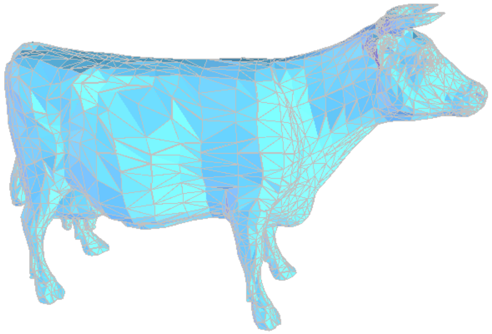
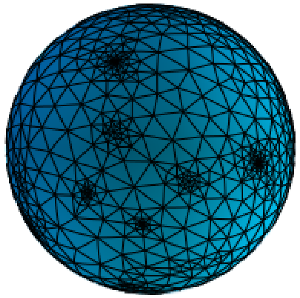




Fast spherical drawing of planar triangulations: an experimental study of graph drawing tools

june 27-29 2018, SEA2018, L'Aquila

Luca Castelli Aleardi

Gaspard Denis

Éric Fusy

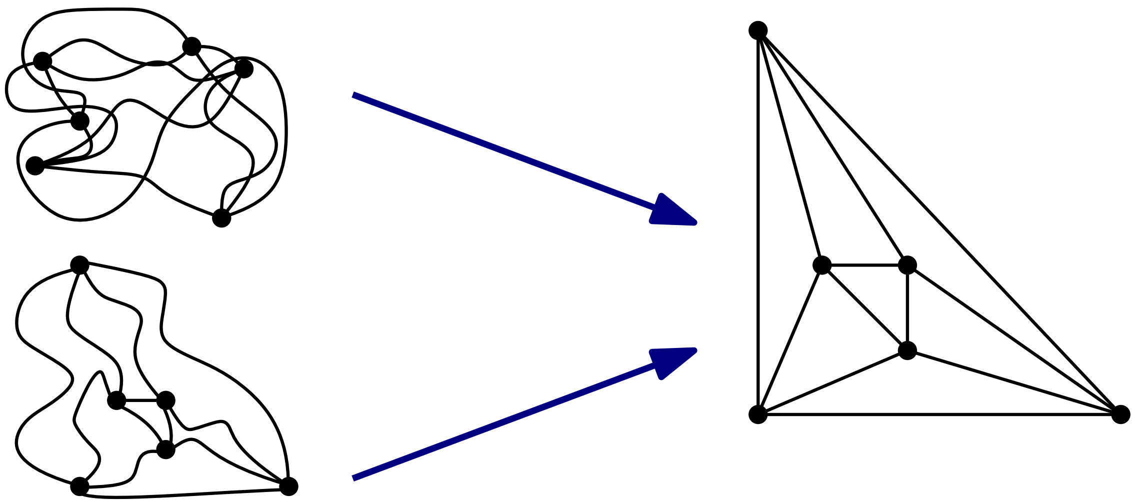
(planar) graph drawing

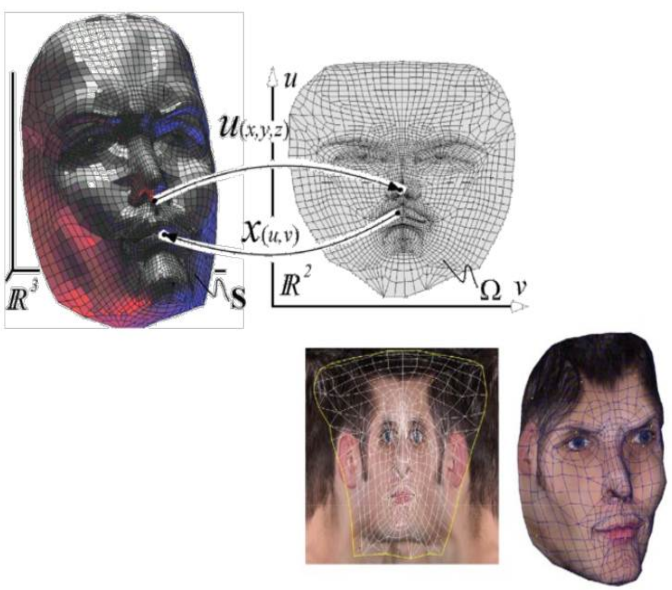
(a very short survey)

Straight-line planar drawings of planar graphs

Problem definition

Input: a planar graph (or planar map)
Output: a straight-line planar drawing (crossing-free)

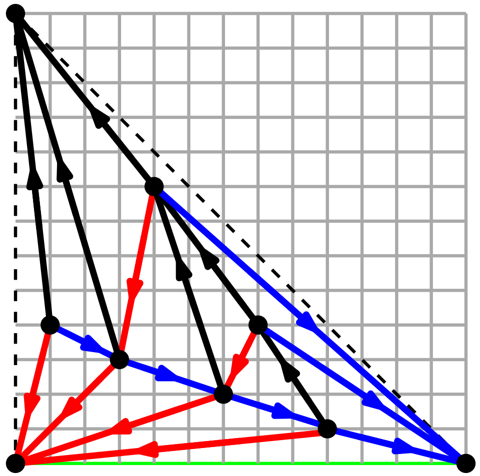




Bennis et al., 1991
Maillot et al., 1993

Straight-line planar drawings of planar graphs

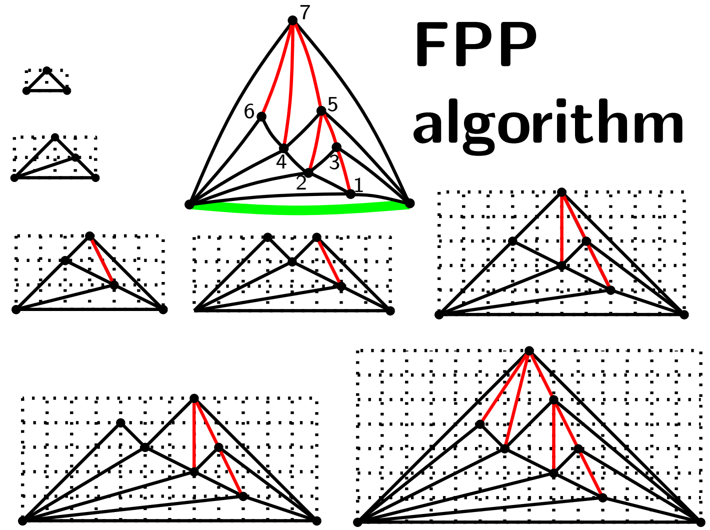
Thm (Schnyder 1990)



face counting via Schnyder woods

Thm (De Fraysseix, Pack Pollack 1989)

FPP algorithm



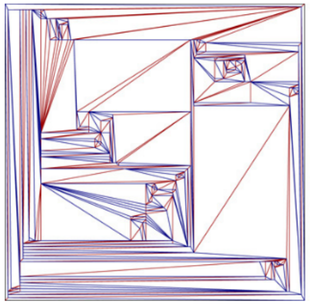
shift algorithm via Canonical orderings

linear time algorithms

$O(n) \times O(n)$ **grid drawings**

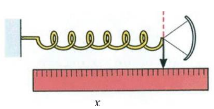
not trivial to implement

extremey fast: they can process millions of vertices per second



(images by E. Fusy)

Spring embedder (Eades, 1984)
(Fruchterman and Reingold, 1991)



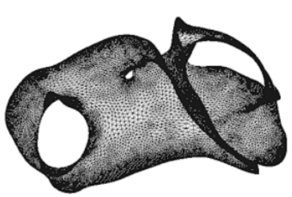
force-directed paradigm

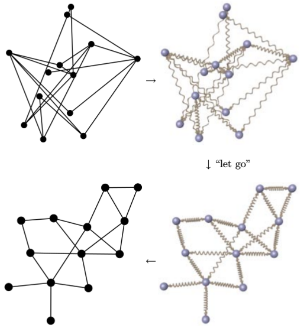
easy to implement

pretty slow: $O(n^2)$ or $O(n \log n)$ time per iteration

$$\mathbf{F}_a(v) = c_1 \cdot \sum_{(u,v)\in E} \log(dist(u,v)/c_2)$$

$$\mathbf{F}_r(v) = c_3 \cdot \sum_{u \in V} \frac{1}{\sqrt{dist(u,v)}}$$





images from Kaufman Wagner (Springer, 2001)

[Tutte'63] Tutte barycentric embedding

minimize the spring energy

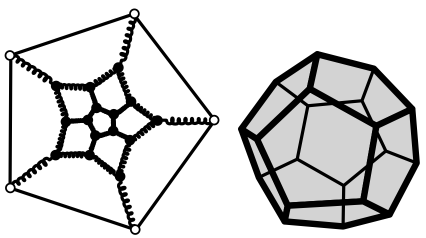
$$E(\rho) := \sum_{(i,j)\in E} |\mathbf{x}(v_i) - \mathbf{x}(v_j)|^2 = \sum_{(i,j)\in E} (x_i - x_j)^2 + (y_i - y_j)^2$$

solve large sparse linear systems

$$\mathbf{x}(v_i) = \sum_{j \in \mathcal{N}(i)} \frac{1}{deg(v_i)} \mathbf{x}(v_j)$$

easy to implement

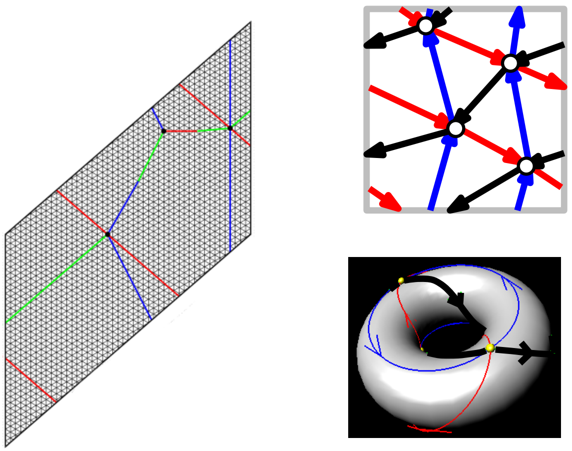
not very fast: they can process $\approx 10^4$ vertices per second



Straight-line drawings of toroidal graphs ($g = 1$)

(Goncalves Leveque 2014)

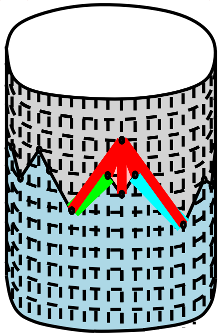
$O(n^2) \times O(n^2)$ **grid drawing**



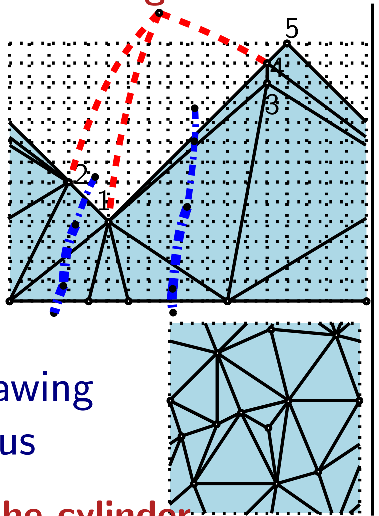
face counting in the universal cover

(Castelli Aleardi, Devillers, Fusy 2012)

$O(n) \times O(n\sqrt{n})$ **grid drawing**



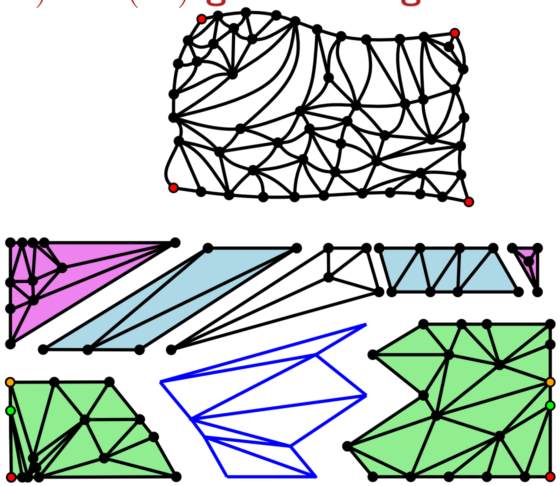
periodic drawing
flat torus



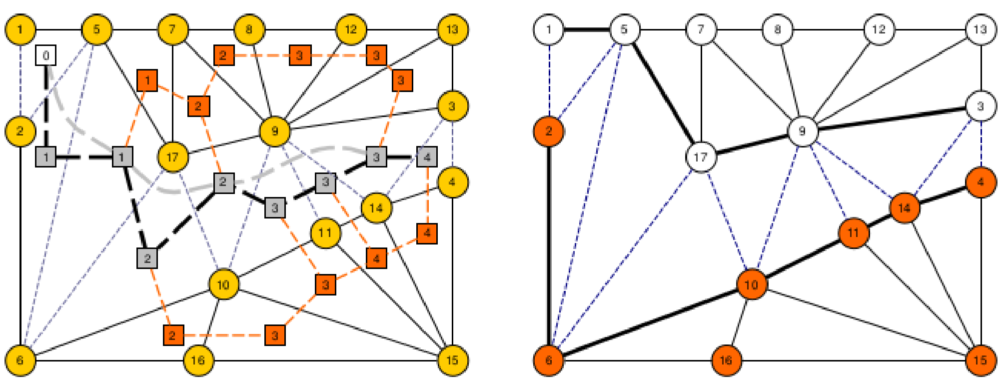
shift algorithm on the cylinder

(Castelli Aleardi, Fusy, Kostrygin 2014)

