

Finding a square dual of a graph

Thomas Picchetti

Summer 2011

Master 1 research internship, supervised by Stefan Felsner (TU Berlin)

Contents

1	Introduction	1
2	Rectangular Duals	1
2.1	Definitions	1
2.2	Restrictions on the graphs	2
2.3	Rectangular Duals and Transversal Structures	4
3	The set of Transversal Structures	7
3.1	Transversal structures and α_4 -orientations	7
3.2	The lattice of α_4 -orientations	7
3.3	Flipping in Transversal Structures	8
3.4	Minimal transversal structure	10
4	Finding the good one	10
4.1	Writing equations	10
4.2	An algorithm	12
4.3	Resolution	12
5	Conclusion	14
5.1	My contribution	14
5.2	Before and after	14

1 Introduction

In many situations, it is useful to partition a plane area into a set of polygonal regions, respecting adjacency constraints between these regions. This partition is called a cartogram. Cartograms can be used as a schematical way to visualize data, or to design the layout of an electronic chip, of a building, or other things. A cartogram C is said to represent a plane graph G if the regions of C are in bijection with the vertices of G , and two regions are adjacent if and only if the corresponding vertices share an edge. Many combinatorial problems have the following form : given a graph with property x , is it possible to represent it using a cartogram with property y ?

Constraints on the cartograms can be in terms of the number of sides of the polygons, their convexity, their area (one might impose that each region have a determined area)...

The main topic of this internship were rectangular dissections : these cartograms are formed by an outer rectangle, which is partitioned into many smaller ones, as in figure 1. If a graph is represented by this dissection, the dissection is called a rectangular dual of the graph. It has long been known which triangulations have rectangular duals, and how to compute one in linear time (see [1] for references). A more difficult problem is to know which triangulations have a *square* dual, or *squaring*: a rectangular dual where all the inner rectangles are squares. Schramm proved that this class is the same as with rectangular duals, and that for any graph in this class, the square dual is unique. However, no algorithm is known to compute the square dual.

An approach is to start with a rectangular dual, and try to adjust the rectangles' sides to obtain a squaring that is combinatorially equivalent to the initial rectangular dissection. This is not possible with any rectangular dual, and the idea is to go from one rectangular dual to another according to a given algorithm, until we find a suitable one. In this internship I developed a program to study the latter algorithm, in order to better understand how it behaves, and make easier the task of proving or disproving its correctness.

Section 2 describes the graphs on which we will focus, and the link between rectangular duals and transversal structures, an additional structure given to the graph. Section 3 deals with the set of transversal structures, which has a lattice structure, and shows how we go from one transversal structure to another. Section 4 is about trying to make a square dual out of a rectangular dual, and the algorithm that looks for a suitable rectangular dual.

2 Rectangular Duals

2.1 Definitions

A graph is *planar* if it can be embedded in the plane, without intersecting edges.

A *map* is a particular embedding of a given planar graph. A graph can have many non-topologically equivalent embeddings.

A map is also called *planar map* or *plane graph*. In addition to vertices and edges, it has faces, and the edges of a vertex v are ordered around v . A *triangular map* is a planar map where all faces except the outer one are triangles. It is a *triangulation* if moreover it has no loop of multiple edges. A graph is *n -connected* if, to disconnect it by removing vertices, you need to remove at least n vertices (e.g. a tree is 1-connected, but not 2-connected). A triangulation is 4-connected if you cannot disconnect it by removing 3 vertices. It can be shown that this is equivalent to the absence of *separating triangles* (triangles inside which lies a part of the graph).

2.2 Restrictions on the graphs

Let R be a rectangular dissection. It represents a planar map M whose vertex set is the set of inner rectangles in the dissection, and two of them are linked by an edge when they touch in the dissection (see figure 1). If we consider the dissection itself as a map, its faces correspond to M 's

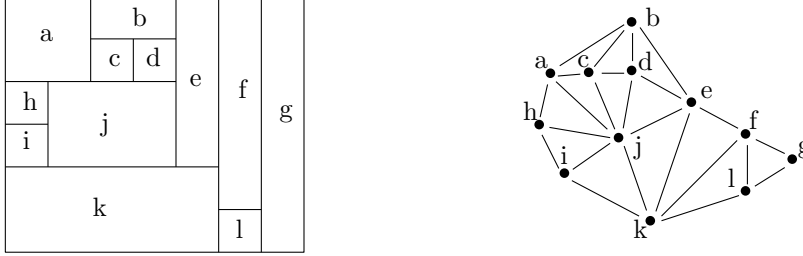


Figure 1: A rectangular dissection and the corresponding planar map

vertices, its edges to M 's edges, and its vertices to M 's faces. This is why we say R is a rectangular *dual* of M .

From now on, we will wonder if a graph has a rectangular dual only if it is a triangulation. Note that, in general, a graph with faces other than triangles and quadrangles can not have a rectangular dual (for no point in a rectangular dissection is incident to more than 4 rectangles).

If the dissection contains the pattern drawn in figure 2 (corresponding to a face of degree 4 in the

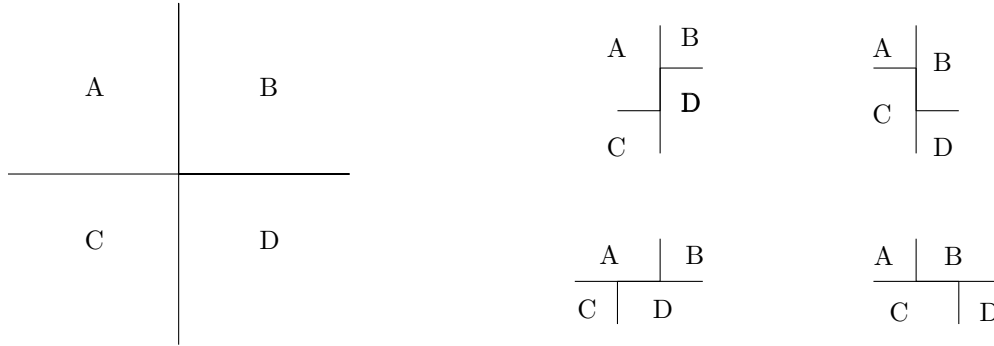


Figure 2: A degenerated pattern and the four possible interpretations

represented graph), we consider it as a degenerated occurrence of one of 4 patterns represented aside. Depending on which one it is, the dissection can represent several different triangulations.

