b and the fact that R_i will have height $1/2^i$ and will be tangent to the active frontier. If $d_{G_i}(v_i) > 2$, then all but the leftmost and rightmost neighbors of v_i have their corresponding regions carved, in order to create the new region R_i. In this case the coordinates of the R_i can also be determined in constant time given the coordinates of the leftmost and rightmost neighbors and the fact that R_i will have height $1/2^i$ and will be tangent to the active frontier. Note that the updates of the regions between the leftmost and rightmost are considered in the modification step.

Consider the modifications of existing regions. As can be seen from the hierarchy of regions on Figure 4, there are exactly 15 different kinds of regions and each region begins as a triangle and undergoes at most 4 modifications (e.g., from triangle, to quadrilateral, to pentagon, to hexagon, to quadrilateral). Moreover, once a region goes from one type to the next, it can never change back to the same type (i.e., all the arrows point downward). Finally note that the total number of region modifications is proportional to $|E|$ and since G is planar, $|E| = O(|V|)$ Thus, each region needs at most a constant number of modifications from the time it is created to the end of the algorithm.

The algorithm described in this section, yields the following theorem:

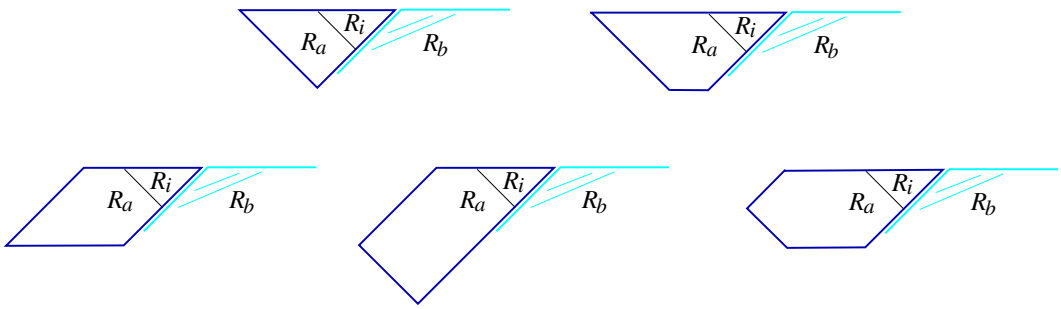

Fig. 5. Introducing region R_i between R_a and R_b, assuming R_i is carved out of R_a. All the possible cases are shown, assuming that R_a and R_b were convex, at most 6-sided regions with slopes 0, 1, -1. (There are five more symmetric cases when R_i is carved out of R_b.) Note that these five regions correspond to the non-filled regions from the region-creating hierarchy in Fig. 4 with two static regions in the first row and the three closing regions in the second row.

Theorem 3. *A planar graph can be converted into a set of touching convex polygons with complexity at most six, in linear time in the number of vertices of the graph.*

As defined, the above algorithm requires exponential area, if polygonal endpoints are to be placed at integer grid points. We show in the Appendix how to compact the initial exponential area drawings. However, the compaction approach is not guaranteed to always find a small area drawing. Therefore, we next show with a different algorithm that, in fact, $O(n) \times O(n)$ area suffices.

4 Hexagonal Representation of Planar Graphs using $O(n) \times O(n)$ Area

One drawback to the algorithms described in Sections 3 is it is not easy to obtain a good bound on the drawing area. Using a different approach, we can show that any general n-vertex planar graph can be represented by touching convex hexagons, drawn on the $O(n) \times O(n)$ grid. This approach is based on Kant's algorithm for hexagonal grid drawing of 3-connected, 3-planar graphs [17]. In Kant's algorithm the drawing is obtained by looking at the dual graph, and processing its vertices in the canonical order. In the final drawing, however, there are two non-convex faces, separated by an edge which is not drawn as a straight-line segment. These problems can be addressed by pre-processing the graph, by adding several extra vertices. When the dual of this augmented graph is embedded, the faces corresponding to the extra vertices can be removed to yield the desired grid drawing on area $O(n) \times O(n)$.