$O(n^4) \times O(n^4)$ **grid drawing**



(Duncan, Goodrich, Kobourov 2011)



cut along a polygonal scheme + shift algorithm

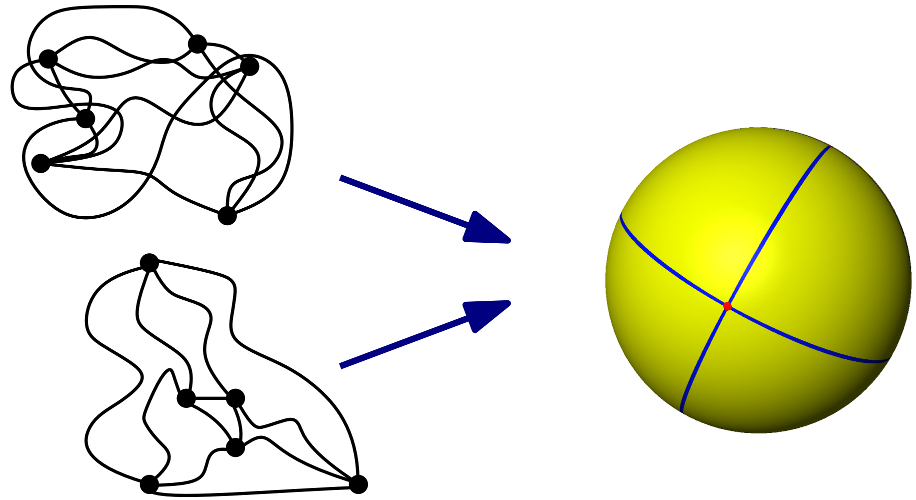
Spherical drawing and parameterizations

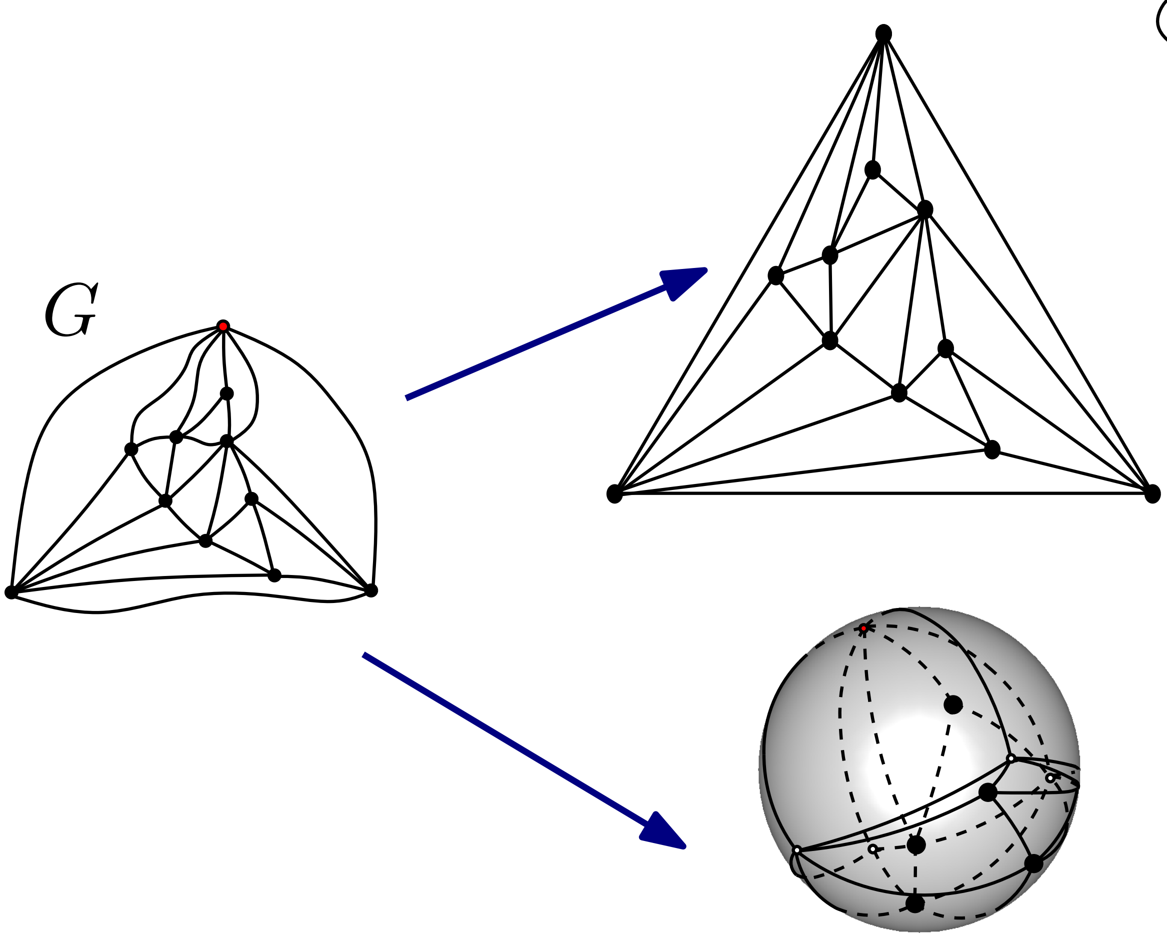
Spherical drawings/parameterizations of planar graphs

Problem definition

Input: a planar graph (or planar map)

Output: a geodesic spherical drawing (crossing-free)

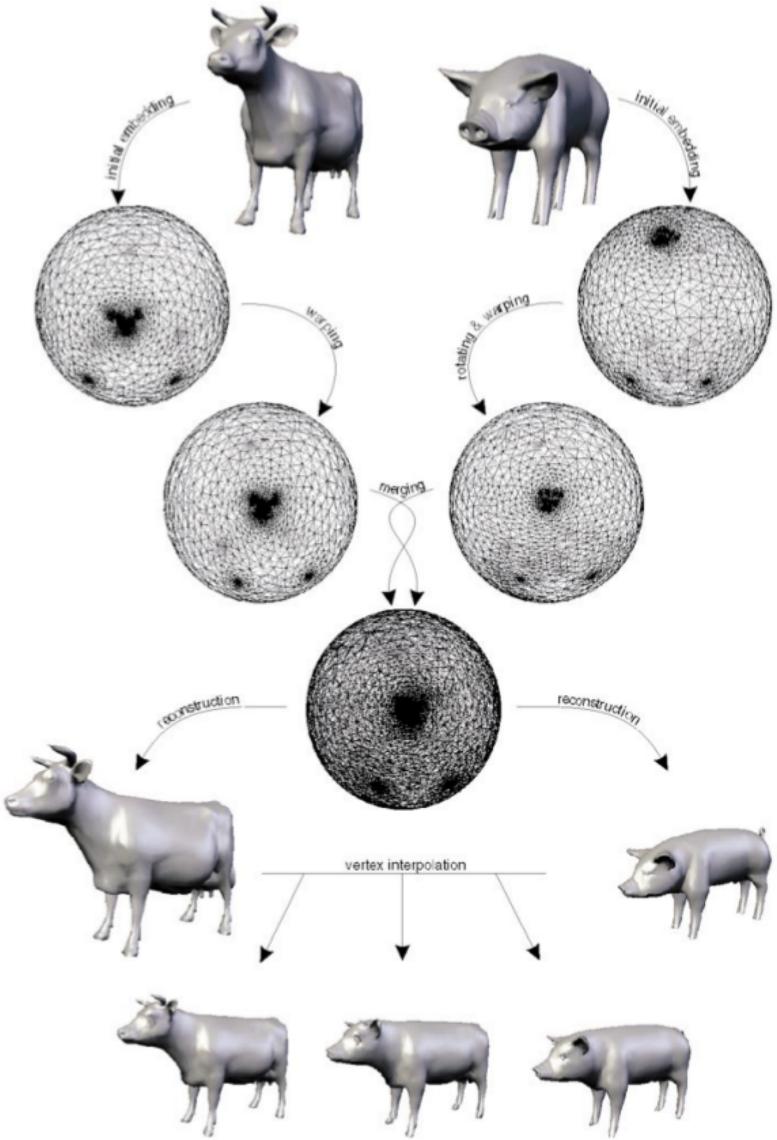




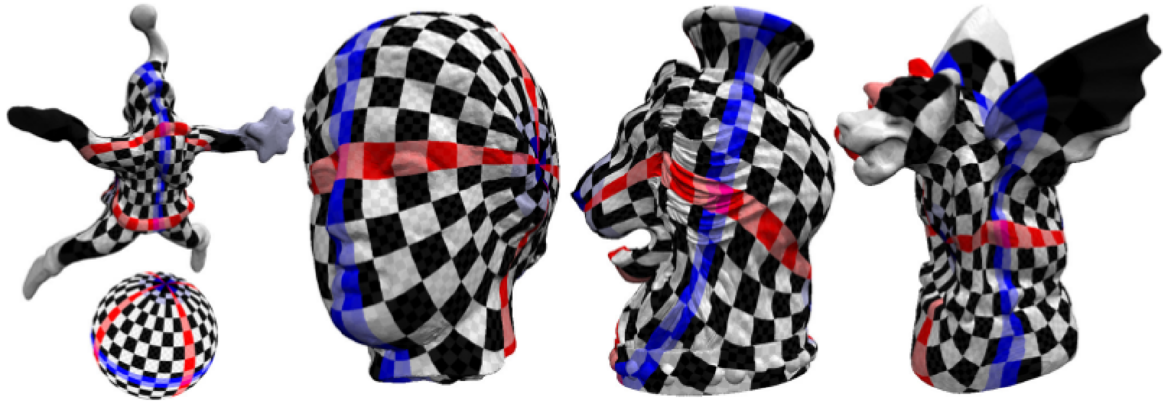
Spherical parameterization: applications

Morphing

(Alexa, 2000)



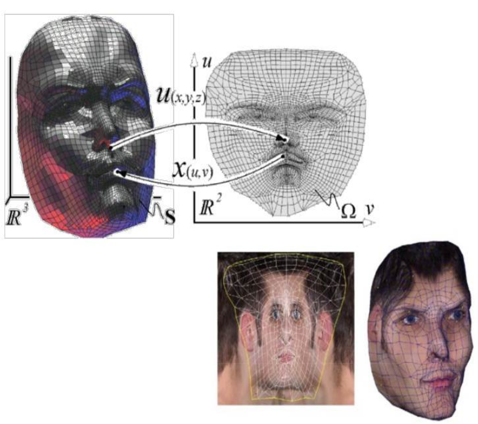
Texture mapping



(image by Friedel, Schroder, Desbrun, 2007)

Spherical vs. planar parameterization

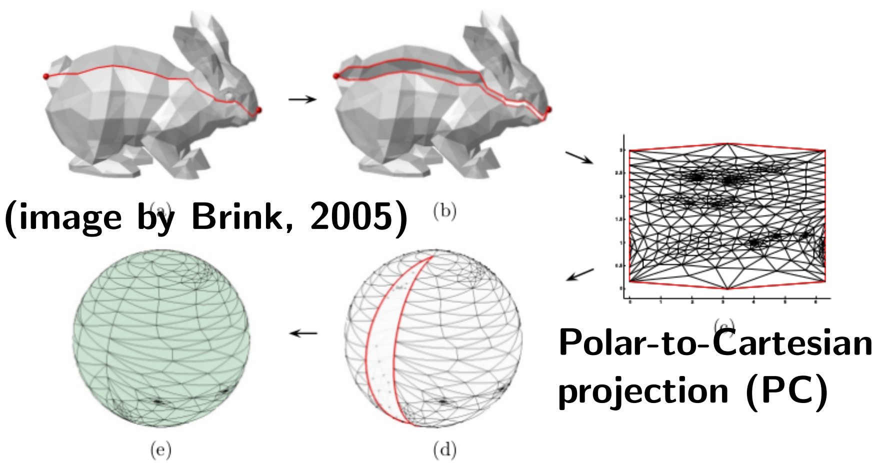
Main advantage: the spherical domain allows for seamless and continunous parameterization in the case of genus 0 meshes (without boundaries)