We will apply another operation on our rectangular dissection R . Consider R' , where we have added an outer “frame” consisting of 4 rectangles (figure 3). In terms of the represented graph, this corresponds to adding an outer quadrangle formed by 4 vertices N, S, E and W. From now on, we will only consider graphs with an outer quadrangle N-E-S-W and in the dissections, we will omit the four outer rectangles.

Also, a triangulation having a rectangular dual must be 4-connected. Indeed, there is no way to

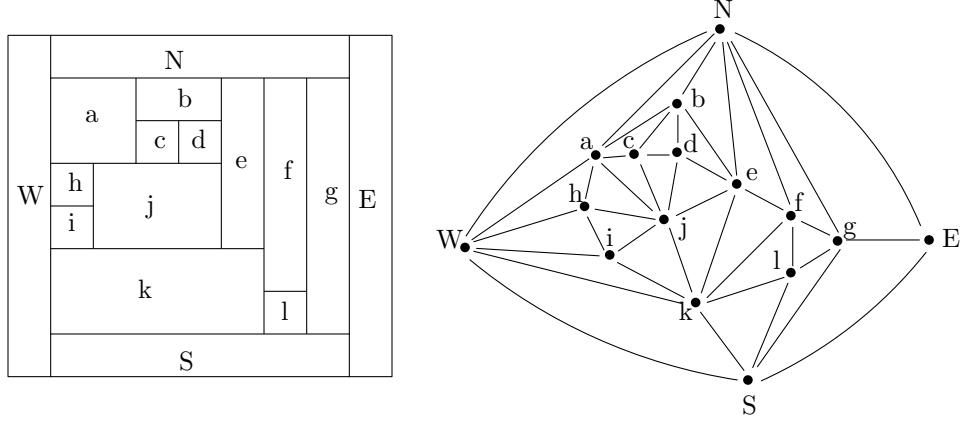


Figure 3: A frame is added to the dissection, and an outer quadrangle is added to the map

represent a separating triangle in a rectangular dissection: if 3 rectangles are pairwise adjacent, there is no way to place rectangles (even degenerated, zero-sized rectangles) inside the region they enclose.

From now on, we will therefore focus on 4-connected triangulations of an outer quadrangle N-E-S-W. A graph having such an embedding happens to have only this one planar embedding. Here is an algorithm to determine the embedding (consisting of, for each vertex v , the order of all v 's neighbours clockwise around v), starting from the raw graph (the adjacency lists):

For every inner vertex v , once we know two consecutive neighbours of v in clockwise order, we can determine the order of all of v 's neighbours. Suppose b comes right after a around v (see figure 4).

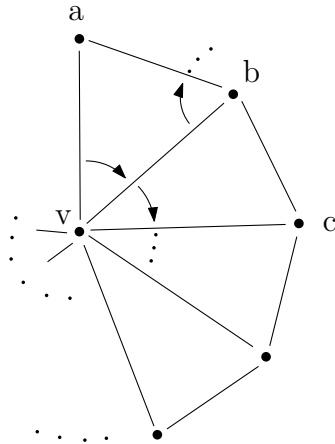


Figure 4: How to embed the neighbourhood of a vertex, starting from two consecutive neighbours

Then, b and v have only two common neighbours, a and some vertex c (if they have more than two common neighbours, this implies the presence of a separating triangle). This means that c comes right after b around v .

By iterating this step, we can go around v and order all its neighbours. This also allows to do the

same for all neighbours of v : for example, a comes right after v in the clockwise-ordered list of b 's neighbours. By this propagation mechanism, we can order the neighbours of all vertices of the graph.

This propagation is initiated at the four outer vertices : we know that E comes right after W around N, and similar statements about the three others.

This is the algorithm I use in the program, to generate the embedding of a graph given by its adjacency lists.

2.3 Rectangular Duals and Transversal Structures

A *transversal structure* of a triangulation of a quadrangle is an orientation of its inner edges, and their partitioning into blue edges and red edges, respecting the following local conditions (see figure 5):

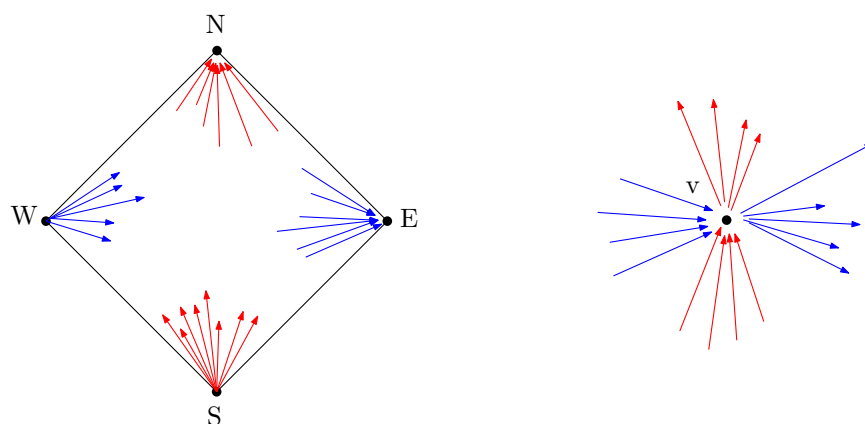


Figure 5: The two local conditions defining a transversal structure

- W has only outgoing blue edges, N has only incoming red edges, E has only incoming blue edges and S has only outgoing red edges.
- Around every inner vertex, the edges form 4 contiguous blocks of, in clockwise order: incoming blue edges, outgoing red edges, outgoing blue edges, and incoming red edges.

Transversal structures are the most important object in this work. They were introduced by Kant and He in [3], under the name “regular edge labeling”, and very deeply studied in Eric Fusy’s PhD Thesis [2]. A transversal structure fully describes a rectangular dual of the underlying triangulation, as explicated below.