Let $H = (V, E)$ be a 3-connected, 3-planar graph. Note that the dual $D(H)$ is fully triangulated, as each face in the dual corresponds to exactly one vertex in H. So, for f faces in H, we have f vertices in $D(H)$. We first compute a canonical ordering on the vertices of $D(H)$ as defined by de Fraysseix et al. [7]. Let $v_1, ..., v_f$ be the vertices in $D(H)$ in this canonical order.

Kant's algorithm now constructs a drawing for H such that all edges but one have slopes $0°$, $60°$, or $-60°$, with the one edge with bends lying on the outer face. The typical structure of those drawings is shown in Figure 6(a).

The algorithm incrementally constructs the drawing by adding the faces of H in reverse order of the canonical order of the corresponding vertices in $D(H)$. We let w_i be the vertices of H. Let face F_i correspond to vertex v_i in $D(H)$. The algorithm starts with a triangular region for the face F_f that corresponds to vertex v_f. The vertex w_x which is adjacent to F_f, F_1 and F_2 is placed at the bottom. Let w_y and w_z be the neighbors of w_x in F_f. These three vertices form the corners of the first face F_f. (w_x, w_z) and (w_x, w_y) are drawn upward with equal lengths and slopes -1 and 1, respectively. All the edges on the path between w_y and w_z along F_f are drawn horizontally between the two vertices. From this first triangle, all other faces are added in reverse canonical order to the upper boundary of the drawing region. If a face is completed by only one vertex w_i, this vertex is placed appropriately above the upper boundary such that it can be connected by two edges with slopes -1 and 1, respectively. If the face is completed by a path, then the two end segments of the path have slopes -1 and 1, while the other edges

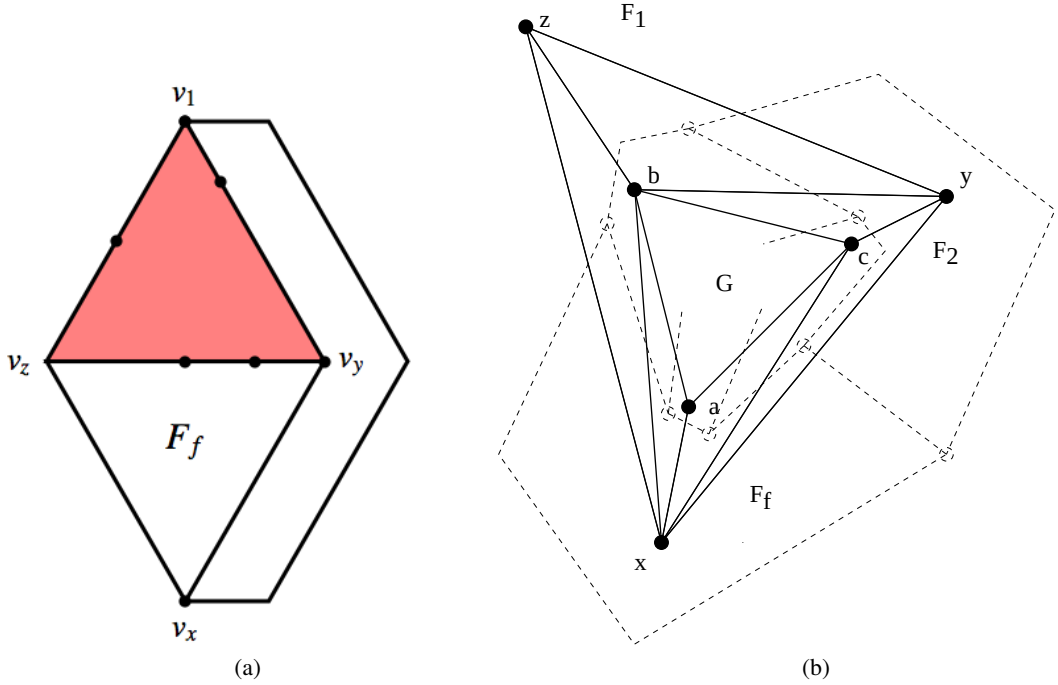

Fig. 6. (a) Polygonal structure obtain from Kant's algorithm. (b) Graph G augmented by vertices z, y and x together with its dual which serves as input graph for Kant's algorithm.

are horizontal. The construction ends when w_1 is inserted, corresponding to the outer face F_1. Note that there is an edge between w_1 and w_x, which is drawn using some bends. This edge is adjacent to the faces F_1 (the outer face) and F_2.