Bennis et al., 1991
Maillot et al., 1993

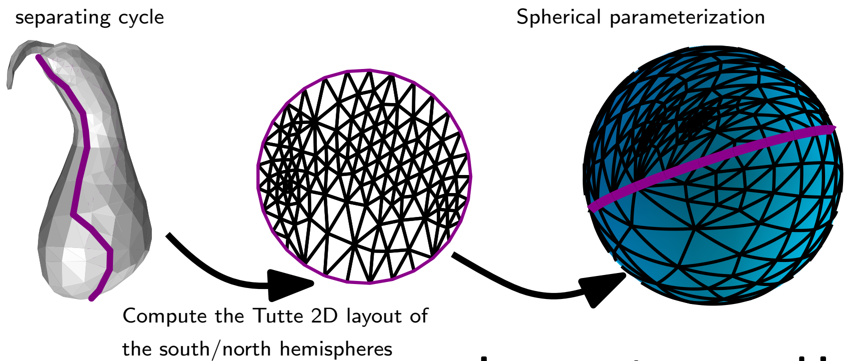
Spherical drawings/parameterizations: related works

Spherical parameterizations: use planar parameterization + spherical projections (**not crossing-free**)



(image by Brink, 2005)

Polar-to-Cartesian projection (PC)



Inverse stereographic projection (ISP)

compute a **Steinitz representation** and project on the sphere
(Shapiro, Tal, 1998)

Non linear optimization: (try to) solve Tutte equations on the sphere

(Alexa, 2000)
(Gotsman Gu Sheffer, 2003)
(Sheffer Gotsman Dyn, 2004)
(Saba, Yavneh, Gotsman, Sheffer, 2005)
(Zayer, Rössl, Seidel 2006)
(Friedel, Schroder, Desbrun 2007)

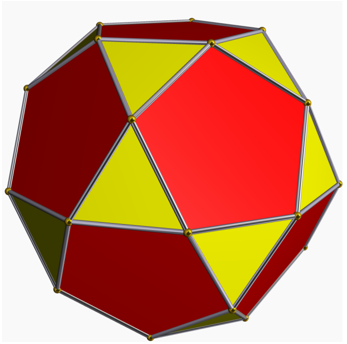
(Aigerman, Lipman, 2015)
(Aigerman, Kowalsky, Lipman, 2017)

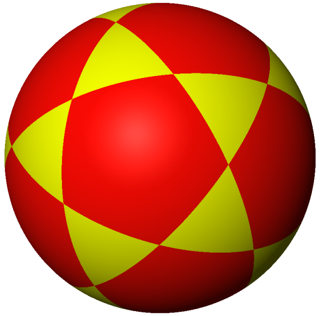
Extends graph drawing tools on the sphere: use non-euclidean spring embedders
(Kobourov Wampler, 2005)

Spherical drawings/parameterizations: related works

Spherical parameterizations: use planar parameterization + spherical projections (**not crossing-free**)

(layouts from Wikipedia)





(Pak, Wilson, 2017)

best upper bound on the grid size:

$$O(n^3) \times O(n^5) \times exp(O(\sqrt{n}\log n))$$

compute a **Steinitz polyhedral representation** and project on the sphere

(Shapiro, Tal, 1998) it takes $O(n^2)$ in the worst case

Non linear optimization: (try to) solve Tutte equations on the sphere

(Alexa, 2000)

(Gotsman Gu Sheffer, 2003)

(Sheffer Gotsman Dyn, 2004)

(Saba, Yavneh, Gotsman, Sheffer, 2005)

(Zayer, Rössl, Seidel 2006)

(Friedel, Schroder, Desbrun 2007)

(Aigerman, Lipman, 2015)

(Aigerman, Kowalsky, Lipman, 2017)

Extends graph drawing tools on the sphere: use non-euclidean spring embedders

(Kobourov Wampler, 2005)

Spherical drawings/parameterizations: related works

Spherical parameterizations: use planar parameterization + spherical projections (**not crossing-free**)

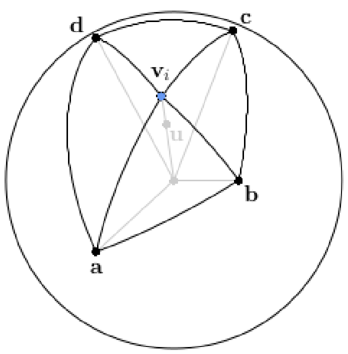
Non linear optimization: (try to) solve Tutte equations on the sphere

(Alexa, 2000)
(Gotsman Gu Sheffer, 2003)
(Sheffer Gotsman Dyn, 2004)
(Saba, Yavneh, Gotsman, Sheffer, 2005)
(Zayer, Rössl, Seidel 2006)
(Friedel, Schroder, Desbrun 2007)

(Aigerman, Lipman, 2015)
(Aigerman, Kowalsky, Lipman, 2017)

$$x_i^2 + y_i^2 + z_i^2 = 1 \qquad i = 1,..,n$$
$$\alpha_i x_i - L_W[i]x = 0 \qquad i = 1,..,n$$
$$\alpha_i y_i - L_W[i]y = 0 \qquad i = 1,..,n$$
$$\alpha_i z_i - L_W[i]z = 0 \qquad i = 1,..,n$$

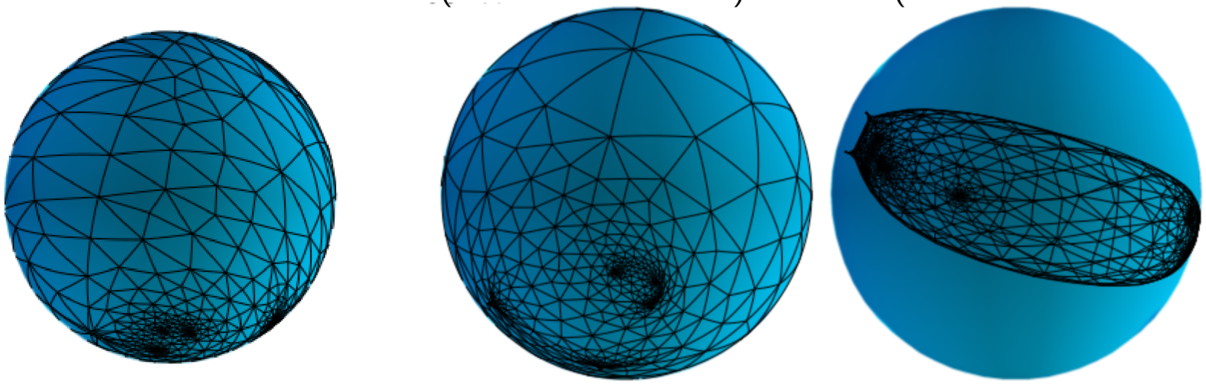
$$\mathbf{v}_i = \frac{\mathbf{u}}{||\mathbf{u}||}.$$



perform iterative relaxation

```
Projected Gauss-Seidel(x^0, λ, ε)
r = 0; // iteration counter
do {
   for(i = 1; i ≤ n; i++) {
      s = (1 − λ)x^r(v_i) + λ Σ_j w_ij x^r(v_j)
      x^{r+1}(v_i) = s / ||s||
   }
   r++;
} while (||x^r − x^{r−1}|| > ε))
```

$\mathbf{x}^0$ (initial layout) x^{50} (after 50 iterations) x^{1600} (after 1600 iterations)



collapse to degenerate solution

Extends graph drawing tools on the sphere: use non-euclidean spring embedders

Spherical drawings/parameterizations: related works

Spherical parameterizations: use planar parameterization + spherical projections (**not crossing-free**)

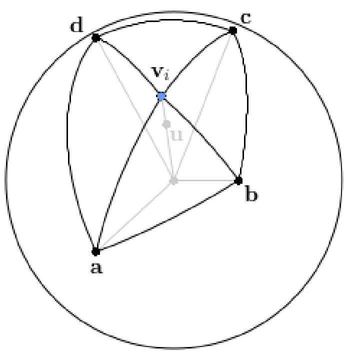
Non linear optimization: (try to) solve Tutte equations on the sphere

(Alexa, 2000)
(Gotsman Gu Sheffer, 2003)
(Sheffer Gotsman Dyn, 2004)
(Saba, Yavneh, Gotsman, Sheffer, 2005)
(Zayer, Rössl, Seidel 2006)
(Friedel, Schroder, Desbrun 2007)

(Aigerman, Lipman, 2015)
(Aigerman, Kowalsky, Lipman, 2017)

$$x_i^2 + y_i^2 + z_i^2 = 1 \qquad i = 1,..,n$$
$$\alpha_i x_i - L_W[i]x = 0 \qquad i = 1,..,n$$
$$\alpha_i y_i - L_W[i]y = 0 \qquad i = 1,..,n$$
$$\alpha_i z_i - L_W[i]z = 0 \qquad i = 1,..,n$$

$$\mathbf{v}_i = \frac{\mathbf{u}}{||\mathbf{u}||}.$$



perform iterative relaxation

Alexa($\mathbf{x}^0$, ε) **(Alexa, 2000)**

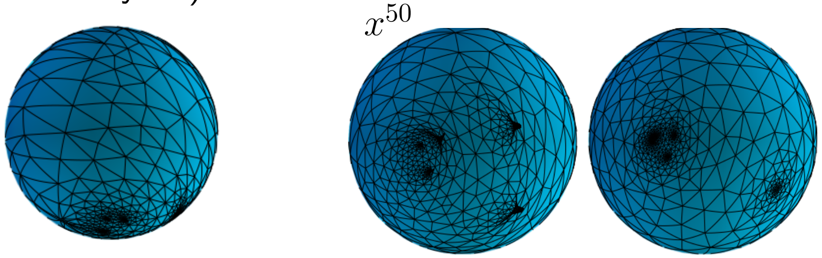
Idea: penalize long edges

$$c_i^r = [\max_{j \in N(i)} \{\|\mathbf{x}^r(v_i) - \mathbf{x}^r(v_j)\|\}]^{-1}$$

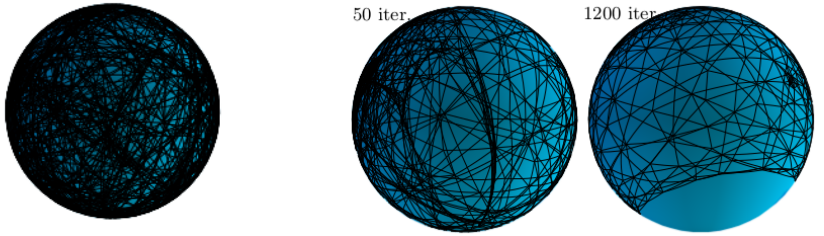
$$\mathbf{q}^r = \frac{c_i^r}{deg(v_i)} \sum_{j \in N(i)} (\mathbf{x}^r(v_i) - \mathbf{x}^r(v_j))\|\mathbf{x}^r(v_i) - \mathbf{x}^r(v_j)\|$$

$$\mathbf{x}^{r+1}(v_i) = \frac{\mathbf{x}^r(v_i) - \mathbf{q}_i^r}{\|\mathbf{x}^r(v_i) - \mathbf{q}_i^r\|}$$

$\mathbf{x}^0$ (initial layout)



$\mathbf{x}^0$ (initial random layout)



Extends graph drawing tools on the sphere: use non-euclidean spring embedders

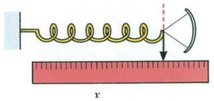
Spherical drawings/parameterizations: related works

Spherical parameterizations: use planar parameterization + spherical projections (**not crossing-free**)

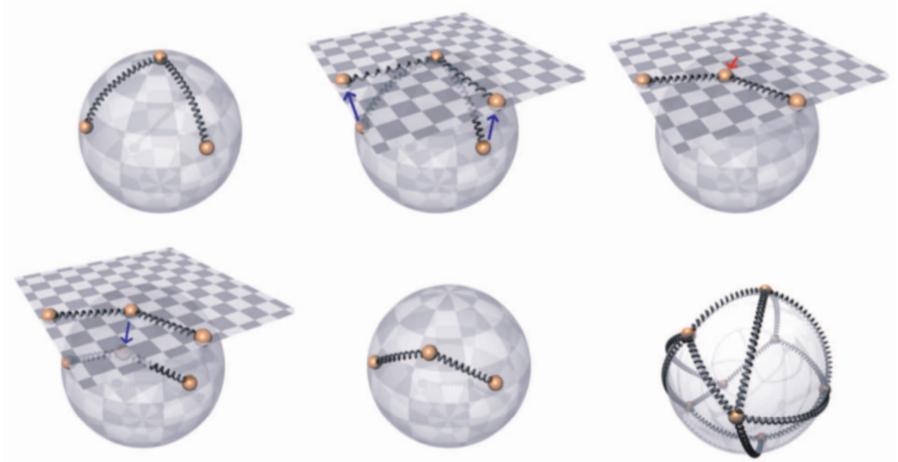
compute a **Steinitz representation** and project on the sphere
(Shapiro, Tal, 1998)