Take a rectangular dissection R , with the represented triangulation T , and colour and orient the edges of T in the following manner (figure 6):

- when rectangle a lies directly above rectangle b in the dissection, colour the edge ab in red and orient it towards a
- when rectangle c lies directly to the right of rectangle d , colour the corresponding edge in blue and orient it towards c

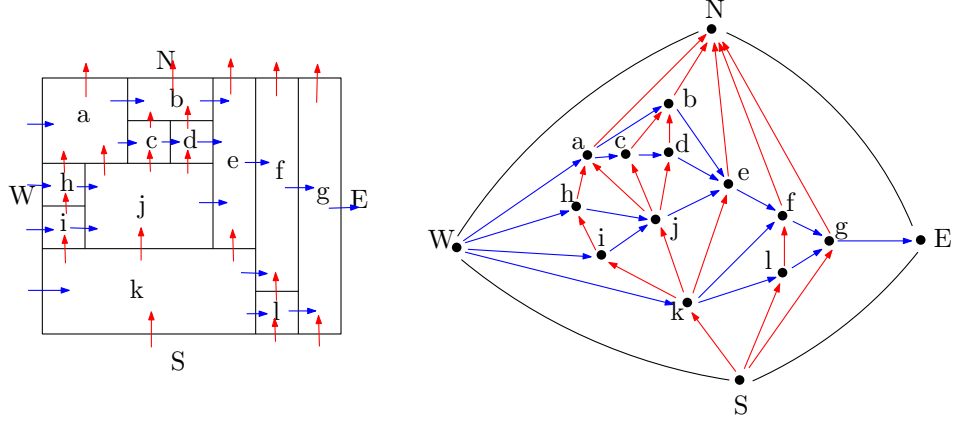


Figure 6: A rectangular dual corresponds to a transversal structure

This puts a transversal structure on T . Reciprocally, if this transversal structure is given, it allows to build the original rectangular dual of T (in linear time, moreover). To do this, consider the following: if we look at T with only its blue edges, we see an acyclic directed graph with only one source W and only one sink E . This is called a *bipolar orientation*. If we look at the red edges, we see another bipolar orientation going from S to N .

Now, to any bipolar orientation B , we can associate a dual bipolar orientation, whose underlying graph is the dual of that of B , and whose edges cross B 's edges from left to right (see figure 7).

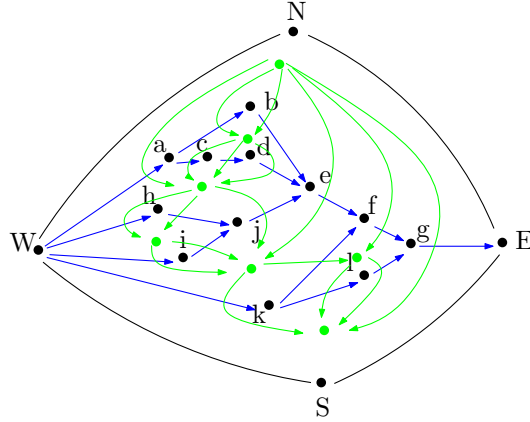


Figure 7: The blue bipolar orientation and its dual

If we look at the dual of the blue bipolar orientation, its vertices correspond to horizontal lines in the rectangular dissection, and an edge $u \rightarrow v$ means that the lines corresponding to u and v contain respectively the top point and the bottom point of some vertical segment (figure 8). If we see the transversal structure and want to reconstruct the rectangular dissection, we know we have to associate a horizontal line to each of u and v , ensuring that v is lower than u . We can do this by giving each horizontal line u a y coordinate equal to the length of the longest path from u to the dual source.

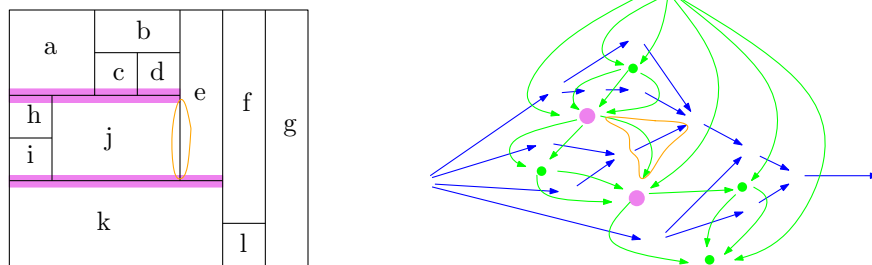


Figure 8: Correspondence between the dissection and the dual of the blue bipolar orientation

Similarly, by considering the dual of the red orientation, we can compute the set of vertical lines, and give an x coordinate to each of them.

From there, we can obtain all the useful coordinates to draw the rectangular dual associated to a given transversal structure. Kant and He showed that a triangulation of a quadrangle admits a transversal structure if and only if it is 4-connected, and give linear algorithms to compute a transversal structure of such a triangulation ([3]). Hence, given a 4-connected triangulation of a quadrangle, we can compute in linear time a rectangular dual.

Square Duals

The following question is interesting: what about square duals, i.e. rectangular duals where all the inner rectangles are squares ?

Schramm proved that a 4-connected triangulation of an outer quadrangle has a unique square dual. The proof uses topological tools. Lovasz ([4]) has given a formulation that uses a pair of blocking polyhedra, which was the topic of a speech I gave during my stay in Berlin. The proof defines two blocking polyhedra P and Q in an n -dimensional space where n is the number of inner vertices (P and Q are defined according to the paths in the triangulation). Then we define a point p between x , the point in P with minimum euclidian distance to zero, and y , its analog in Q . Finally, we show that p is related to a certain transversal structure, and that by associating to inner vertex i a square of side p_i and disposing these squares as indicated by the latter transversal structure, we obtain a square dual of our triangulation.

This proof does not seem to lead to an efficient algorithm to compute a square dual (the difficult step is to find the point with minimal distance to zero in a polyhedron), but the unicity tells us that, to produce a square dual, it will be important to lay hands on the unique suitable transversal structure, the one corresponding to the unique square dual.