From this construction, we can observe that the angles at faces $F_f, ..., F_3$ have size $\leq \pi$ as the first two edges do not enter the vertex from above, and the last edge leaves the vertex upwards. Hence, we have the following result.

Lemma 5. *The faces $F_f, ..., F_3$ are convex, and as the slopes of the edges are -1,0 or 1, they are drawn with at most 6 sides.*

This property is exactly what we are aiming for, as the vertices of our input graph G should be represented by convex regions of at most 6 sides. Unfortunately, Kant's algorithm creates two non-convex faces F_1 and F_2 separated by an edge which is not drawn as a line segment. Furthermore, the face F_f is drawn as large as all the remaining faces $F_3, ...F_{f-1}$ together.

Kant also gave an area estimate for the result of his algorithm. A corollary of Kant's algorithm is the following.

Corollary 1. *For a given 3-connected, 3-planar graph H of n vertices, $H - w_x$ can be drawn within an area of $n/2 - 1 \times n/2 - 1$.*

4.1 From Hexagonal Grid Drawing to Touching Hexagons

To apply Kant's result to the problem of constructing touching hexagons representation, we enlarge the embedded input graph G so that the dual of the resulting graph G' can be drawn using Kant's algorithm in such a way that the original vertices of G correspond to the faces $F_3, ..., F_{f-1}$.

We have to add 3 vertices which will correspond to the faces F_1, F_2 and F_f in Kant's algorithm. Since G is fully triangulated, let a, b and c be the vertices at the outer face of G in clockwise order. We add the vertices x, y and z in the outer face and connect to$G$ appropriately. We want z to correspond to the outer face F_1, y correspond to F_2 and x to F_f. First, we add x and connect it to a, b and c such that b and c are still in the outer face. Then

we add y and connect it to x, b and c such that b is still in the outer face. Finally, we add z and connect it to x, b and c such that z, y and x are now in outer face, as shown in the Figure 6(b).

Since the vertices x, y, z are on the outer face, we can choose which one is first, second and last in the canonical order. We can then apply Kant's algorithm with the canonical order $v_1 = z, v_2 = y$ and $v_f = x$. After constructing the final drawing, we remove the regions corresponding to vertices z, y and x, leaving us with a hexagonal representation of G. Since Kant's algorithm runs in linear time, and our emendations can be done in constant time, we can summarize:

Theorem 4. *For a fully triangulated planar graph G on n vertices, we can construct a contact graph of convex hexagons in time O(n). The sides of the hexagons have slope 1, 0, or -1.*

Given any planar graph G, if it is not biconnected, we can make it biconnected using a procedure attributed to Read [27], adding a vertex and two edges at each articulation point. Once biconnected, we can fully triangulate the graph by adding a vertex inside each non-triangular face and connecting that vertex to each vertex on the face. We can then apply Theorem 4, to get a hexagonal representation of the extended graph. Finally, removing the added vertices and their edges, we obtain a hexagonal representation of G. This gives us:

Theorem 5. *For any planar graph G on n vertices, we can construct a contact graph of convex hexagons in time O(n). The sides of the hexagons have slope 1, 0, or -1.*

4.2 Area estimation

For a triangulated input graph $G = (V, E)$, we have n vertices and, by Euler's formula, $2n - 4$ faces. Since we enhanced our graph to $n + 3$ vertices, we have $f = 2n + 2$ faces. Those faces are the vertices in the dual $D(G)$ which is the input to Kant's algorithm. His area estimation gives an area of $n/2 - 1 \times n/2 - 1$ for $f = n$ vertices when we coalesce the faces F_1, F_2 and F_f into a single outer face by removing the corresponding vertices and edges. Thus, we get an area bound of $n \times n$ using exactly the same argument as he did.

