Lego like spheres and tori, enumeration and drawings

Mathieu Dutour Sikirić
Rudjer Bosković Institute, Croatia

Michel Deza
Ecole Normale Supérieure, France

September 30, 2016

I. Legos

$(\{a, b\}, k)$-maps

- By a $(\{a, b\}, k)$-map, we mean a k-valent map of genus g with faces of size a or b with $a < b$.
- If $g = 0$ we speak of $(\{a, b\}, k)$-plane graph and if $g = 1$ we speak of $(\{a, b\}, k)$-torus.
- Euler-Poincaré formula $V - E + F = 2 - 2g$ can be reformulated as

$$2k(2 - 2g) = \{2k - a(k - 2)\}\, p_a + \{2k - b(k - 2)\}\, p_b$$

- We have essentially three cases for the plane graphs:
  - $2k - b(k - 2) > 0$ (Elliptic case): finite number of possibilities.
  - $2k - b(k - 2) = 0$ (Parabolic case): p_a is constant (independent of p_b). Number of graphs growing polynomially in p_b.
  - $2k - b(k - 2) < 0$ (Hyperbolic case): p_a growing with p_b. Number of graphs growing at least exponentially in p_b (conjecture)

List of parabolic cases

k	(a,b)	smallest one	existence	conn.	p_a	v	$NrGr$
3	$(2,6)$	Bundle$_3$=$3 \times K_2$	$p_6 = T-1$	2	3	$2+2p_6$	2
3	$(3,6)$	Tetrahedron	p_6 is even	2 or 3	4	$4+2p_6$	5
3	$(4,6)$	Cube	$p_6 \neq 1$	3	6	$8+2p_6$	16
3	$(5,6)$	Dodecahedron	$p_6 \neq 1$	3	12	$20+2p_6$	28
4	$(3,4)$	Octahedron	$p_4 \neq 1$	3	8	$6+p_4$	18
4	$(2,4)$	Bundle$_4$=$4 \times K_2$	p_4 is even	2	4	$2+p_4$	5
6	$(2,3)$	Bundle$_6$=$6 \times K_2$	p_3 is even	2	6	$2+\frac{p_3}{2}$	22
6	$(1,3)$	Trifolium	$p_3 = 2T-1$	1	3	$\frac{1+p_3}{2}$	3

Notes:

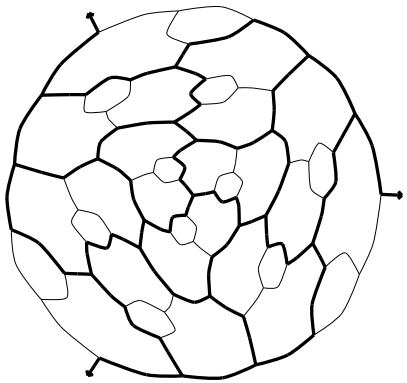
- T is a number of the form k^2+kl+l^2
- NrGr is the number of possible groups
- A graph is k-connected is after removing $k-1$ vertices it remains connected.

Definition of lego

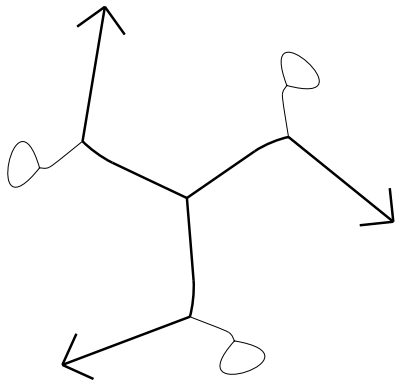
- A $(\{a, b\}, k)$-map M admits a lego decomposition if there exist a number m of cluster of faces such that the m clusters put together yield the map M.
- We impose in this work $m = \min(p_a, p_b)$. That is each cluster has either just one a-gon or just one b-gon. This implies

$$\frac{p_a}{p_b} \in \mathbb{N} \quad \text{or} \quad \frac{p_b}{p_a} \in \mathbb{N}$$

