# Drawing Big Graphs using Spectral Sparsification*

Peter Eades
Quan Nguyen
Seok-Hee Hong

University of Sydney

*Supported by
the Australian Research Council,
Tom Sawyer Software, and
NewtonGreen Technologies

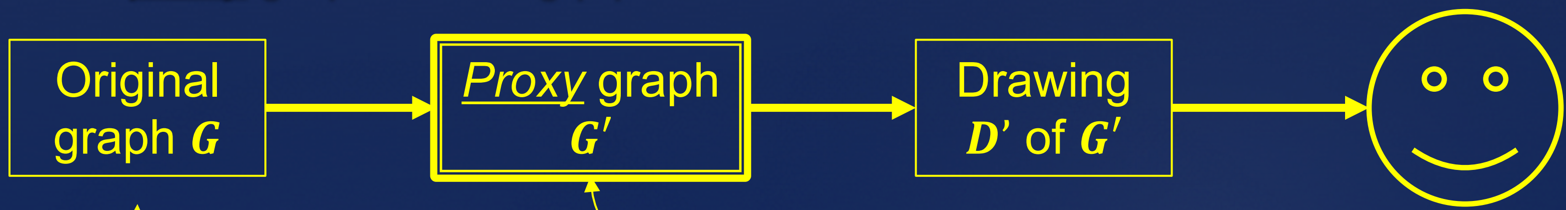
## Drawing Big Graphs using Spectral Sparsification

1. *Background*
    a) *Big graphs*
    b) *Sparsification*
    c) *Spectra of graphs*
    d) *Spectral sparsification*
2. Spectral Sparsification for Graph Drawing
    a) Deterministic Spectral Sparsification (***DSS***) algorithm
    b) Stochastic Spectral Sparsification (***SSS***) algorithm
3. *Testing **DSS** and **SSS***
    a) *Two typical examples*
    b) *An atypical example*
    c) *Experiments*
4. *Conclusion*

*The graph drawing pipeline*

## *Proxy graphs and sparsification*

*The proxy graph drawing pipeline*




*Big graph*

*Smaller graph that represents the original graph*

For example the proxy graph $G'$ could be:

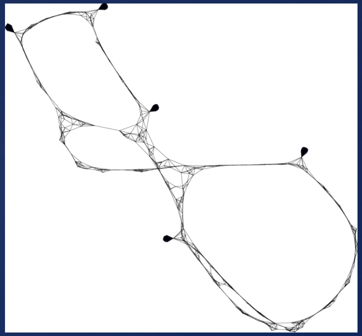
- (some kind of) spanning tree of $G$
- (some kind of) sample of $G$
- (some kind of) *sparsification* of $G$

If $G = (V, E)$ and $G' = (V, E')$ where $E' \subset E$, then $G'$ is a *sparsification* of $G$

## *Proxy graphs and sparsification*

Original graph $G$ → Drawing $D$ of $G$

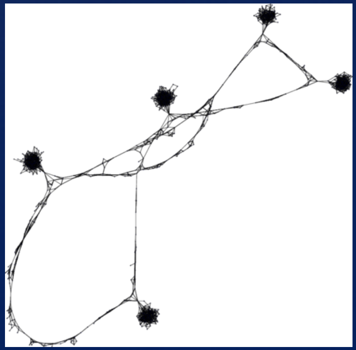
~1000 vertices
~22000 edges



Original graph $G$ → Sparsification $G'$ → Drawing $D'$ of $G'$

~1000 vertices
~22000 edges

~1000 vertices
~3000 edges



## *Graph spectra*[1]

Suppose that $G$ is an $n$-vertex graph.

Define:

- $A$ = adjacency matrix of $G$ : $A_{uv} = \begin{cases} 1 \text{ if } (u,v) \in E \\ 0 \text{ otherwise} \end{cases}$
- $D$ = degree matrix of $G$ : $D_{uv} = \begin{cases} deg(u) \text{ if } u = v \\ 0 \text{ otherwise} \end{cases}$
- *Laplacian* $L$ of $G$ : $L = D - A$
- *Eigenvalues/eigenvectors* of $G$ are the eigenvalues/eigenvectors of the Laplacian $L$
- *Spectrum* of $G$ is the $1 \times n$ vector $[\lambda_1, \lambda_2, \ldots \lambda_n]$ of eigenvalues of the Laplacian, in nondecreasing order.