Before looking for the right one, let us study the structure of the set of all transversal structures of a given triangulation.

3 The set of Transversal Structures

3.1 Transversal structures and α_4 -orientations

To study the set of transversal structures of a 4-connected triangulation T , it is useful to consider a derived map Q , as follows : the vertex set of Q , $V(Q)$ is the union of $V(T)$ and $F(T)$, the set of faces of T . $v \in V(T)$ and $f \in F(T)$ are joined by an edge in Q if they are adjacent in T . This map is called the *angular map* of T , because an edge in Q corresponds to an angle in T (see figure 9). A direct observation is that Q is a *quadrangulation*, that is, all its faces are quadrangles. This

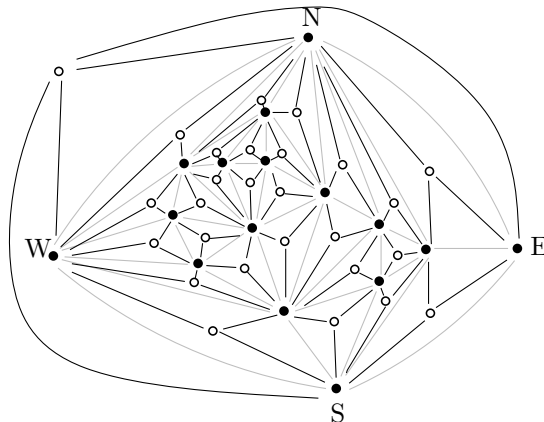


Figure 9: The angular map of the triangulation.

implies that Q is bipartite, and the two components are clearly $V(T)$ (from now on we call these *black* vertices) and $F(T)$ (which we call *white* vertices).

Now, suppose T is given a transversal structure, and orient the edges of Q as follows: given an edge e , it corresponds to an angle in T . If this angle lies between a red edge and a blue edge, orient e towards its white vertex. If it lies between two edges of the same colour, orient e towards its black vertex (figure 10). From this orientation, it is easy to recover the partition of T 's edges into blue ones and red ones (by starting on the outer vertices and propagating information about colour through the graph). Now, given this colouring, it is also easy to orient them so as to get the transversal structure back (using the same information propagation approach).

In the orientation of Q , every black vertex will have outdegree 4, and every white vertex will have outdegree 1 (with exceptions near the outer face: the four outer vertices N,S,E,W have outdegree 0, and since the four outer edges are not coloured, the white vertices adjacent to them have meaningless outdegrees). An orientation of Q having these outdegrees for these vertices is called an α_4 -orientation. It is shown in Fusy's PhD Thesis ([2]) that every α_4 -orientation is associated to a transversal structure, so the set of transversal structures of T can be identified with the set of α_4 -orientations of Q .

3.2 The lattice of α_4 -orientations

More generally, an α -orientation of a graph is an orientation of its edges giving each vertex a prescribed outdegree. For a given outdegree attribution $\alpha : V \rightarrow \mathbb{N}$, there are generally many

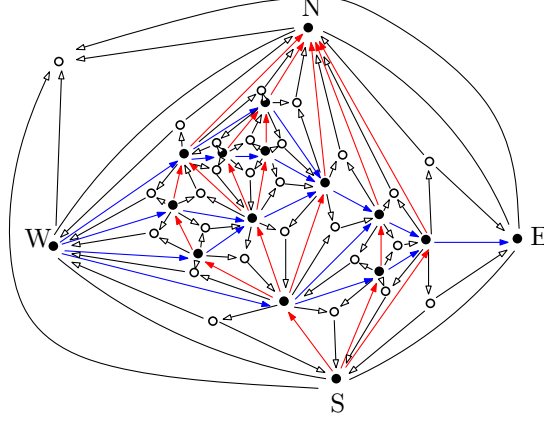


Figure 10: The orientation of the angular map associated to a given transversal structure.

α -orientations. Many structures of planar graphs can be defined in terms of α -orientations, which makes them a central object in combinatorics.

The set of *alpha*-orientations of a graph has a distributive lattice structure. Here is a quotation from Fusy's thesis, for the definition of this structure ([2]):

“

A *lattice* is a partially ordered set $(E, \leq)$ such that, for each pair (x, y) of elements of E , there exists a unique element $x \wedge y$ and a unique element $x \vee y$ satisfying the conditions:

- $x \wedge y \leq x$, $x \wedge y \leq y$, and $z \in E, z \leq x$ and $z \leq y$ implies $z \leq x \wedge y$,
- $x \vee y \geq x$, $x \vee y \geq y$, and $z \in E, z \geq x$ and $z \geq y$ implies $z \geq x \vee y$.

In other words, each pair admits a unique common lower element dominating all other common lower elements, and the same holds with common upper elements.

The lattice is said to be *distributive* if the operators $\wedge$ and $\vee$ are distributive with respect to each other.

We define a flip (respectively flop) as the operation of moving from an element of the lattice to a lower (respectively upper) covering element.

”

In the case of α -orientations, the flip (resp. flop) operation consists in reversing an *essential* clockwise (resp. counterclockwise) oriented cycle. An oriented cycle C is *essential*, as opposed to *chordal* if it doesn't enclose a *chordal path*, an oriented path whose extremities are both on C (the reversal of a chordal cycle can be decomposed into the reversals of several essential cycles enclosed in it).