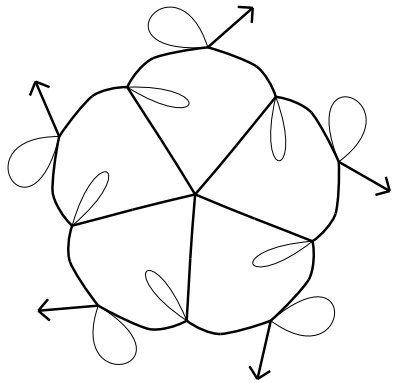
Examples:



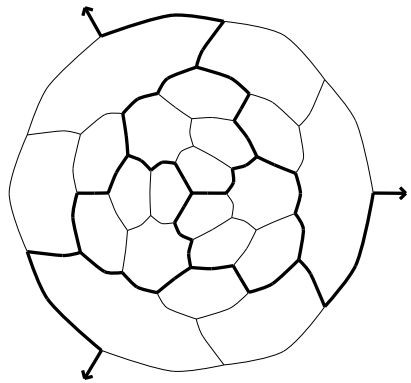
68, T



8, C_3



12, D_5



44 $D_{3h}(D_3)$

When do lego exists?

Plane graph case:

- In the elliptic cases we have a finite number of graphs to consider. Easy.
- In the parabolic cases:
  - If $\frac{p_a}{p_b}$ is integer then again a finite list of graphs to consider.
  - If $\frac{p_b}{p_a}$ is integer then we have an infinite set of possibilities and existence can be proved in all of them.
- In the hyperbolic cases:
  - If $\frac{p_b}{p_a}$ is integer then we can prove that there are only a finite number of possibilities.
  - If $\frac{p_a}{p_b}$ is integer then a priori infinite number of possibilities but existence is not proved.

Torus case:

- If $\frac{p_b}{p_a}$ is integer then we can prove that there are only a finite number of possibilities.
- If $\frac{p_a}{p_b}$ is integer then a priori infinite number of possibilities but existence is not proved.

Classification results I

For a hyperbolic $(\{a, b\}, k)$-sphere the number $\frac{p_b}{p_a}$ is an integer if and only if its p-vector is either $(p_1, p_3) = (\frac{k}{2}, \frac{k}{2})$, $k \geq 8$, $k \equiv 0 \pmod 4$ or one of 42 cases:

k	a,b	v	(p_a, p_b)
3	3,7	20	(6,6)
3	3,7	68	(12,24)
3	3,8	44	(12,12)
3	4,7	44	(12,12)
3	2,7	12	(4,4)
3	2,7	32	(6,12)
3	2,7	92	(12,36)
3	2,8	20	(6,6)
3	2,9	44	(12,12)
4	2,5	14	(8,8)
5	2,4	12	(10,10)
7	2,3	16	(14,28)

k	a,b	v	(p_a, p_b)
7	2,3	44	(28,84)
8	2,3	10	(16,16)
9	2,3	20	(36,36)
3	1,7	8	(3,3)
3	1,7	20	(4,8)
3	1,7	44	(6,18)
3	1,7	116	(12,48)
3	1,8	12	(4,4)
3	1,8	68	(12,24)
3	1,9	20	(6,6)
3	1,10	44	(12,12)
4	1,5	6	(4,4)

Classification results II

k	a,b	v	(p_a, p_b)
4	1,5	22	(8,16)
4	1,6	14	(8,8)
5	1,4	4	(4,4)
5	1,4	52	(20,60)
5	1,5	12	(10,10)
6	1,4	5	(6,6)
7	1,3	4	(4,8)
7	1,3	16	(7,35)
7	1,3	44	(14,98)

k	a,b	v	(p_a, p_b)
7	1,3	100	(28,224)
8	1,3	10	(8,24)
8	1,3	26	(16,64)
8	1,4	10	(16,16)
9	1,3	20	(18,54)
9	1,4	20	(36,36)
10	1,3	7	(10,20)
12	1,3	14	(24,48)
13	1,3	28	(52,104)

Notes:

- Of those cases, only the $(\{1,3\},7)$-spheres with 16 vertices cannot be realized as a lego.