Theorem 6. *For a fully triangulated planar graph G of n vertices, we can achieve a contact representation of convex hexagons with area $n \times n$.*

5 Conclusion and Future Work

Thomassen [29] had shown that not all planar graphs can be represented by touching pentagons, where the external boundary of the figure is also a pentagon and there are no holes between pentagons. Our results in this paper are more general, as we do not insist on the external boundary being a pentagon or on there being no holes between pentagons. Finally, it is possible to derive algorithms for convex hexagonal representations for general planar graphs from several earlier papers, e.g., de Fraysseix *et al.* [7], Thomassen [29], and Kant [17]. However, the work of Thomassen, Kant, and de Fraysseix *et al.* does not immediately lead to algorithmic solutions to the problem of touching polygons graph representation with convex low-complexity polygons. To the best of our knowledge, this problem has never been formally considered or described.

In this paper we presented several results about touching n-sided graphs. We showed that for general planar graph six sides are necessary. Then we presented an algorithm for representing general planar graphs with convex hexagons. Finally, we discussed a different algorithm for general planar graphs which also yields an $O(n) \times O(n)$ drawing area.

Several interesting related problems are open. What is the complexity of the deciding whether a given planar graph can be represented by touching triangles, quadrilaterals, or pentagons? In the context of rectilinear catrograms the vertex-weighted problem has been carefully studied. However, the same problem without the rectilinear constraint has received less attention. Finally, it would be interesting to characterize the subclasses of planar graphs that allow for touching triangles, touching quadrilaterals, and touching pentagons representations.

6 Acknowledgments

We would like to thank Therese Biedl for pointing out the very relevant work by Kant and Thomassen.

References

1. G. D. Battista, W. Lenhart, and G. Liotta. Proximity drawability: A survey. In *Proc. Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 328–39. Springer-Verlag, 1994.
2. T. Biedl, A. Bretscher, and H. Meijer. Rectangle of influence drawings of graphs without filled 3-cycles. In *Proc. 7th Int'l. Symp. on Graph Drawing '99*, pages 359–368, 1999.
3. M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In *Proc. Joint Eurographics/IEEE TVCG Symp. Visualization, VisSym*, pages 33–42, 2000.
4. A. L. Buchsbaum, E. R. Gansner, C. M. Procopiuc, and S. Venkatasubramanian. Rectangular layouts and contact graphs. *ACM Transactions on Algorithms*, 4(1), 2008.
5. M. Chrobak and T. Payne. A linear-time algorithm for drawing planar graphs. *Inform. Process. Lett.*, 54:241–246, 1995.
6. M. de Berg, E. Mumford, and B. Speckmann. On rectilinear duals for vertex-weighted plane graphs. *Discrete Mathematics*, 309(7):1794–1812, 2009.
7. H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3:233–246, 1994.
8. H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fary embeddings of planar graphs. In *Procs. 20th Symposium on Theory of Computing (STOC)*, pages 426–433, 1988.
9. H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.
10. K. R. Gabriel and R. R. Sokal. A new statistical approach to geographical analysis. *Systematic Zoology*, 18:54–64, 1969.
11. X. He. On finding the rectangular duals of planar triangular graphs. *SIAM Journal of Computing*, 22(6):1218–1226, 1993.
12. X. He. On floor-plan of plane graphs. *SIAM Journal of Computing*, 28(6):2150–2167, 1999.
13. P. Hliněný. Classes and recognition of curve contact graphs. *Journal of Comb. Theory (B)*, 74(1):87–103, 1998.
14. P. Hliněný and J. Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Mathematics*, 229(1-3):101–24, 2001.
15. J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.
16. J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80:1502–17, 1992.
17. G. Kant. Hexagonal grid drawings. In *18th Workshop on Graph-Theoretic Concepts in Computer Science*, pages 263–276, 1992.
18. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996. (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).
19. G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theoretical Computer Science*, 172:175–93, 1997.
20. P. Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig. Math.-Phys. Klasse*, 88:141–164, 1936.
21. K. Koźmiński and W. Kinnen. Rectangular dualization and rectangular dissections. *IEEE Transactions on Circuits and Systems*, 35(11):1401–16, 1988.
22. Y.-T. Lai and S. M. Leinwand. Algorithms for floorplan design via rectangular dualization. *IEEE Transactions on Computer-Aided Design*, 7:1278–89, 1988.
23. Y.-T. Lai and S. M. Leinwand. A theory of rectangular dual graphs. *Algorithmica*, 5:467–83, 1990.
24. C.-C. Liao, H.-I. Lu, and H.-C. Yen. Compact floor-planning via orderly spanning trees. *Journal of Algorithms*, 48:441–451, 2003.
25. G. Liotta, A. Lubiw, H. Meijer, and S. H. Whitesides. The rectangle of influence drawability problem. *Computational Geometry: Theory and Applications*, 10:1–22, 1998.
26. M. Rahman, T. Nishizeki, and S. Ghosh. Rectangular drawings of planar graphs. *Journal of Algorithms*, 50(1):62–78, 2004.
27. R. C. Read. A new method for drawing a graph given the cyclic order of the edges at each vertex. *Congressus Numerantium*, 56:31–44, 1987.
28. P. Steadman. Graph-theoretic representation of architectural arrangement. In L. March, editor, *The Architecture of Form*, pages 94–115. Cambridge University Press, 1976.
29. C. Thomassen. Plane representations of graphs. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 43–69, 1982.
30. C. Thomassen. Interval representations of planar graphs. *Journal of Comb. Theory (B)*, 40:9–20, 1988.
31. G. K. Yeap and M. Sarrafzadeh. Sliceable floorplanning by graph dualization. *SIAM Journal on Discrete Mathematics*, 8(2):258–80, 1995.