3.3 Flipping in Transversal Structures

In the particular case of α_4 -orientations, in the angular maps we are considering, essential cycles can occur in two forms :

- either they are cycles of length 4, i.e. consist of one face of Q , enclosing one edge of T ,

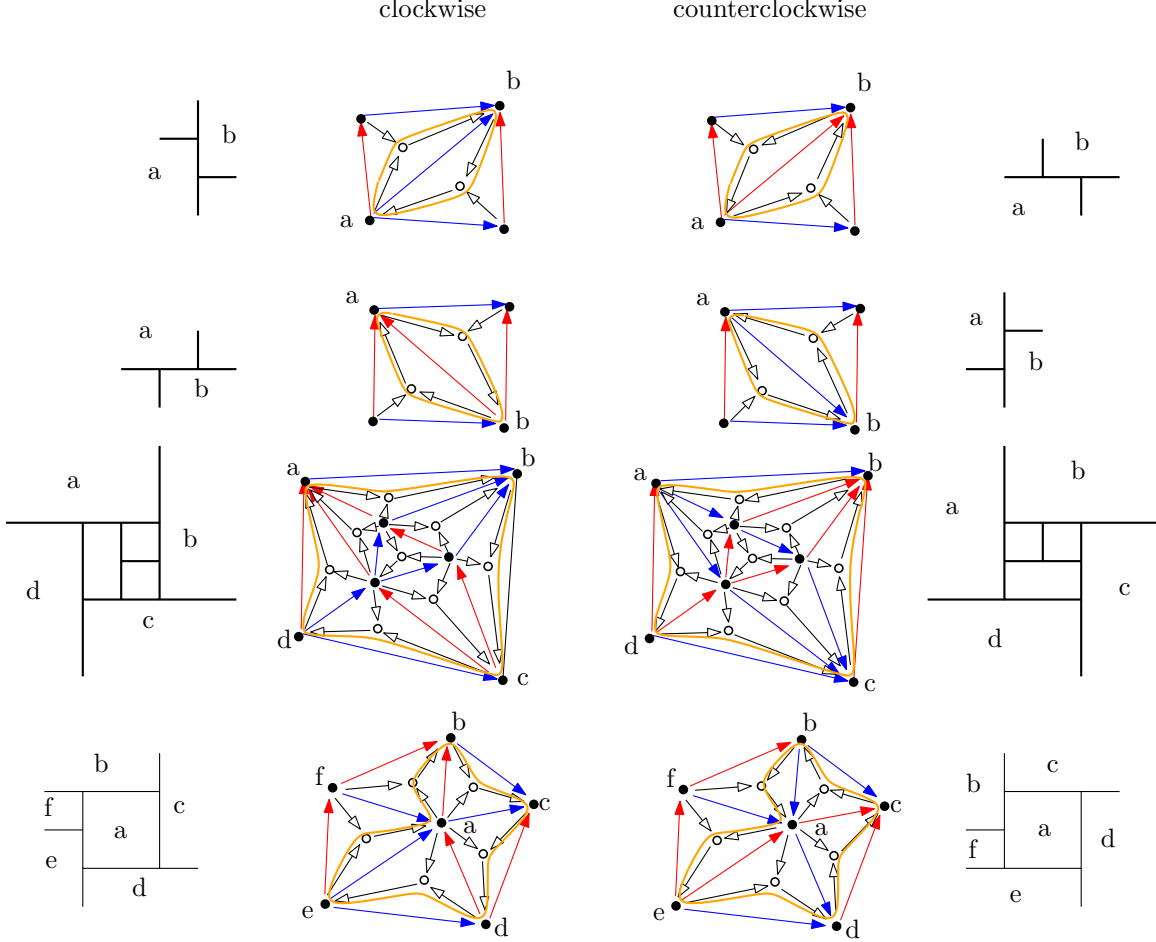


Figure 11: Examples of cycle reversal in the angular map, with the effect on the transversal structure and the rectangular dual. On top, two kinds of essential cycles of size 4. Below, an essential cycle of size 8, enclosing 3 vertices, and at the bottom, an arbitrary oriented cycle.

- or they are cycles of length 8, enclosing several vertices and edges of T .

In both cases they are associated to an *alternating 4-cycle* in T : a quadrangle with two opposite blue edges and two opposite red edges. Also, this quadrangle is triangulated and the triangulation inside it carries a transversal structure. The reversal of the cycle corresponds to switching the colour of all edges inside the quadrangle, and reversing the orientation only of those of one colour, the (previously) red ones if the cycle was clockwise, and the (previously) blue ones if it was counterclockwise.

These transformations are still valid if the cycle that is reversed is a bigger cycle : the edges enclosed it change their colour, and part of them change their orientation.

In the program that I wrote to study the search for a square dual, navigation from one transversal structure to another is done by flipping essential cycles or general cycles. For a general cycle, the destination structure is determined by switching the colours of the edges enclosed in the region. The colouring then allows to determine the orientation of the edges, as mentioned in subsection

3.1.

3.4 Minimal transversal structure

A consequence of the lattice structure is that the triangulation T has a minimal (resp. maximal) transversal structure, whose corresponding α_4 -orientation has only counterclockwise (resp. clockwise) cycles.

In the program, when a graph is loaded, it is first given its minimal transversal structure, using an algorithm given in Fusy's thesis.

generation

Also, 4-connected triangulations of a 4-gon can be randomly generated implementing a procedure given in Eric Fusy's thesis. This uniform sampling allows, in linear time, to construct such a triangulation with n inner vertices, every such triangulation of size n having equal probability to be obtained.

The procedure is based on transversal structures, with the pleasant consequence that the generated triangulation already comes along with its minimal transversal structure. It starts by generating a binary word with n bits set to 1 and $2n + 1$ bits set to 0, then computes a ternary tree represented by this word, whose edges we colour in red and blue. After this, an operation called "closure" leads from the tree to a triangulation with the properties we expect.

4 Finding the good one

Now, with this knowledge of the set of transversal structures, which represents the set of rectangular duals of a given triangulation, our goal is to find one element in this set that corresponds to a square dual.

4.1 Writing equations

Consider once again a triangulation T , with a transversal structure.

Let us create a variable x_v for each vertex v , and a variable x_{uv} for each edge uv . To each vertex v , we associate four equations:

$$x_v = \sum_{u \xrightarrow{\text{blue}} v} x_{uv} ; x_v = \sum_{u \xrightarrow{\text{red}} v} x_{uv} ; x_v = \sum_{u \xleftarrow{\text{blue}} v} x_{uv} ; \text{ and } x_v = \sum_{u \xleftarrow{\text{red}} v} x_{uv} .$$

(for each of the four outer vertices, only one of the above equations has a meaning)

If there exists a solution to this equation system where all the variables have a non-negative value, then it describes a square dual that can be obtained by deforming the rectangular dual, changing dimensions of the rectangles, but keeping the layout (relative position of the inner rectangles) unchanged. Indeed, by giving each segment the length given by the corresponding variable, we obtain a rectangular dual where all the inner rectangles are squares: the rectangle representing vertex v is a square of side x_v (see figure 12).