Classification results III

The parameters of putative $(\{a, b\}; k)$-tori with integer $\frac{p_b}{p_a}$

k	a,b	v	$\frac{p_b}{p_a}$	k	a,b	v	$\frac{p_b}{p_a}$	k	a,b	v	$\frac{p_b}{p_a}$
3	3,7	$8p_3$	3	4	2,5	$3p_2$	2	4	1,5	$4p_1$	3
3	3,9	$4p_3$	1	4	2,6	$2p_2$	1	4	1,7	$2p_1$	1
3	4,7	$6p_4$	2	5	2,4	$2p_2$	2	6	1,4	$\frac{3}{2}p_1$	2
3	4,8	$4p_4$	1	6	2,4	p_2	1	6	1,5	p_1	1
3	5,7	$4p_5$	1	7	2,3	$2p_2$	4	7	1,3	$4p_1$	9
4	3,5	$2p_3$	1	8	2,3	p_2	2	8	1,3	$2p_1$	5
3	2,7	$10p_2$	4	10	2,3	$\frac{1}{2}p_2$	1	10	1,3	p_1	3
3	2,8	$6p_2$	2	3	1,7	$12p_1$	5	10	1,4	$\frac{1}{2}p_1$	1
3	2,10	$4p_2$	1	3	1,11	$4p_1$	1	14	1,3	$\frac{1}{2}p_1$	2

I. Enumeration methods

List of problems to be solved

- Problem I is:
  - We have a list of p'_a a-gonal faces and p'_b b-gonal faces
  - We want to find all possible lego pieces

  This is done, when possible, by exhaustive enumeration by adding pieces one after the other in all configurations.
- Problem II is:
  - We have a list of maps on the sphere or on torus
  - We want to find all legos that occur here.

  This kind of enumeration is limited to the $(\{a, b\}, k)$-map classes for which enumeration is feasible and there is a program for it.
- Problem III is:
  - We have a set of lego pieces
  - We want to find all the ways in which they can fit together.

  This kind of method is a priori more clever since we build first the list of pieces and then put them together.

List of feasible cases

- The program `CPF` (by Thomas Harmuth, Master Thesis) can enumerate the $(\{a, b\}, 3)$-spheres with a fixed number of vertices and $a \geq 3$, which are the most important cases.
- The program `CGF` (by Thomas Harmuth, PhD Thesis) can enumerate all the $(\{a, b\}, 3)$-maps of fixed genus and number of vertices with $a \geq 3$.
- The program `ENU` can enumerate the $(\{a, b\}, 4)$-spheres with fixed number of vertices and $a \geq 3$.
- The program `plantri` can do specific plane graph enumeration but it is much slower and hard to use.

In all other cases, we are on our own to get the maps. Possible methods is to reduce to one of the above classes. For some cases, e.g. $(\{3, 7\}, 3)$-spheres with 68 vertices, the program cannot terminate. Reductions are then needed.

Exact covering problem

Problem is:

- Given n points and m subsets $S_1, \ldots, S_m$ of $\{1, \ldots, n\}$
- to find all partitions of $\{1, \ldots, n\}$ by subsets S_i.

Features:

- The existence problem can be formulated as Integer Programming Problem and it is NP.
- It is an exhaustive enumeration problem and so harder a priori.
- Fortunately, there exist the program `LibExact` by Kaski and Pottonen which implements a very fast enumeration procedure using "dancing links".
- It does not uses symmetry so we have to do isomorphism checks afterwards.