A Compact Grid Drawings

Figure A shows several examples of planar graphs and the corresponding output from our algorithm after a compaction step which minimizes the required integer grid.

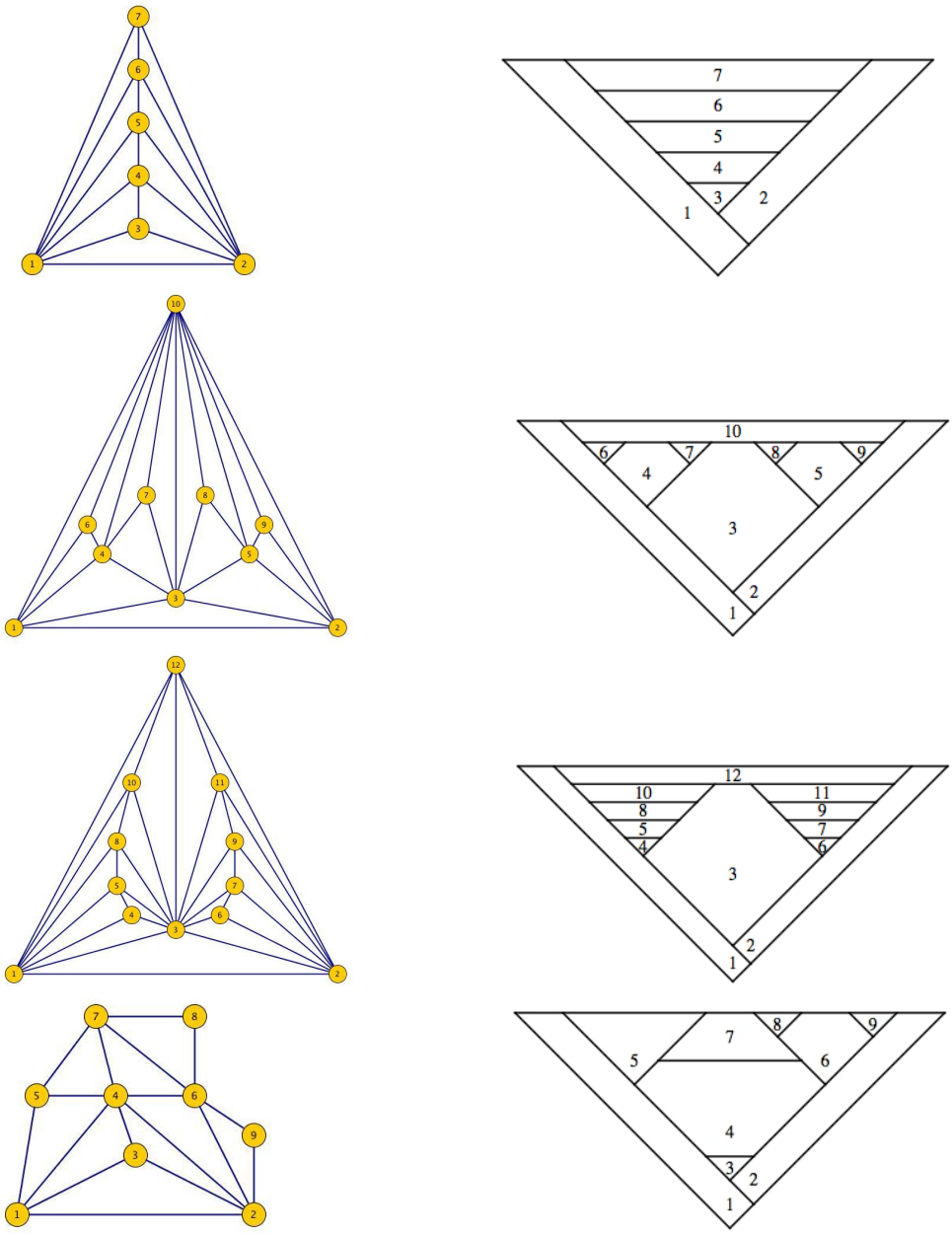

Fig. 7. Examples illustrating input graphs and the corresponding plane partitions.

The algorithm given in Section 3.2 provides a touching hexagons representation of any planar graph. The incremental process carves out polygons within an ever smaller band of active front, therefore in practice the drawing is highly skewed. In this section we describe an algorithm to get a drawing on the grid.

It is easy to see that when looking at the vertices and edges created in the algorithm for touching hexagons, if the horizontal edges are ignored, then the resulting graph is a "binary" tree, in the sense that each node has a degree of no more than 2. To achieve a drawing on the grid, we employ a divide and conquer algorithm. We need each edge to be of either 45 or -45 degree relative to the x axes, and that we have to constrain vertices linked by the omitted horizontal edges to the same y-coordinates.

Let G_{T6G} be the graph derived from the touching hexagons algorithm. Define a cap as a maximal connected component of vertices and horizontal edges of G_{T6G}. After removing the edges in the caps from G_{T6G}, the remaining graph G_T is a binary tree.

Define the frontal vertex set F of a tree as the set of leaf vertices, such that if a vertex in this set is also in a cap, then all the vertices in the same cap must also be leaves. This means that if a vertex that belongs to a cap is in the front, every vertex in this cap is also in the front. We initialize the current tree to be $G_c = G_T$, and the horizontal shift of every vertex to 0.