Non linear optimization: (try to) solve Tutte equations on the sphere
(Alexa, 2000)
(Gotsman Gu Sheffer, 2003)
(Sheffer Gotsman Dyn, 2004)
(Saba, Yavneh, Gotsman, Sheffer, 2005)
(Zayer, Rössl, Seidel 2006)
(Friedel, Schroder, Desbrun 2007)

(Aigerman, Lipman, 2015)
(Aigerman, Kowalsky, Lipman, 2017)



non-euclidean spring embedders

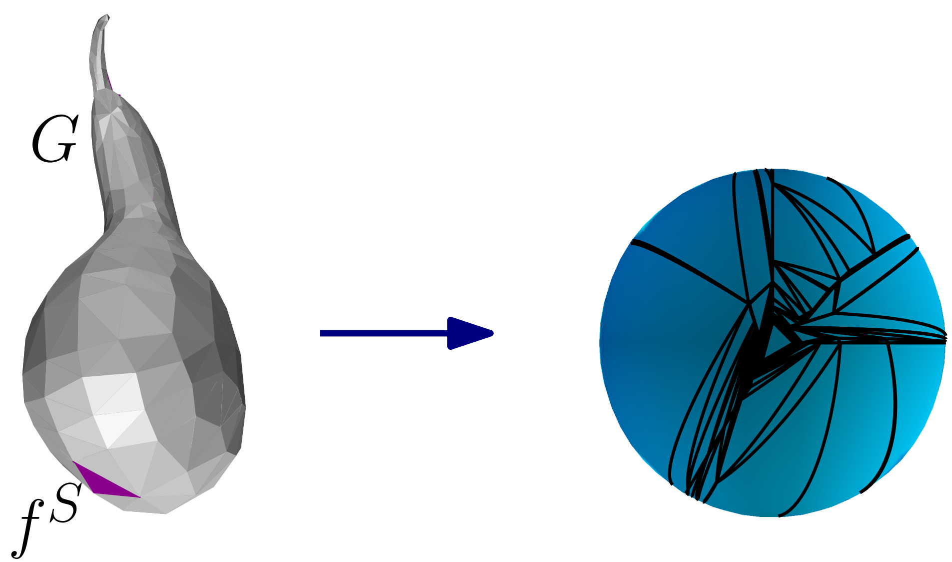


Extends graph drawing tools on the sphere:
use non-euclidean spring embedders

(Kobourov Wampler, 2005)

Fast and robust spherical drawing: our solution

Theorem
Let G be a planar triangulation with n vertices, having two non-adjacent faces f^S and f^N .
Then one can compute in $O(n)$ time a spherical drawing of G, where edges are drawn as (non-crossing) geodesic arcs of length at least $\Omega(\frac{1}{n})$.



our SFPP algorithm: spherical FPP

Our combinatorial solution

take 2 vertex disjoint faces

compute 3 vertex-disjoint paths (Menger theorem)

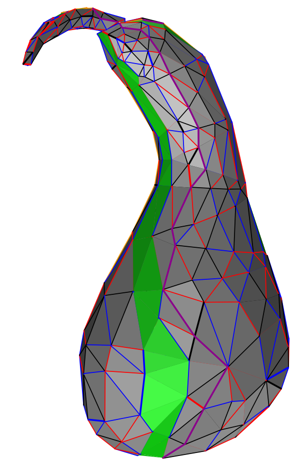
(you can use **Schnyder woods**)

and decompose G into 3 regions G_0, G_1 and G_2

compute 3 **rivers** $+$ compute the **canonical labeling** for each sub-region

(Duncan, Goodrich, Kobourov 2011)

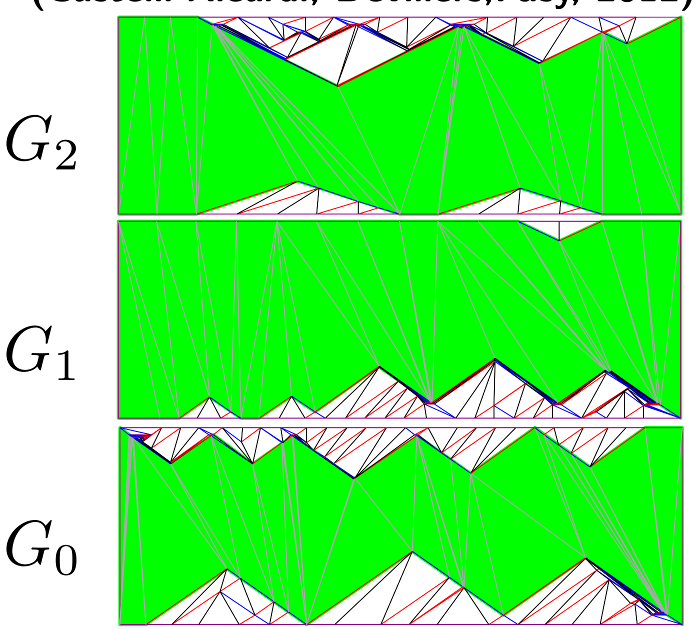
each region G_i is decomposed by the its river into 2 bottom and top sub-regions

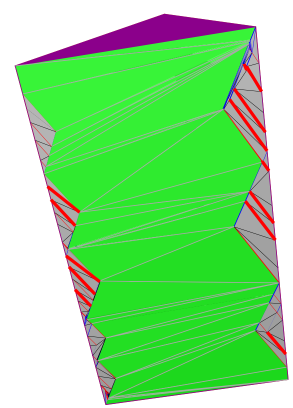


compute and align the 6 rectangular layouts

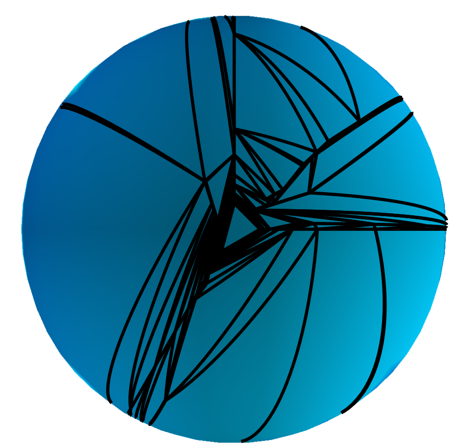
(Castelli Aleardi, Fusy, Kostrygin 2014)

(Castelli Aleardi, Devillers,Fusy, 2012)





glue together the rectangular drawings to obtain a prism



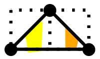
perform central projection

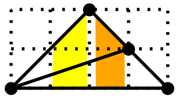
incremental shift algorithm (original FPP)

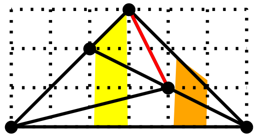
[de Fraysseix, Pollack, Pach'89]

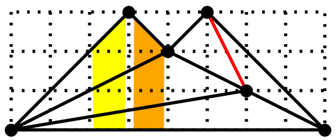
use the canonical ordering

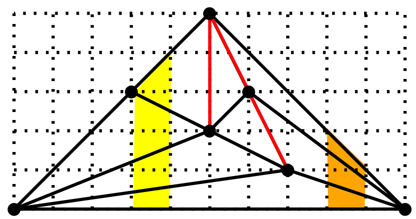
Grid size of G_k: $2k \times k$

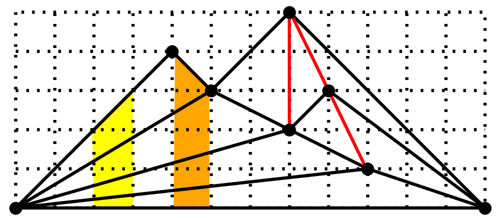
1. 

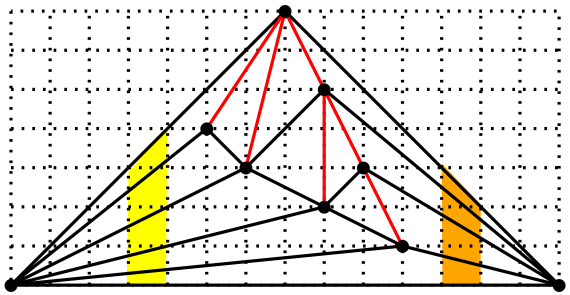
2. 

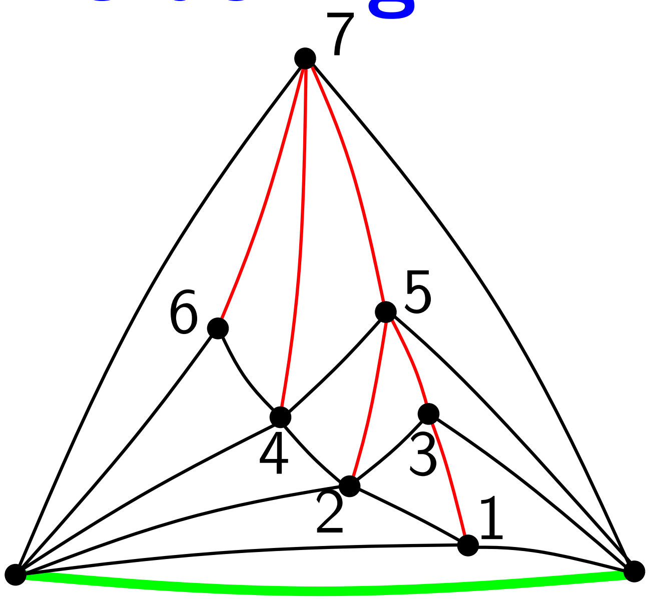
3. 

4. 

5. 

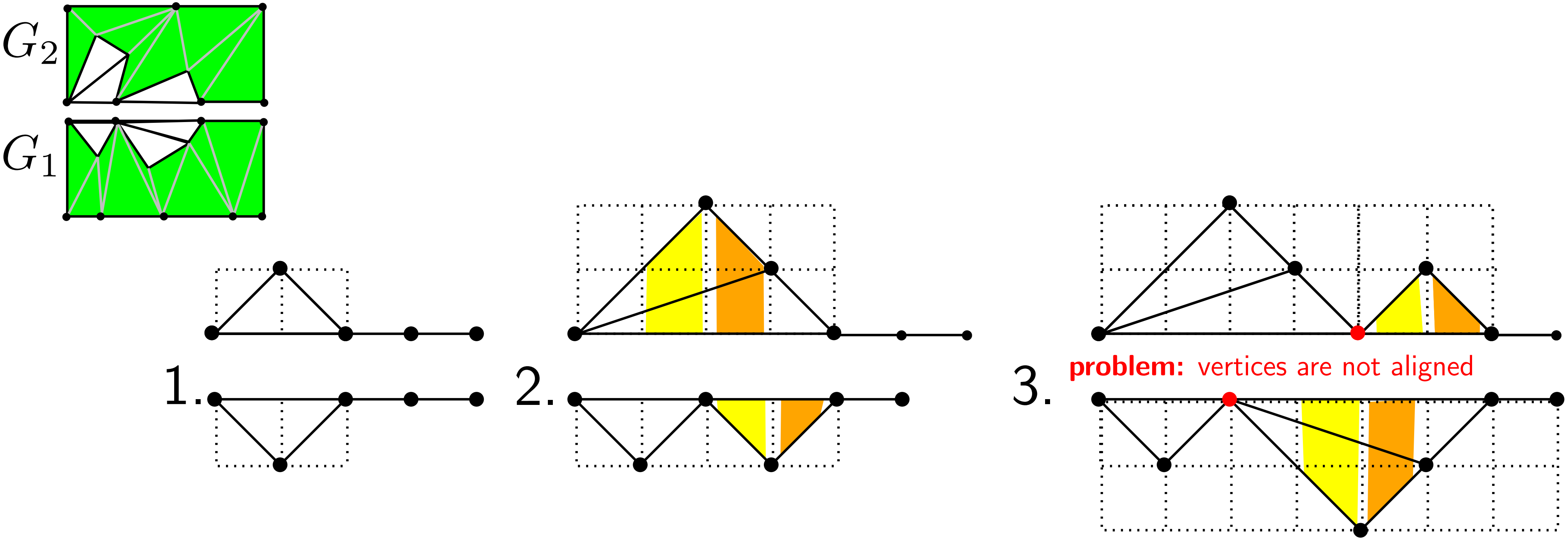
6. 

7. 