Satisfiability problems SAT

A satisfiability problem (SAT) is a problem of the type:

- A number n of variables $v_1, \ldots, v_n$ that can be true or false.
- A number of clauses of the form

$$w_1 \vee w_2 \vee \cdots \vee w_p \text{ with } w_j = v_{i_j} \text{ or } \overline{v_{i_j}}.$$

- A set of clauses to be satisfied, i.e.

$$c_1 \wedge c_2 \wedge \cdots \wedge c_M.$$

The goal is to find whether there exist a set of variables that satisfies all the clauses.
The program `minisat` can check satisfiability very fast despite SAT being a NP problem. This allows to solve some combinatorial problems. We can also enumerate all the solutions of a SAT problem.

SAT for legos

SAT problems can be used to solve Sudoku, N-queens problems and generally all kinds of combinatorial problems. What about legos:

- We take a set of N lego pieces, each identical and each having p sides.
- We need to put the condition of adjacency between the sides. So, this makes about $(Np)^2$ variables.
- We have conditions around the edges since we want the degree to be a specified value of say k.
- But we cannot handle questions of connectivity.

In general the approach failed and this seems to be generally the experience when using SAT to solve combinatorial problems such as t-designs, distance regular graphs, etc.

Direct enumeration method

So, instead of SAT, we used a more classical enumeration method:

- The idea is to take one lego and add pieces one at a time until the obtained graph is complete.
- In the case of place graph, one can prove that we can add the pieces so that at all time the space that is not covered is connected.
- The method is typical backtrack enumeration and is all done in C++.

Results:

- The method generally works for up to say 8-12 lego pieces.
- This allows to solve many cases.

I. Graph drawing

Problematic

- The problem that we have is how to represent a graph on the plane or torus in a practical way.
- The difficulty is how to represent 1-gons and 2-gons, which most method do not.
- Another issue is that many methods require the graph to be 3-connected, which is a strong requirement.

Additional wishes

- We want the program to be as fast as possible. Drawing should be a non-issue
- We want details to be clearly visible.
- We want the symmetries to be visible.

Representation of oriented maps

- General maps are best represented via flag system (buildings, chamber system, etc.)
- For oriented maps, we can use a simpler yet equivalent system: directed edge.
- Directed edge are basically a pair (v, e) with v a vertex and e an edge.
- Given a directed edge $\overrightarrow{e}$ we can build the next directed edge $n(\overrightarrow{e})$ around the vertex (in trigonometric order) and the inverse directed edge $i(\overrightarrow{e})$
- Vertices, edges and faces the correspond to the orbits of the permutations n, i and $n \circ i$ on the set of directed edges.

Example:

```
16
 8 11 4 9 2 15 14 13 0 3 12 1 10 7 6 5
 1 2 3 0 5 6 7 4 9 10 11 8 13 14 15 12
```

Existing approaches

- Tutte: He proposed to use eigenvectors of the adjacency matrix in order to provide embeddings. The method works well for plane graphs but suffer from one key problem: the inner faces tend to be very small and not visible. Only for plane graphs.
- Dress,Harmuth,Delgado Friedrichs,Brinkmann: The `CaGe` program uses an iterative process in order to get the embedding. A priori only for plane graphs.
- Mohar: Primal-Dual circle packings provide a good theoretical based approach for finding the embeddings. It works for plane graph, torus and hyperbolic maps.
- Force directed graphs. A physical functional such as

$$F = \sum_{e=(i,j)\in E(G)} f(\|x_i - x_j\|)$$

and we minimize over the embedding.

Issues and features

Features:

- All above techniques will respect symmetries since they are based on minimization procedure. But sometimes the number of iterations needs to be adjusted.
- All coordinates are obtained by iteration procedures. We cannot do exact arithmetic computations.