All the above defined equations are homogeneous : in order to avoid the trivial zero solution, we add one last equation : $x_N = 1$, meaning the top of the outer rectangle has length 1.

We reach a total of $4n + 5$ equations, where n is the number of inner vertices. The number of variables is the sum of the number of vertices and the number of inner edges in T . Euler's formula

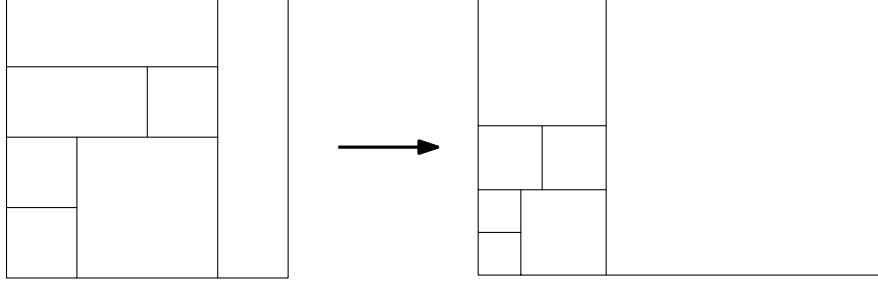


Figure 12: How a rectangular dual can be modified in proportions, but keeping the same layout, to become a square dual.

(in a planar map, the numbers of faces, vertices, and edges satisfy $f + v = e + 2$) implies that we have exactly $4n + 5$ variables as well. Since the system has as many variables as equations, if its determinant is non-zero, then it has exactly one solution.

The determinant

We can see the matrix M of this linear system as a kind of adjacency matrix of a bipartite graph shown in figure 13. Its black vertices are the variables, and its white vertices are the equations.

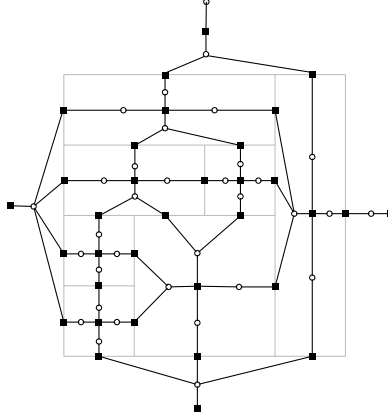


Figure 13: The derived graph associated to the equation system.

The coefficient M_{ij} is ± 1 when white vertex i and black vertex j are connected in the graph. The determinant is the sum $\sum_{\sigma} \prod_i M_{i\sigma(i)}$, and the product $\prod_i M_{i\sigma(i)}$ is non-zero if and only if σ encodes a perfect matching in the derived graph (white vertex i being matched with black vertex $\sigma(i)$). In this case it is equal to ± 1 , and we can define this to be the *signature* of the matching. It can be proved that the derived graph does have perfect matchings, and that, what is more, they all have the same signature. This result in the determinant being non-zero. Hence, the system has a unique solution, which, if non-negative, describes a square dual.

4.2 An algorithm

During this internship, I wrote a program to study an algorithm designed to find a square dual of a triangulation.

The idea is based on the equation system discussed above: for each transversal structure, we have a linear system, and a solution. We start with the minimal transversal structure. If the solution is non-negative, it represents a squaring, and our goal is reached.

Otherwise, in the solution, some values are negative. In this case, some edges in T have a positive value, and some have a negative value. Consider the angles lying between two edges of opposite signs, and take their representatives in the angular map. We have a set of edges in the angular map, call them *transition edges*, which constitute the separation between a negative “region” of the graph, and the rest of the graph. Hence, it is a family of closed paths.

Furthermore, this contour of the negative region, γ , is a family of *oriented* cycles in the angular map. This can be seen by observing that, to preserve consistency in the square dual, a black vertex or a white vertex must have as many incoming transition edges as outgoing ones.

From there, it seems natural to be tempted to reverse these cycles, obtain a new α_4 -orientation of the angular map, a new transversal structure, a new system, and a new solution. If this solution has only non-negative values, our goal is reached, otherwise, we can reiterate.

The big question is whether this process terminates by reaching a suitable transversal structure and obtaining a square dual, or it loops. Empirically, for all the tested triangulations, not only it doesn’t loop, but it finds a square dual relatively quickly (7 iterations generally suffice for graphs having around 100 vertices, and 9 are usually enough with 200 nodes).

4.3 Resolution

The program that I was writing had to solve this linear system. An important decision to take was whether the resolution had to be exact, or in floating point arithmetic. A floating point resolution was an easier solution, because many libraries provide this function. It would also perform faster than with rational numbers, which is important because the resolution is the only operation which could not be done in linear time : it was the critical part of the algorithm.

On the other hand, in order to make it easy for the human user to think about the values and the links between them, it was preferable to have exact values, in the shape of rational or integer numbers. Also, I initially thought that we could not afford to allow rounding errors in this resolution: if a very small value was interpreted as positive instead of negative, it would corrupt the boundary γ mentioned above, and if γ is not a family of oriented cycles, reversing it does not yield a new transversal structure, so in a nutshell everything would be corrupted if a very small value had the wrong sign.

I implemented a resolution with integer numbers, that is, the total width of the outer rectangle, initially supposed to be 1, grows throughout the resolution, to ensure that in the end all the variables have integer values. The use of a gcd function ensures that the outputted solution is the minimal integer solution.

Very quickly, the numbers become very big and the use of a special library for big integers becomes necessary: with 50 vertices it is frequent to go beyond the range of 2^{31} . With 100 vertices, the numbers usually have around 60 bits and with 200 they have 130. Furthermore, the numbers occurring during the system resolution can reach one million bits, even if the values in the solution are smaller.