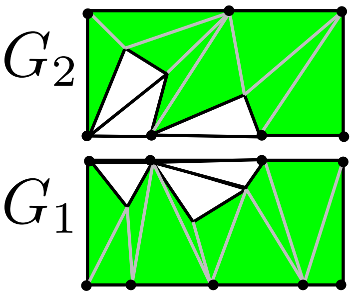
Drawing rectangular drawings with our adaptation of the FPP algorithm

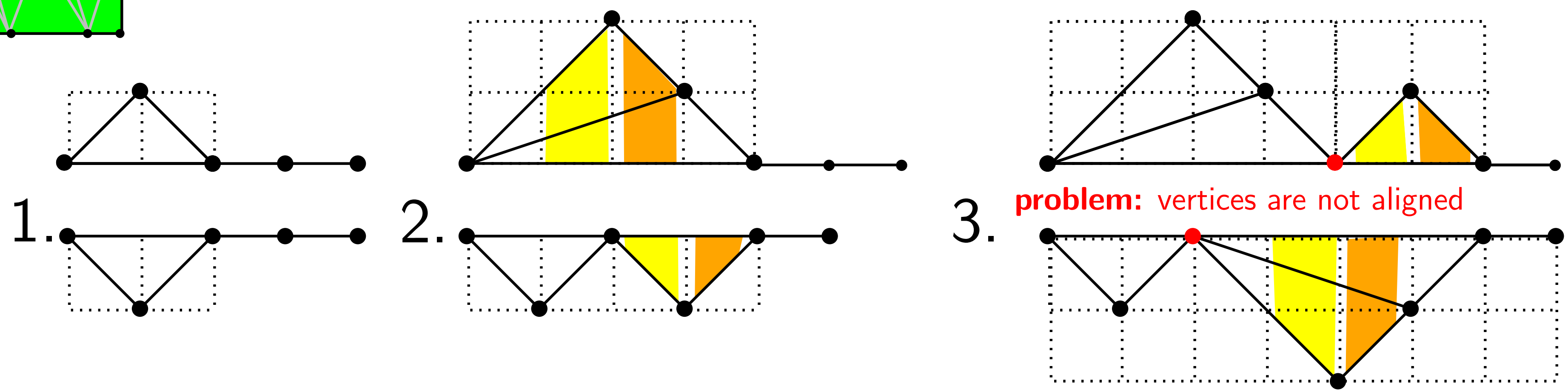
(Castelli Aleardi, Fusy, Kostrygin 2014)
(Castelli Aleardi, Devillers,Fusy, 2012)

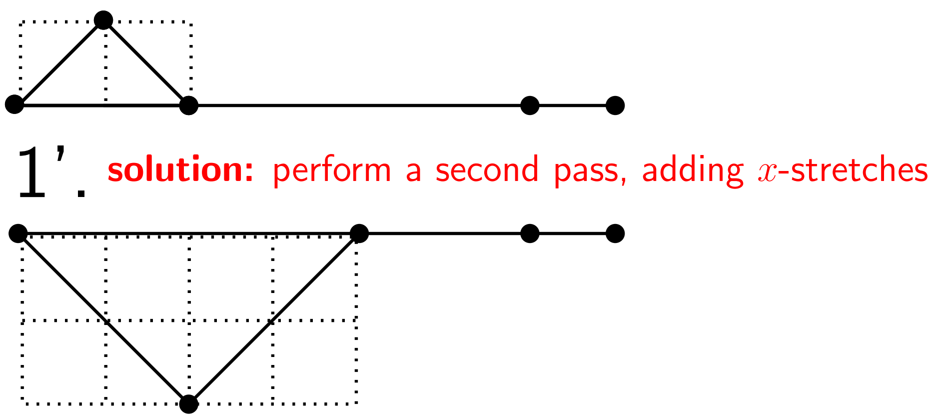


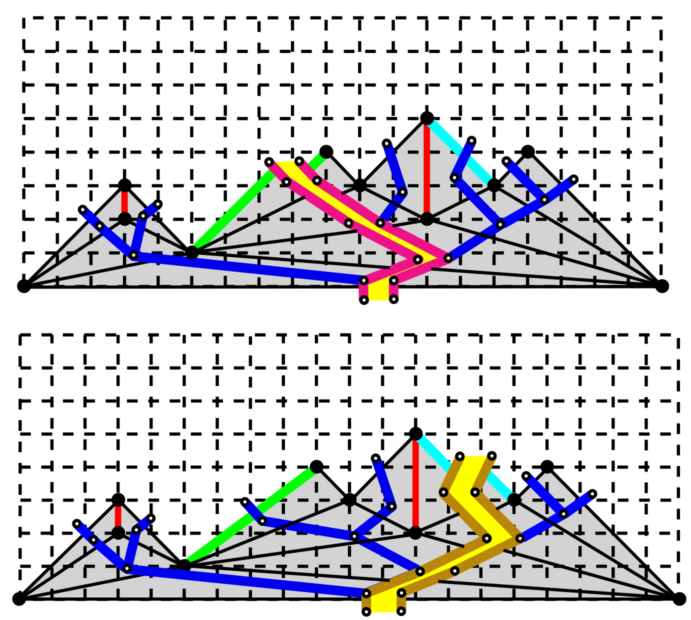
Drawing rectangular drawings with our adaptation of the FPP algorithm

(Castelli Aleardi, Fusy, Kostrygin 2014)
(Castelli Aleardi, Devillers,Fusy, 2012)

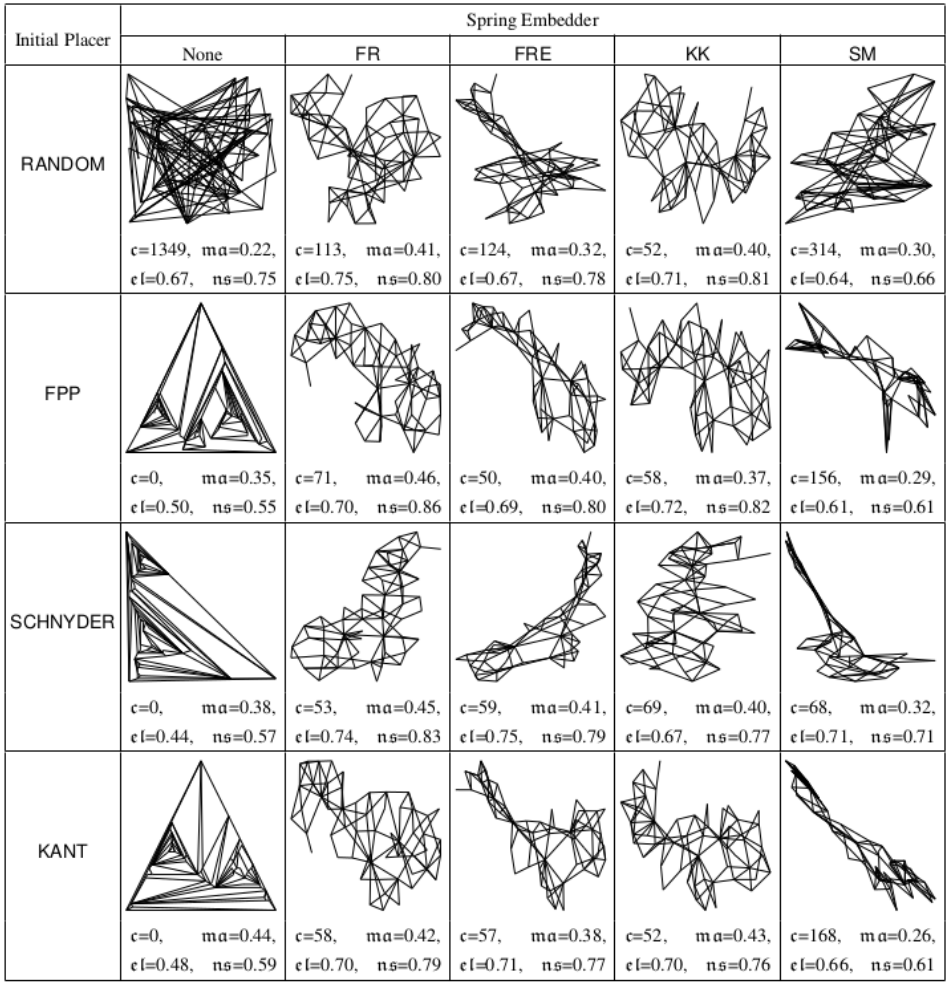








Experimental evaluation:
use our spherical drawing as initial placer for iterative schemes



Initial Placer	Spring Embedder: None	FR	FRE	KK	SM
RANDOM	c=1349, ma=0.22, el=0.67, ns=0.75	c=113, ma=0.41, el=0.75, ns=0.80	c=124, ma=0.32, el=0.67, ns=0.78	c=52, ma=0.40, el=0.71, ns=0.81	c=314, ma=0.30, el=0.64, ns=0.66
FPP	c=0, ma=0.35, el=0.50, ns=0.55	c=71, ma=0.46, el=0.70, ns=0.86	c=50, ma=0.40, el=0.69, ns=0.80	c=58, ma=0.37, el=0.72, ns=0.82	c=156, ma=0.29, el=0.61, ns=0.61
SCHNYDER	c=0, ma=0.38, el=0.44, ns=0.57	c=53, ma=0.45, el=0.74, ns=0.83	c=59, ma=0.41, el=0.75, ns=0.79	c=69, ma=0.40, el=0.67, ns=0.77	c=68, ma=0.32, el=0.71, ns=0.71
KANT	c=0, ma=0.44, el=0.48, ns=0.59	c=58, ma=0.42, el=0.70, ns=0.79	c=57, ma=0.38, el=0.71, ns=0.77	c=52, ma=0.43, el=0.70, ns=0.76	c=168, ma=0.26, el=0.66, ns=0.61

Table 5: Layouts of a representative 55-node graph from the FUSY library.

Planar preprocessing for spring embedders **(Fowler and Kobourov, 2012)**

Edge length statistic (Fowler and Kobourov, 2012)

(average percent deviation of edge length)

high values indicates more uniform edge length

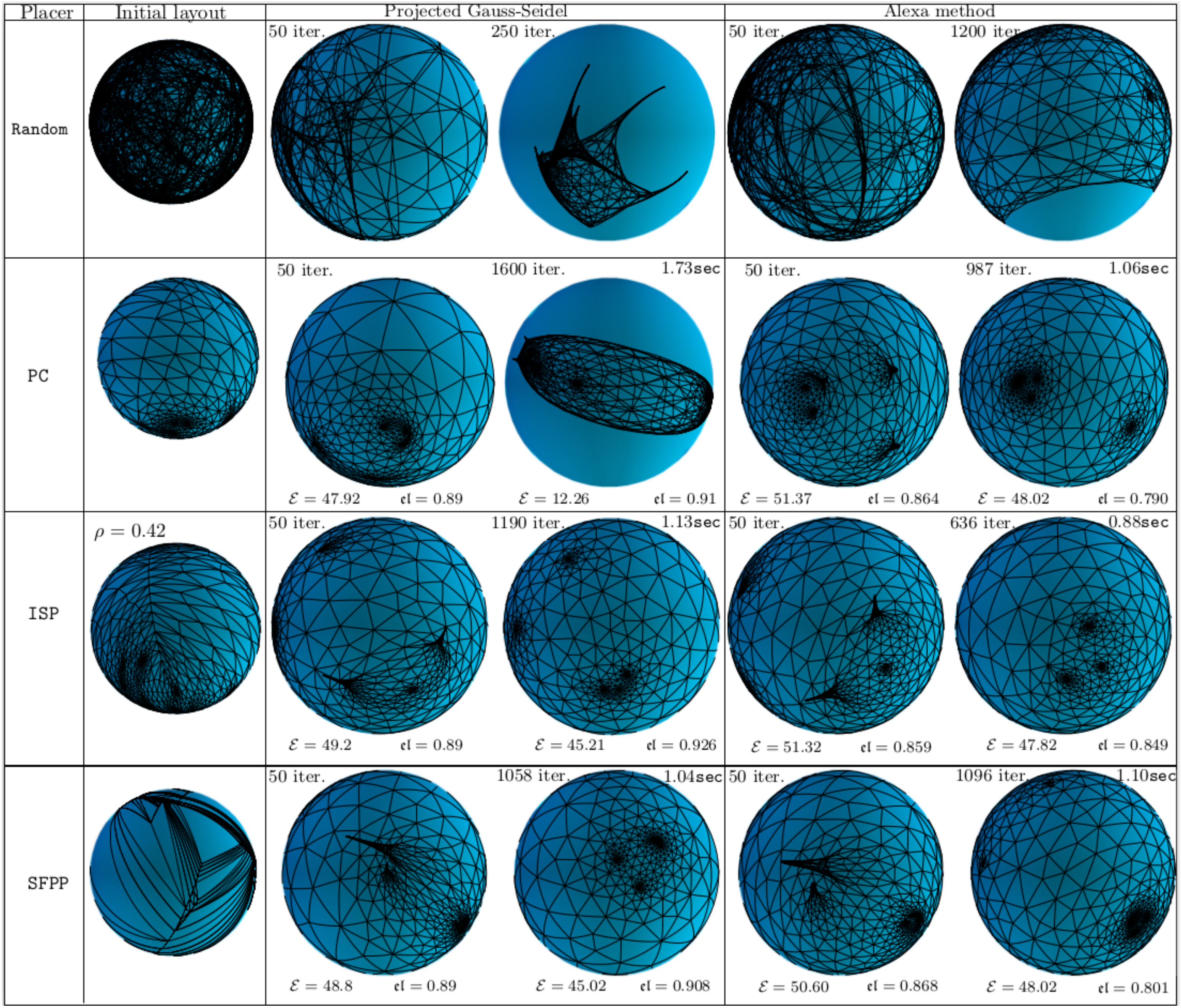
$l(e) :=$ edge length of e

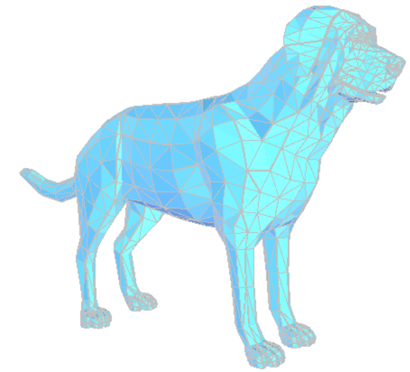
$$\mathfrak{el} := 1 - \left(\frac{1}{|E|}\sum_{e\in E}\frac{|l(e)-l_{avg}|}{\max(l_{avg}, l_{max}-l_{avg})}\right)$$