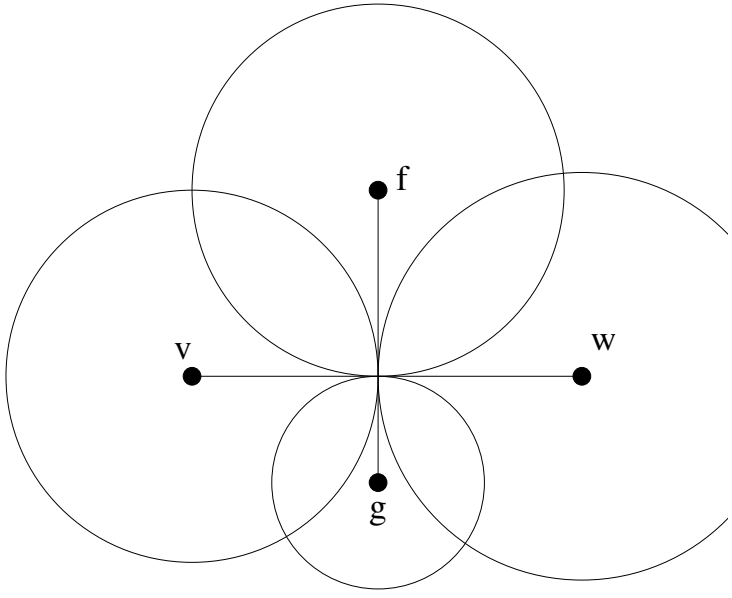
Issues:

- Some technique requires 3-connectivity of the graph
- All fail to work for 2-gons and 1-gons.
- Convergence might take a long time to achieve.
- Some details of the graph might be very hard to see in the final drawing. According to the cases
  - For the plane graph, one factor is the proximity to the exterior face, another is the level of refinement.
  - For the torus, it is flat so the only problem is the level of refinement.
  - For the hyperbolic plane, the problem will necessarily occur in all reasonable representations such as Poincaré plane or similar.

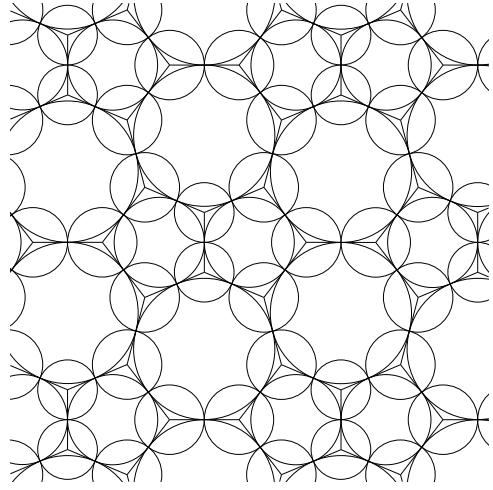
Primal-Dual circle packings

The idea is to put circles in the vertex center and faces such that

- If any two vertices v and v' are adjacent then the corresponding circles are tangent.
- For any face the circle is tangent to the edges.



The local picture of a primal-dual circle representation



The edges, circle and face circles of a primal-dual representation

Primal-Dual circle packings: Equations and Numerics

- In the case of torus, the equations that are satisfied are

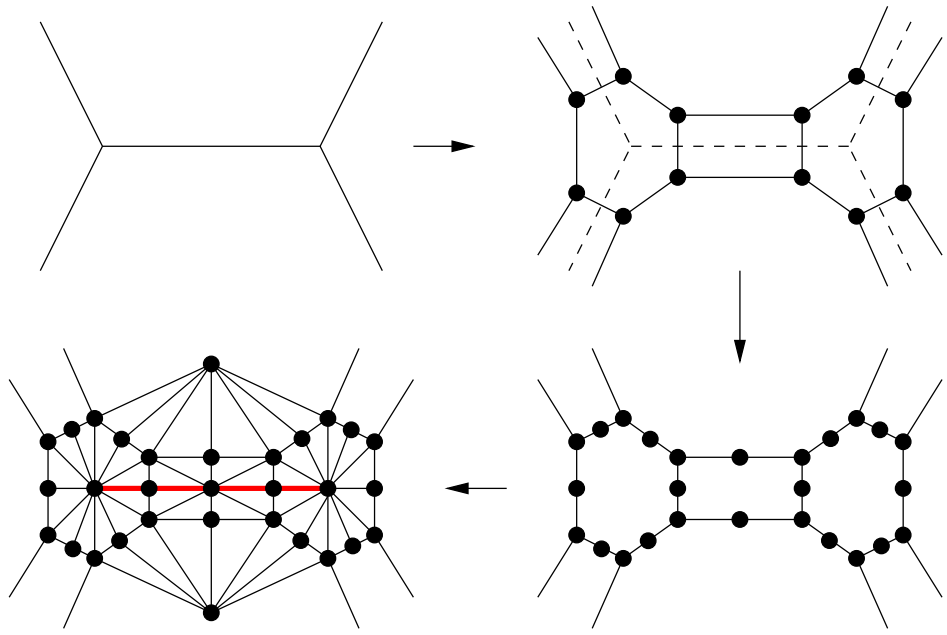
$$\pi = \phi_v = \sum_{uv \in E(Med^*(G))} \arctan\left(\frac{r_u}{r_v}\right)$$