These computations with very big integers made the resolution slow for big instances (starting from 200-300 nodes), and called for a faster resolution using floating point arithmetic. By observing the values obtained with the exact solution, I noticed that the very small values that would have induced a fault usually didn't occur. The sign of a floating point number can be mistaken due to rounding errors when it is smaller than 10^{-15} , and even with 100 or 200 vertices, the ratio between the biggest and smallest value was rarely more than 10^9 . So, by considering all values smaller than 10^{-14} as zero, and checking that all the other values are bigger than 10^{-11} , for instance, one can be pretty confident on the signs of the values. Otherwise the solution is simply declared unreliable. In the final version of the program, the user can therefore choose which kind of solution they want: the integer solution is more valuable because it is exact, but the floating point solution goes faster (it allows processing graphs up to 500-600 vertices in a few seconds, most of the time).

The worst case When wondering what was the size of the integers in the solution, the “Fibonacci” dissections came as an example of square dissections where the sides of the squares grow exponentially with n , the number of squares (figure 14). An interesting question to ask is: it it the

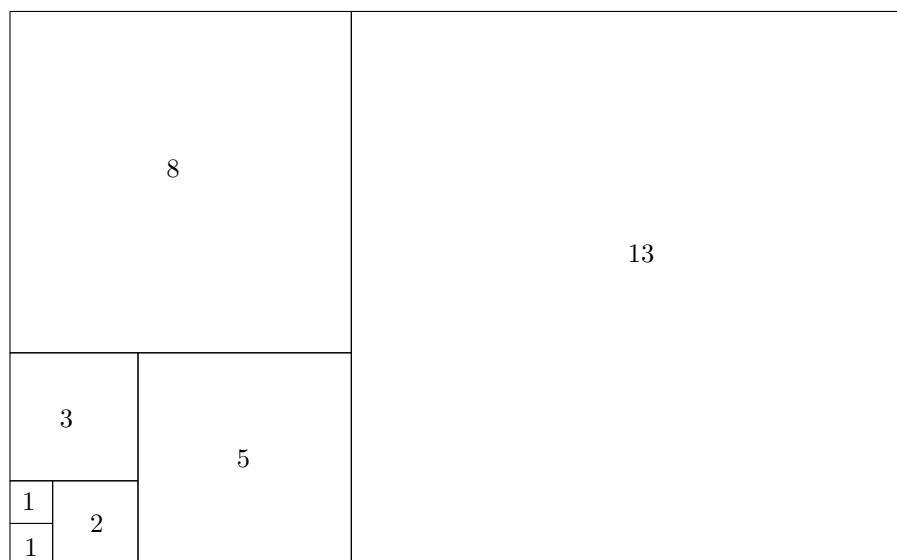


Figure 14: The “Fibonacci” dissections, where the size of the biggest square is given by the Fibonacci sequence

very worst case ? Is $fibonacci(n + 1)$ the maximum size of a square in a square dissection having n squares, where all the sizes have no common divisor other than 1 ?

Experimentally, with random graphs, the exponential factor (the number x such that x^n is the biggest side of the dissection) is most often close to 1.55, and not 1.618, as it is with the Fibonacci dissections.

5 Conclusion

5.1 My contribution

At the beginning of this internship, my ambition was to write the program in a relatively short time and then have a lot of time left to do more theoretical work. Actually, before I could reasonably start thinking about how I was going to make the program and write it, there was much to read and understand about planar graphs, transversal structures, things that are in this report and other things. Even while I was writing the program, I spent a lot of time reading and learning about the combinatorial objects I used. Also, the program was in the end more complete than I first thought it was going to be. So it took me more time than initially planned, and the most important product of my internship is this program.

This doesn't mean it was dull, stupid coding work. It was challenging to implement these algorithms, starting from their description in articles which focus more on the combinatorial aspects than on the algorithmical ones. In many cases, a proof describes a procedure that goes through many steps, considering many complex mathematical structures, whereas in order to implement it efficiently, one can find a shorter way to go directly from the start to the beginning, without using all the intermediary structures, that are only useful to make the proof rigorous and more readable (for example, though the algorithm described in 4.2 uses cycles in the angular map, the angular map is represented nowhere in the program : it serves only as a tool on the theoretical level). Finding this shorter way is an interesting exercise, that requires full understanding of the procedure, as well as creativity.

I spent some time trying to prove that a square dual is eventually reached, using the program I made, but in vain. However, there is good hope that my program will be used to study the algorithm and will help to decide whether or not it always finds a square dual.

During these two months, I also joined Jawaherul Alam, Stefan Felsner, Stephen Kobourov and Torsten Ueckerdt in their reflexions about cartograms, and tried to help them improve some bounds on the number of sides per polygon that are needed to represent this or that class of graphs, and other problems.

5.2 Before and after

Before I came, a student had done similar work about triangle contact representations of 4-connected triangulations. She had written a program too, and it was actually her work that gave Stefan Felsner the idea that I could do analogous work on square duals. Her work was very useful to me as an example and source of inspiration. Her program is used by members of the team who work on this topic, and this makes me think that my program will be used too.

Also, I started to write a “developer's” documentation of my program, to allow the people to apply changes to the program when I'm no longer there.

References

- [1] S. Felsner. Rectangle and square representations of planar graphs. 2011.
- [2] E. Fusy. *Combinatoire des cartes planaires et applications algorithmiques*. PhD thesis, LIX, Ecole Polytechnique, 2007.

- [3] G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theoretical Computer Science*, 172:175–193, 1997.
- [4] L. Lovász. Geometric representations of graphs. pages 95–100, 2009.