Comparison of initial placers

Comparison of convergence rates and aesthetic statistics

Dataset: 3D meshes from `aim@shape` repository

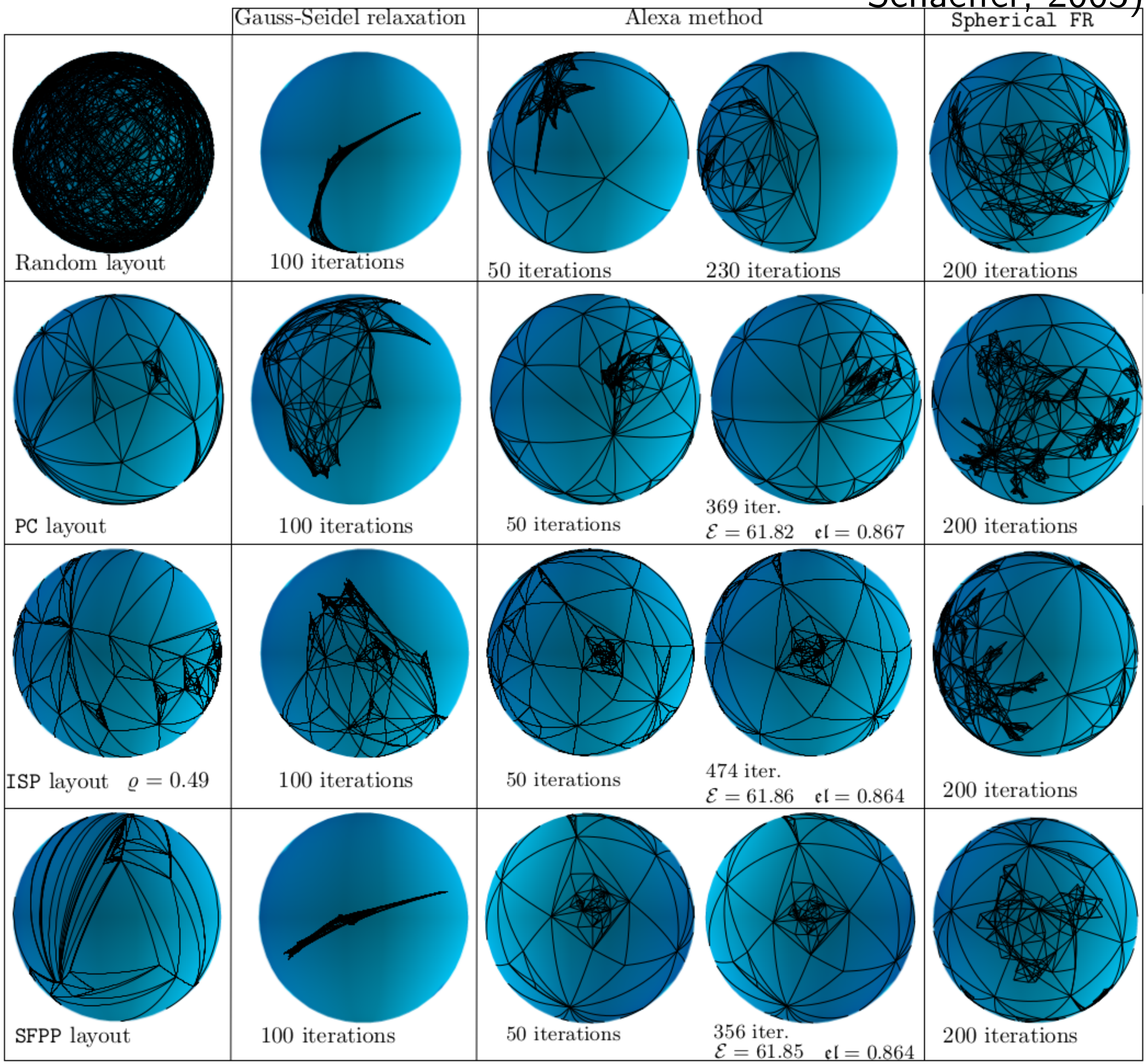


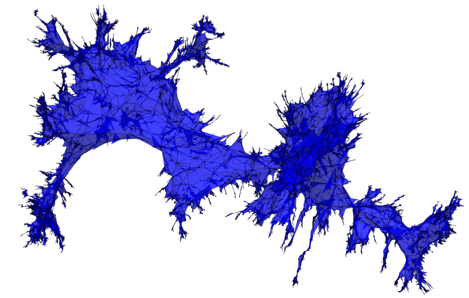


Comparison of initial placers

Comparison of convergence rates and aesthetic statistics

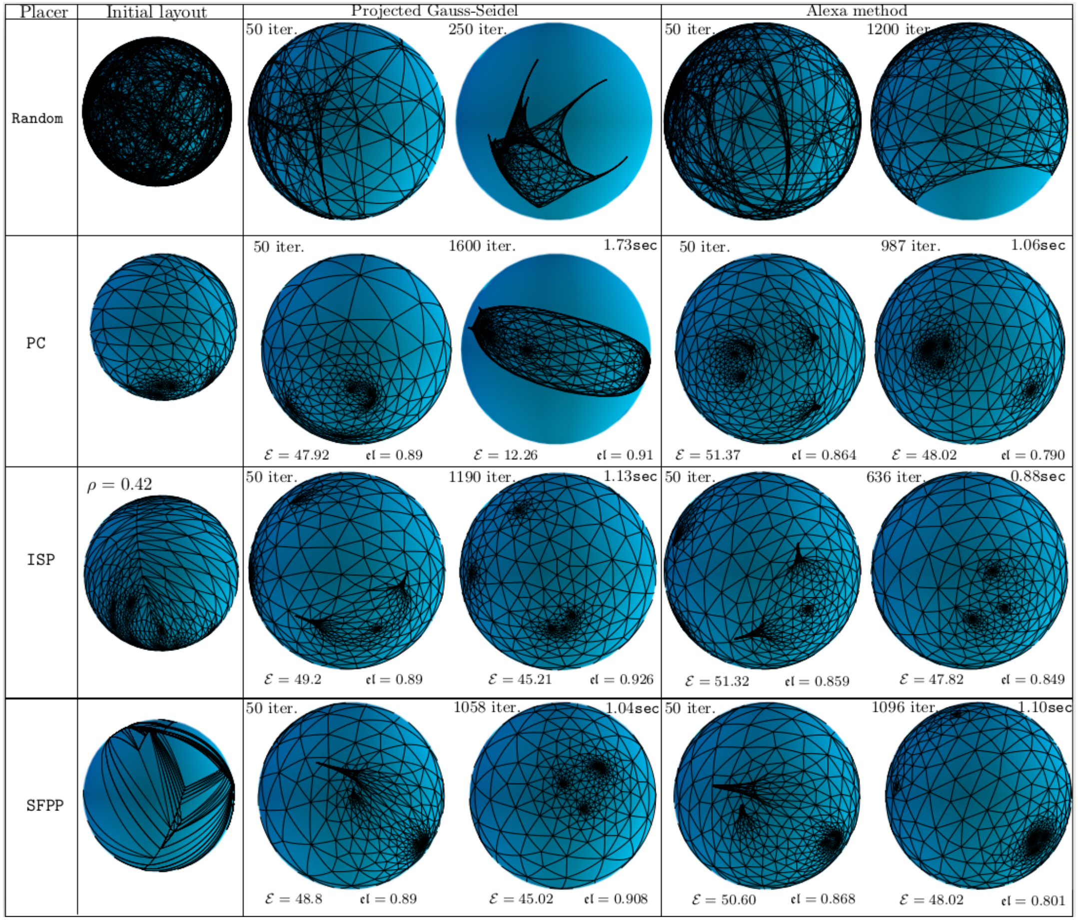
Dataset: random planar triangulations (uniform random sampler by Poulalhon and Schaeffer, 2003)





Comparison of initial placers

Comparison of convergence rates and aesthetic statistics



(geodesic) edge length statistic

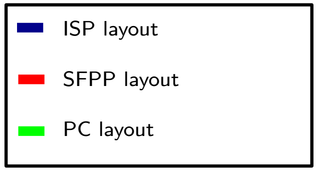
$$\mathfrak{el} := 1 - \left(\frac{1}{|E|} \sum_{e \in E} \frac{|l_g(e) - l_{avg}|}{\max(l_{avg}, l_{max} - l_{avg})}\right)$$

Area statistic

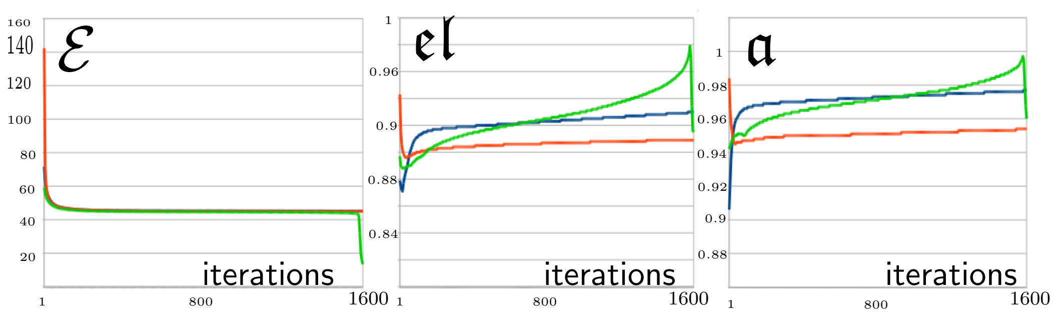
$$\mathfrak{al} := 1 - \left(\frac{1}{|F|} \sum_{t \in F} \frac{|a(f) - a_{avg}|}{\max(a_{avg}, a_{max} - a_{avg})}\right)$$

Spring energy

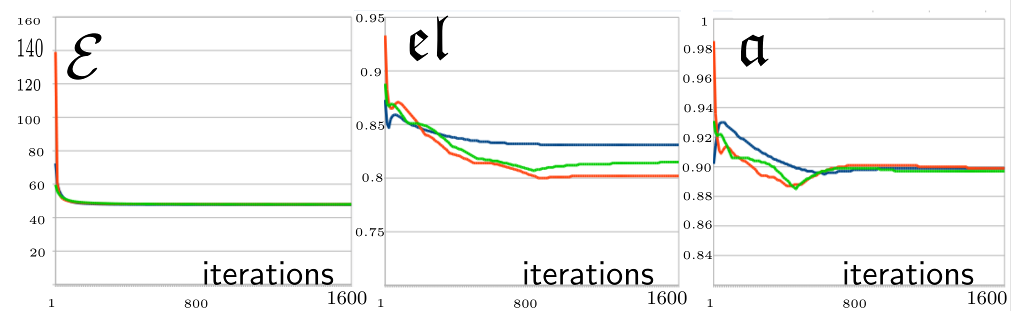
$$\mathcal{E} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} \|\mathbf{x}(v_i) - \mathbf{x}(v_j)\|^2$$