  for each vertex v of the dual medial graph of G.
- Bojan Mohar has given an algorithm for computing primal dual circle packings. It consists of computing the defect at every node and increasing/decreasing the radius value according to $\phi_v > \pi$ or $\phi_v < \pi$. It is a geometric method but in some cases it is very slow.
- Instead our approach is to use a variant of Newton method: We choose the direction of change from the Newton iteration solver and we adjust the amount of change so that we remain in the allowed region of circles of positive radius.

$$x^{(n+1)} = x^{(n)} - c\frac{f\left(x^{(n)}\right)}{f'\left(x^{(n)}\right)} \text{ with } 0 < c \leq 1$$

Refinement techniques

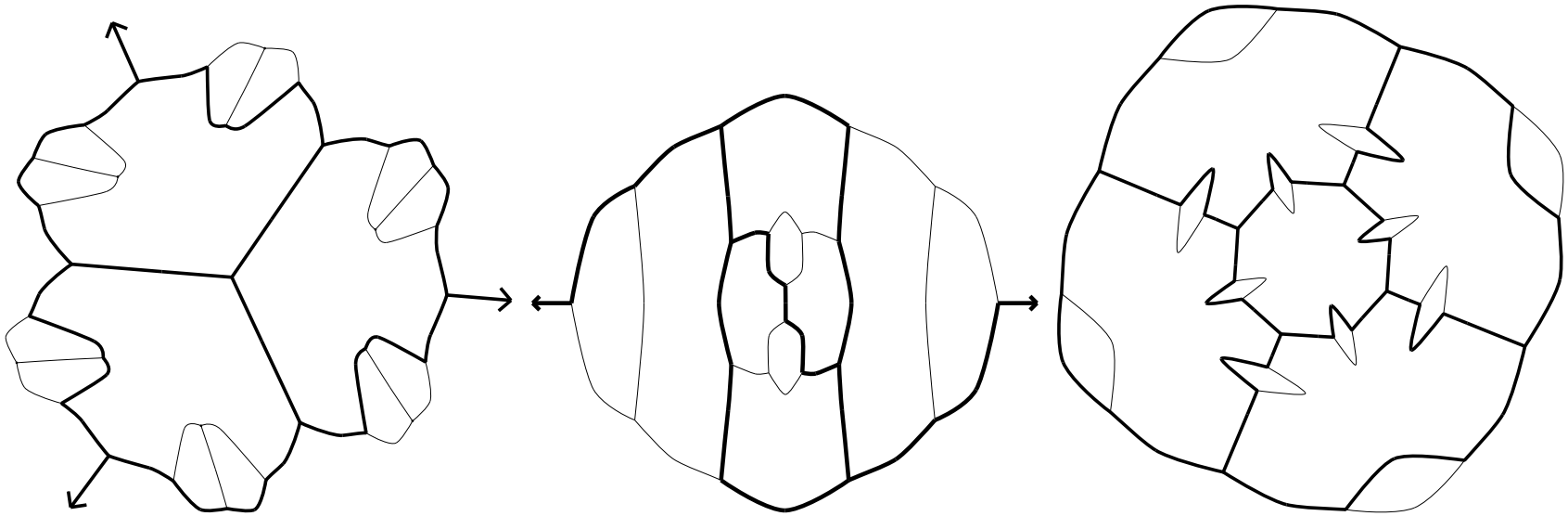
- The primal-dual technique requires 3-connectivity and will not work with 2-gons and 1-gons.
- The technique is to refine it. First for a map M we replace it with the order complex map $Ord(M) = Trunc(Med(M))$.
- Then we insert a vertex and each edge.
- Finally, we put a vertex on each face and connect it with all incident vertices.
- The resulting triangulation is 3-connected.