Appendix: Screenshots of the program

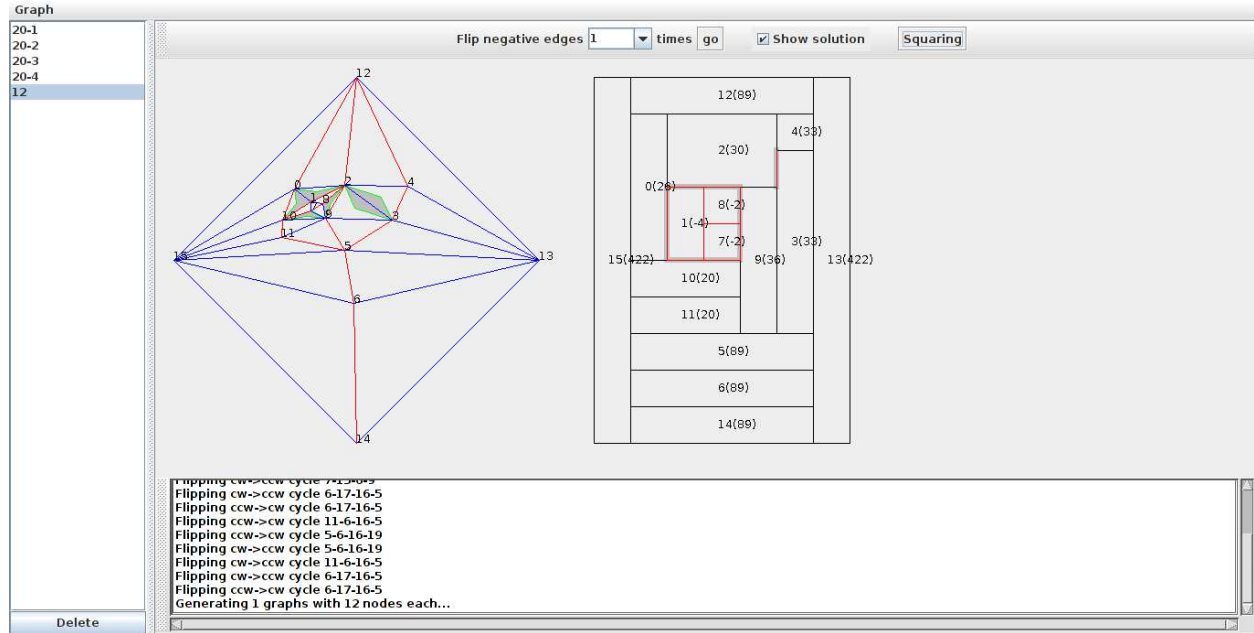


Figure 15: On the left, a generated graph with its minimal transversal structure, and on the right the associated rectangular dual. The number in parentheses in the rectangular dual are the sizes of the squares as dictated by the solution to the linear system. Some edges have negative values: they are displayed in red on the dual.

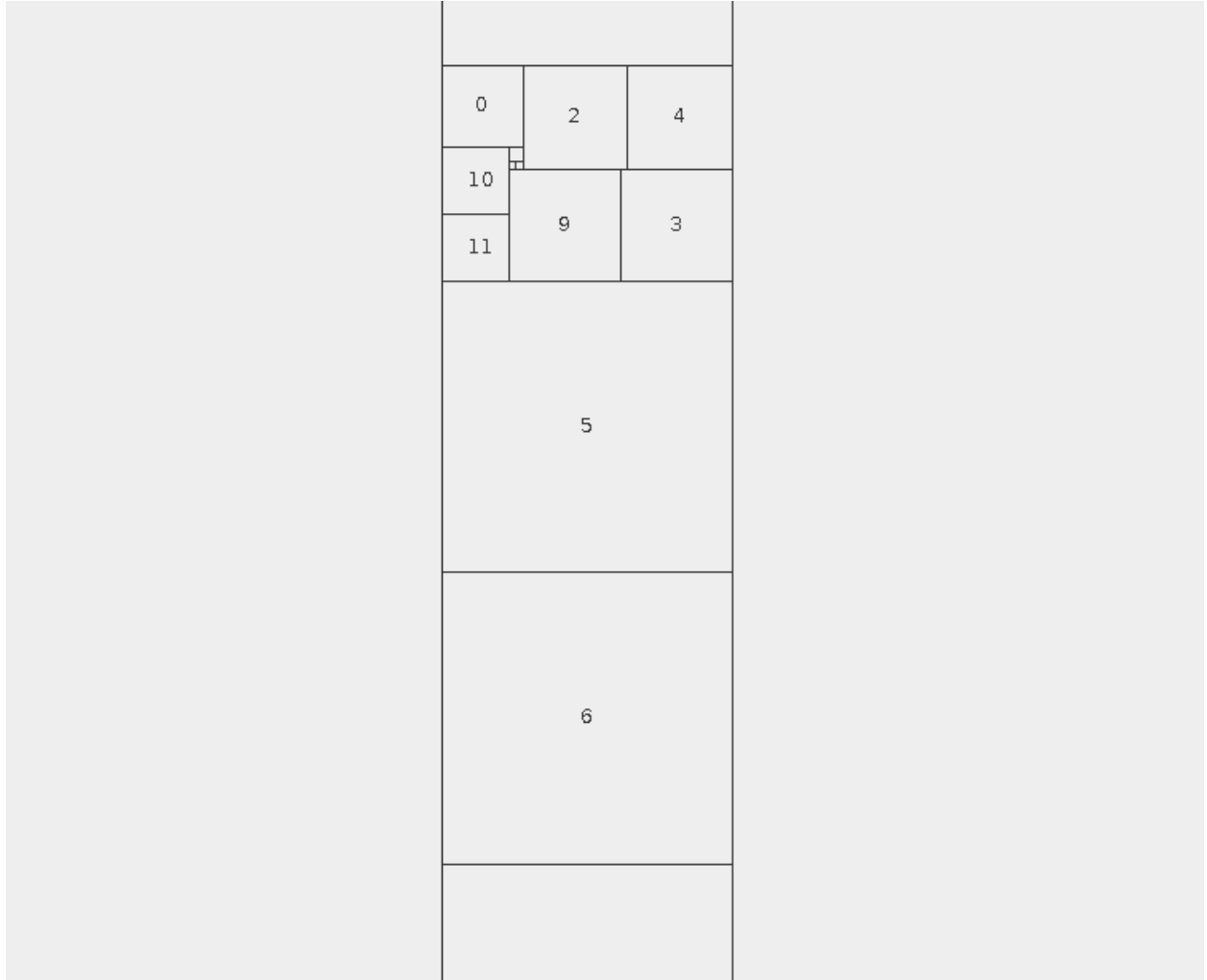


Figure 16: After a few iterations, a squaring can be found.