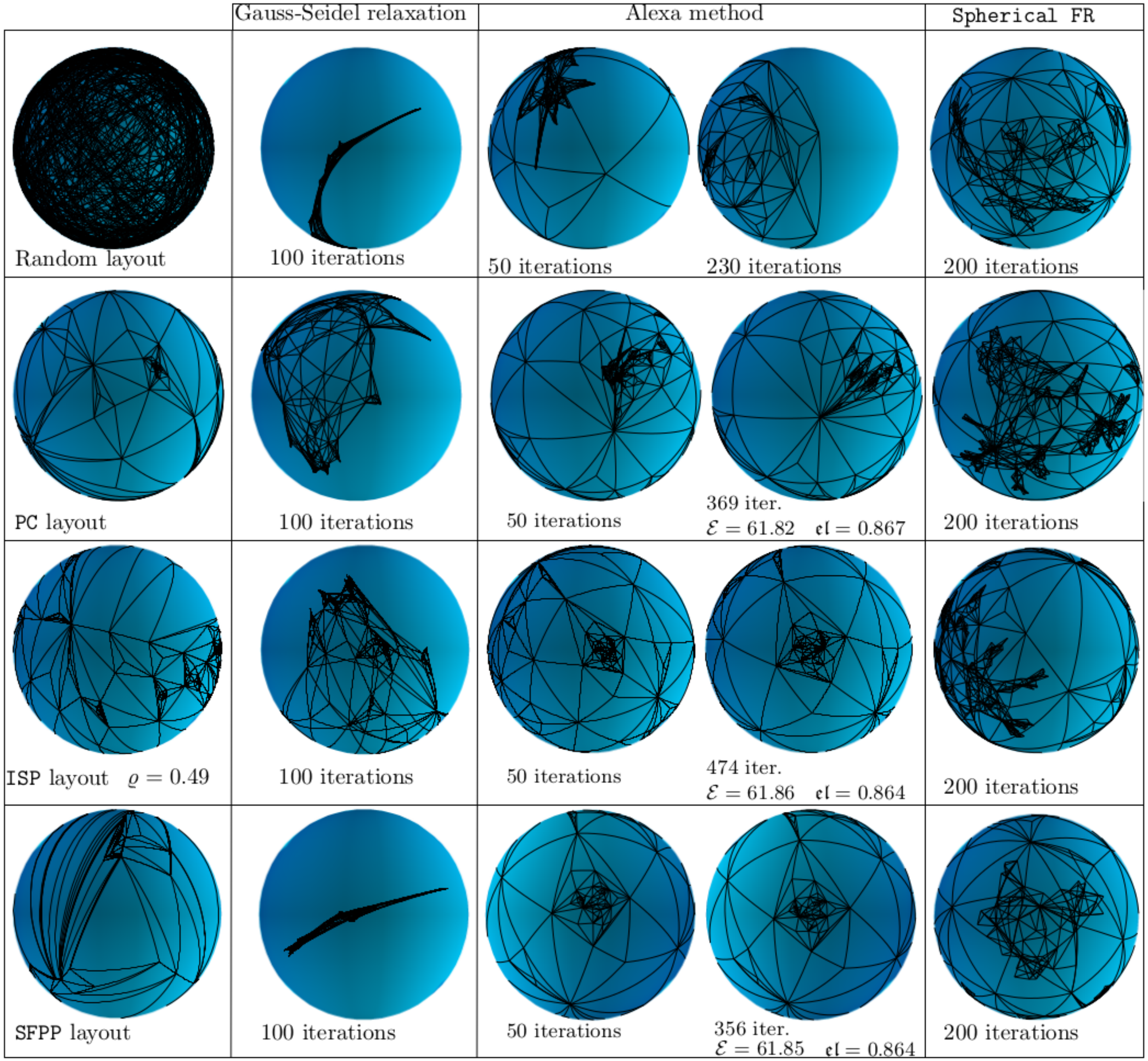
projected Gauss-Seidel

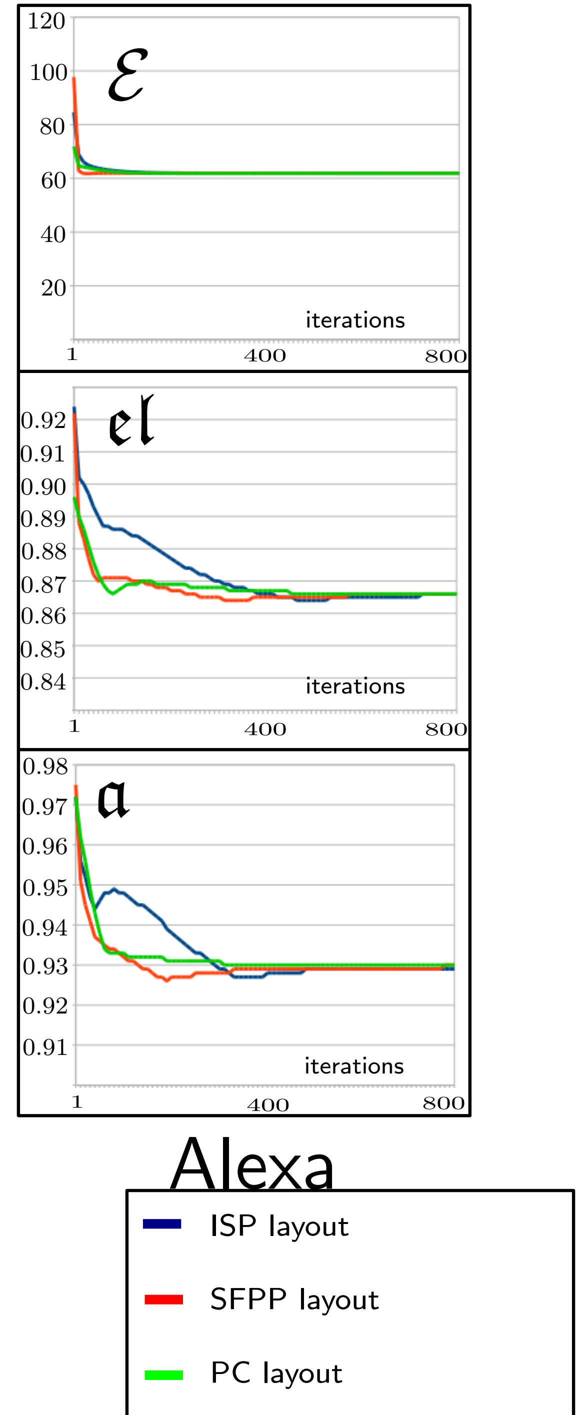


Alexa



Comparison of convergence rates and aesthetic statistics





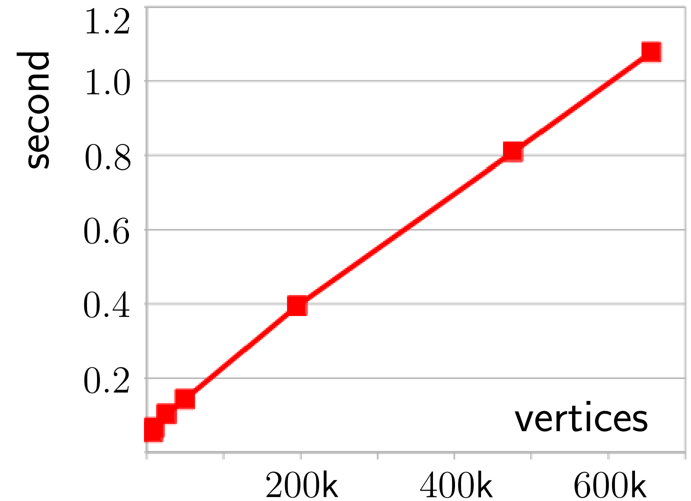
Comparison of initial placers

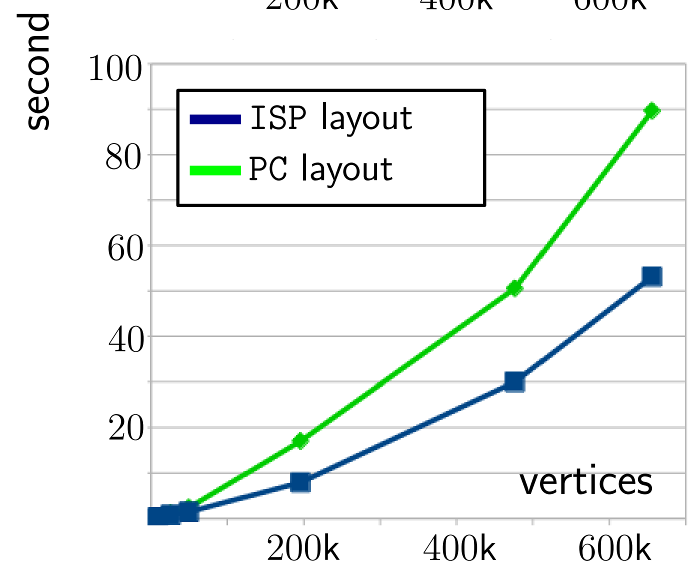
Comparison of timing performances

solve sparse linear systems with the conjugate gradient solver of `MTJ` (Java) library (numeric precision 10^{-6})

mesh	vertices	faces	our **SFPP** algorithm: preprocessing: rivers comput.	canonical labeling	Layout computation: shift algorithm	prism projection	Planar parameterizations: PC linear solver	ISP linear solver
Egea	8268	16K	0.015	0.017	0.005	0.017	0.24	0.16
Gargoyle	10002	20K	0.016	0.018	0.007	0.025	0.26	0.22
Bunny	26002	52K	0.017	0.031	0.019	0.036	1.14	0.75
Iphigenia	49922	99K	0.023	0.049	0.025	0.046	2.38	1.44
Camille's hand	195557	391K	0.076	0.121	0.073	0.125	17.02	7.92
Eros	476596	950K	0.162	0.260	0.132	0.255	50.54	29.99
Chinese dragon	655980	1.3M	0.174	0.314	0.157	0.433	89.64	53.12

Dataset: 3D meshes from aim@shape repository





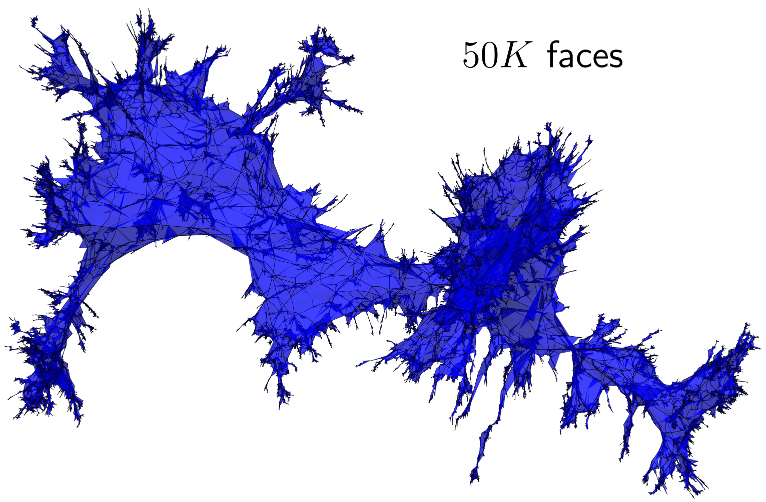
Spherical drawing of the **Chinese dragon** ($655K$ vert.)

Aigerman Lipman (2015): 19 seconds (solving linear systems) (Matlab, 3.50GHz Intel i7 CPU)

our **SFPP algorithm**: 1.07 seconds (total cost) (Java, 2.66GHz Intel i7 CPU)