Symmetric representation

- For plane graph, we want to represent the maps with the maximum amount of symmetry.
- In practical matter if we have an axis of symmetry of order N then we want it visible.
- If the axis pass by a vertex or an edge then we put it to infinity.

Examples:



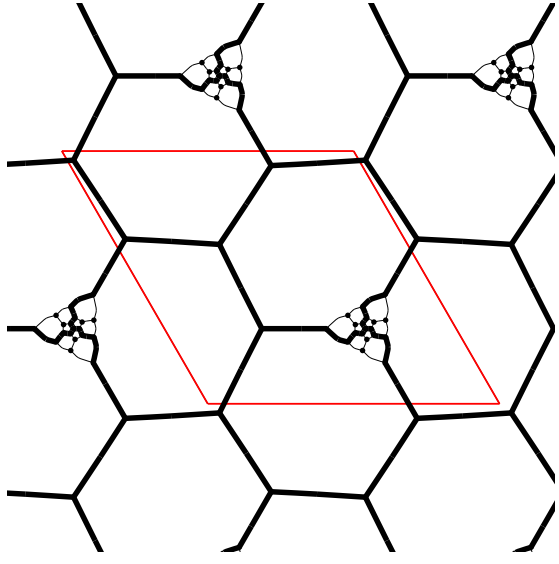
On vertex On edge On face

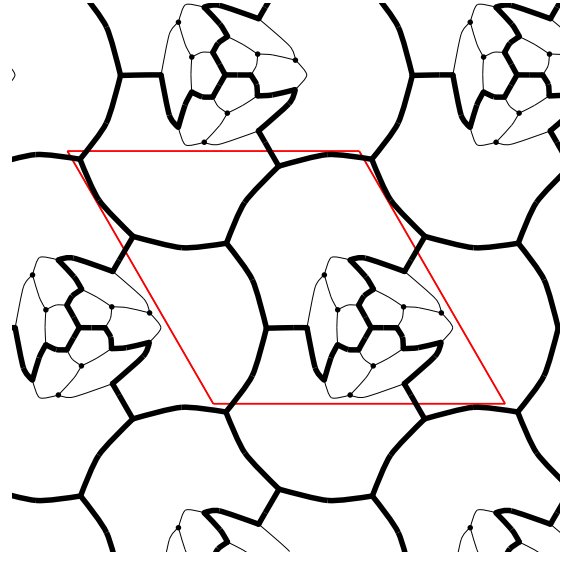
CaGe process

- Suppose that we have a point x and m adjacent points x^1, $\ldots$, x^m then for the CaGe process we have the equation

$$x = \frac{1}{\sum_{i=1}^{m} A_T^2(x, x^i, x^{i+1})} \sum_{i=1}^{m} A_T^2(x, x^i, x^{i+1}) \frac{x + x^i + x^{i+1}}{3}$$