- 1. Let F be the frontal vertex set of G_c. If F is empty, exit.
- 2. For each vertex v in the front F,
  - if v is a leaf in G_T, place v on at $x = 0$ and $y = 0$.
  - if v has one subtree in G_T, say to the right, extend a line of 45 degree from the root of this right subtree by 1 unit down and left to get the position of v.
  - if v has two subtrees in G_T, shift the right subtree horizontally so that the two subtrees have a separation of either distance 1 or 2, and the line of -45 degree from the root of the left subtree and the line of 45 degree from the root of the right subtree meet at a grid point. Record this position of v, and the shift at the root of the right tree.
- 3. For each cap C in the front, set h to be the maximum of the absolute values of y coordinates of vertices in C. For every vertex $v \in C$,
  - if the $y(v) > -h$, set $y(v) = -h$ if v has no subtrees in G_T. Otherwise, by construction, vertex v must have only one subtree, say to the right. Extend the 45 degree line from the root of the subtree till it intersects with line $y = -h$, record the coordinates of the intersection point as the coordinates for v.
- 4. Delete F and its connecting edges from G_C, renaming the resulting tree G_C. Go to Step 1.

At the end of this algorithm, each vertex of G_T has $x-, y-$ coordinates and a horizontal shift. A traversal from the root is carried out to propagate the shifts and to get the final position of every vertex. This gives a drawing of the G_{T6G} on a grid. Figure A shows several graphs, and their corresponding touching hexagons representations on a grid.