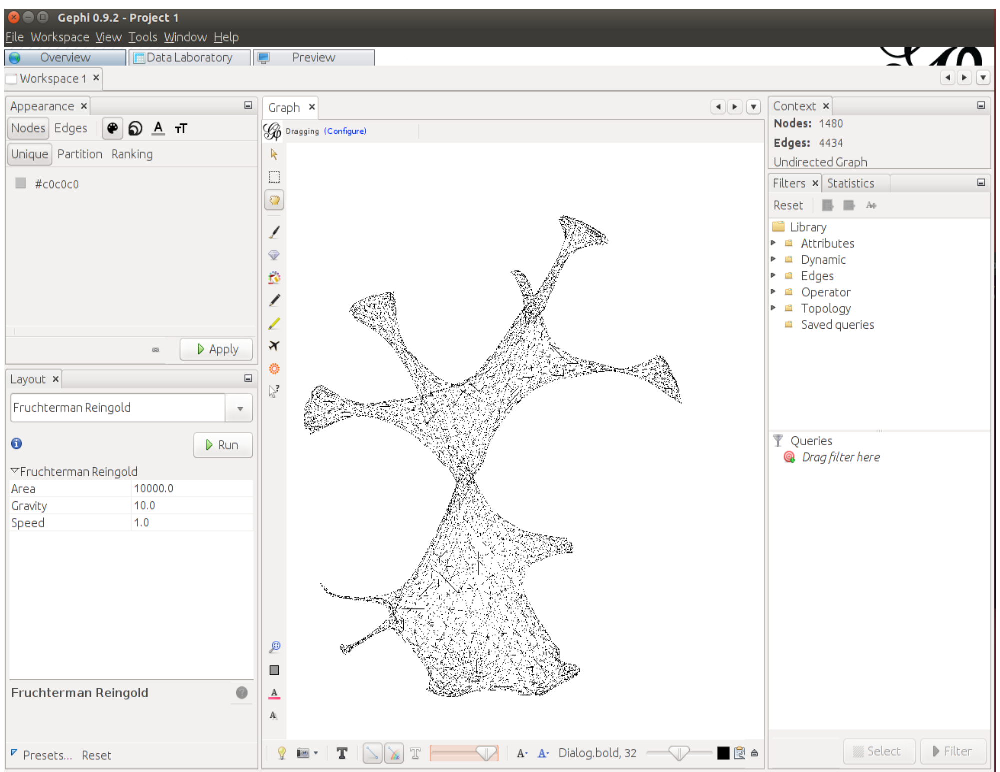
Application

spherical preprocessing for eucliden 3D spring embedders



$50K$ faces

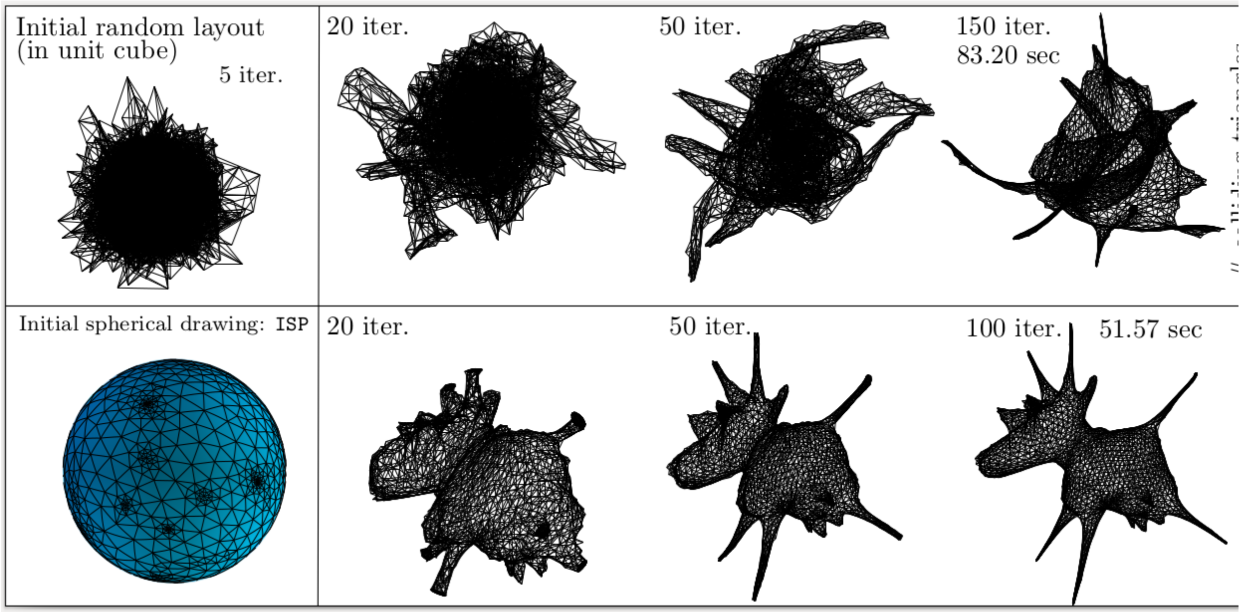
3D layout of a random planar map
by J. Bettinelli
(force-directed layout of`Mathematica`)

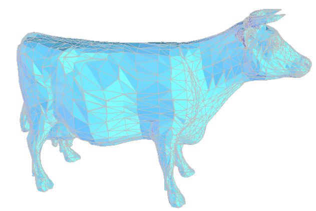


Yifan Hu layout of the dog graph (`Gephi`)

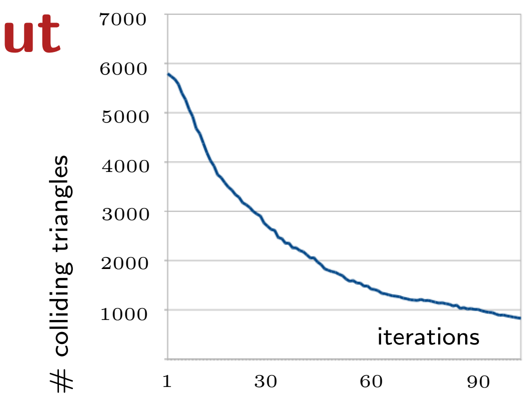
Spherical preprocessing for eulidean spring embedders

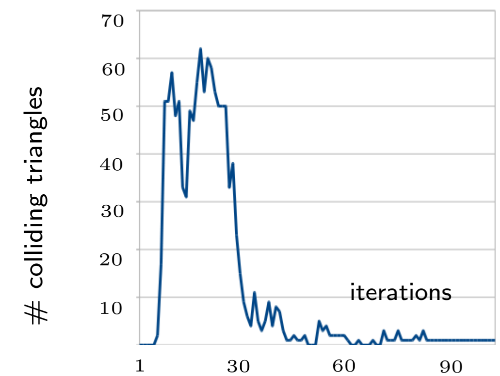
Use spherical drawings as initial layouts for 3D spring embedders: this allows us to better untangle the layout



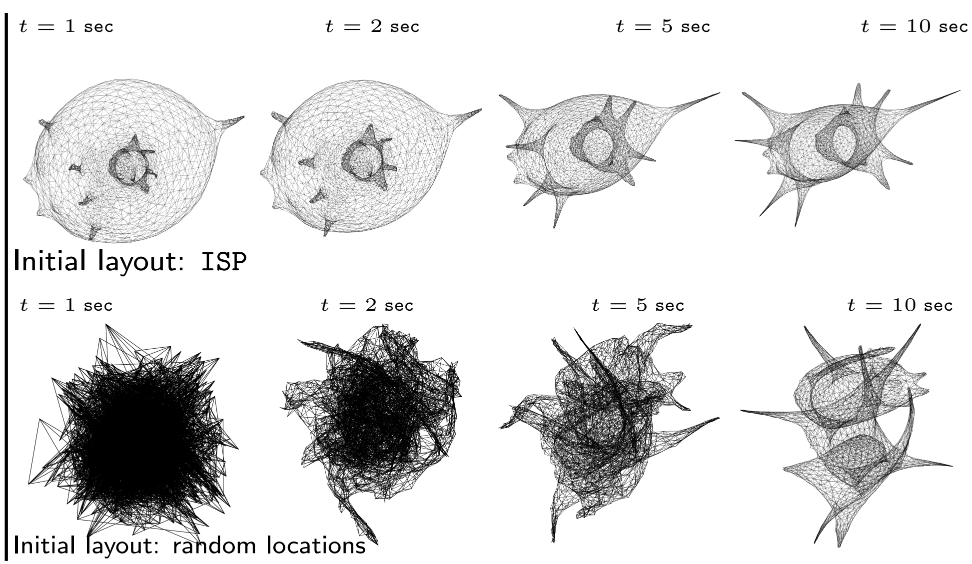


count triangle collisions





Our Java implementation of the FR91 spring embedder (exact computation of repulsive forces)

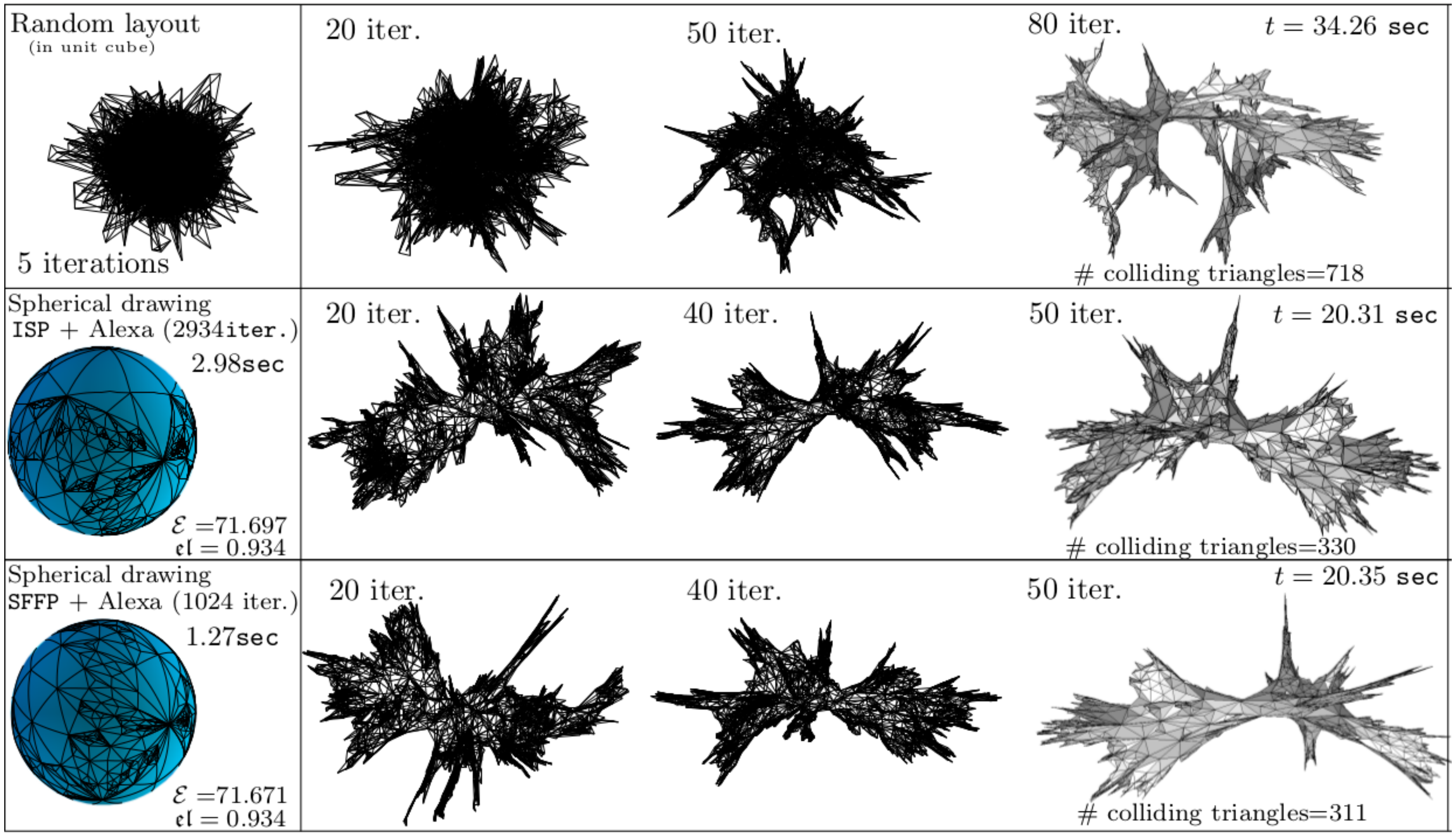


Gephi implementation of the Yifan Hu layout (fast approximate computation of repulsive forces)

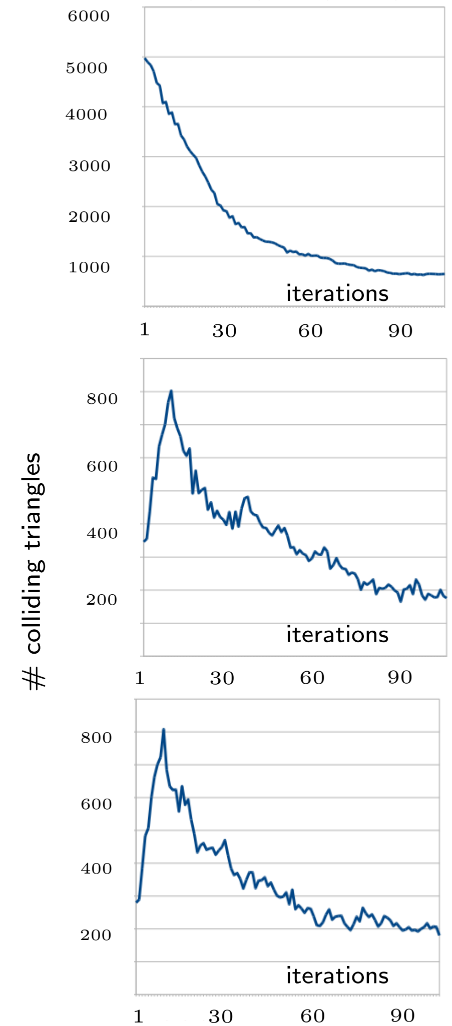
Spherical preprocessing for eulidean spring embedders

Use spherical drawings as initial layouts for 3D spring embedders: this allows us to better untangle the layout

`random planar triangulation`
with $5K$ triangles (generated with an uniform random sampler)



count triangle collisions



Layouts obtained with our Java implementation of the `FR91` spring embedder
(exact computation of repulsive forces)

Thank you for your attention