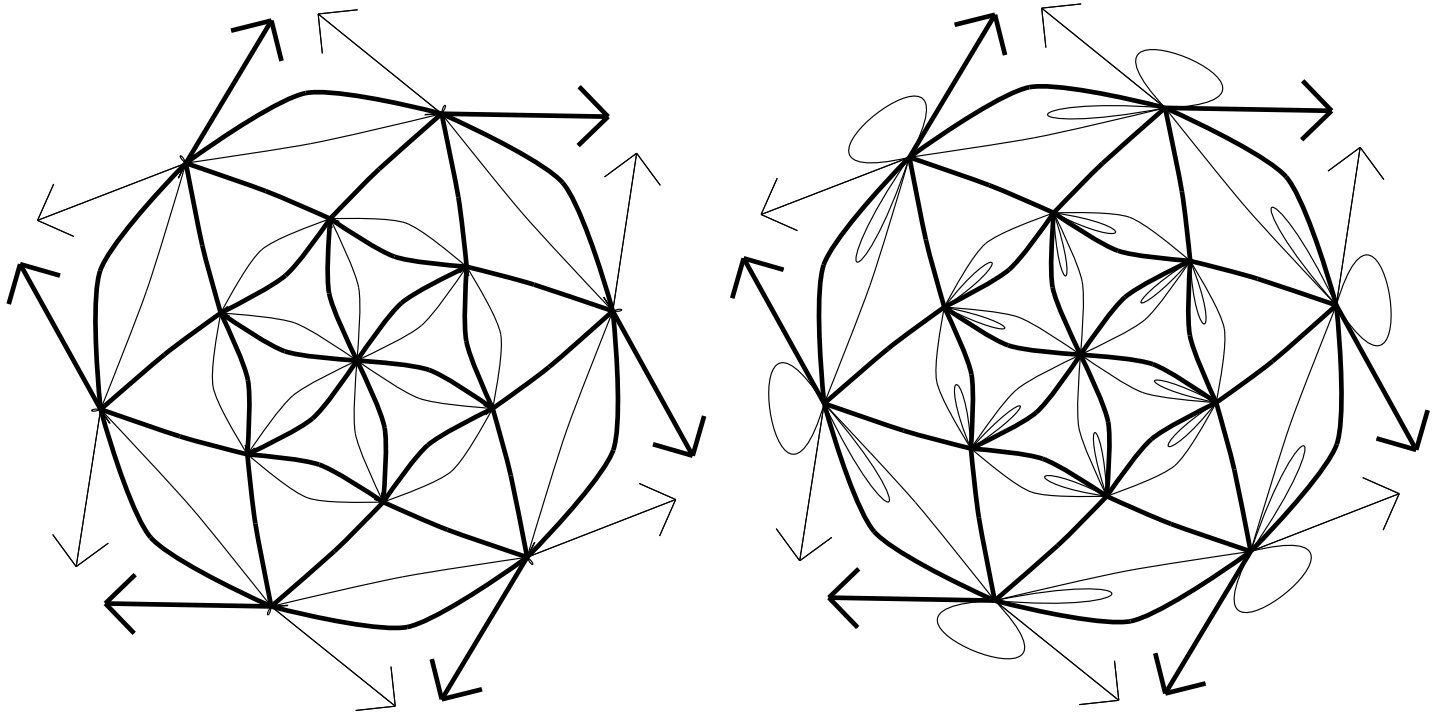
  with $A_T(x, y, z)$ the area of the triangle of vertices x, y and z.
- The equation is solved by fixed point iterations.
- Plane graph: put the vertices of the exterior face in a circle.
- Torus map: first guess obtained by primal dual circle packing and then apply the CaGe process.





Problem of 1-gons

- Our approach of auxiliary graphs works fairly well with 2-gons.
- But with 1-gons, we tend to get far smaller faces than expected.
- The empirical solution is to rescale the 1-gons by large factors (say 200) so that they become visible.



normal expanded

Software availability

- Source code is available at

  https://github.com/MathieuDutSik/Plot_orientedmap
- Code written in `C++11` language.
- Uses the Eigen library for matrix computations.
- Input file in Namelist, fortran style input.
- Output file in `svg` (Scalable Vector Graphics) to be used on web page or in Inkscape.