1. Beware: much of the terminology in spectral graph theory is not standardised.

# *Graph spectra*

The main significance of the spectrum of a graph (informally) :

- If two graphs have the same spectrum, then they are structurally similar.

  - <u>*Connectivity*</u> = the number of zero eigenvalues. The first non-zero eigenvalue is a measure of connectivity, called *algebraic connectivity*.

  - <u>*Clusters*</u>: Spectral clustering solves a relaxation of the *ratio cut problem*.

  - <u>*Stress*</u>: The eigenvalues measure the minimum of a kind of stress in the graph.

  - <u>*Commute distance*</u>: Eigenvalues are related to random walks in the graph, and thus to *commute distances*.

## Courant - Fischer theorem

$$\lambda_i = \min_{x \in X_i} \frac{x^T L x}{x^T x} = \min_{x \in X_i} \left( \frac{1}{x^T x} \sum_{(u,v) \in E} (x_u - x_v)^2 \right)$$

where $X_i$ is the set of vectors orthogonal to the first $i - 1$ eigenvectors.

## *Spectral approximation*

Definition [Spielman -Teng]

Say $G$ has Laplacian $L$ and $G'$ has Laplacian $L'$.

If there is an $\epsilon > 0$ such that for every $x \in R^n$,

$$(1-\epsilon)\frac{x^T L x}{x^T x} < \frac{x^T L' x}{x^T x} < (1+\epsilon)\frac{x^T L x}{x^T x}$$

then $G'$ is an *$\epsilon$-spectral approximation* of $G$ .

From the Courant-Fischer Theorem $\left(\lambda_i = \min_{x \in X_i} \frac{x^T L x}{x^T x}\right)$:

If $G'$ is an $\epsilon$-spectral approximation of $G$, then

- the eigenvalues and eigenvectors of $G'$ are close to those of $G$.
- (informally) $G'$ has a *similar structure* to $G$.
- (informally) $G'$ is a *good proxy* for $G$.

## *Spectral sparsification*

Definition [Spielman-Teng]

If $\boldsymbol{G'} = (\boldsymbol{V}, \boldsymbol{E'})$ is an $\boldsymbol{\epsilon}$-spectral approximation of $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ with $E' \subset E$, then $\boldsymbol{G'}$ is an $\boldsymbol{\epsilon}$-*spectral sparsification* of $\boldsymbol{G}$.

Spectral sparsification theorems [Spielman and others, 2000+]

Spectral sparsifications exist, with

- $\epsilon$ small
- $|E'|$ much smaller than $|E|$

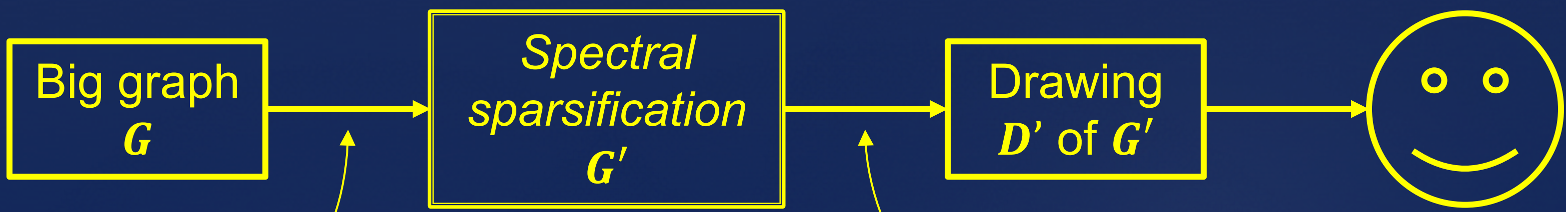
and they can be computed efficiently.

A specific theorem [Spielman-Srivastava, 2009]

Suppose that $\boldsymbol{G}$ is an $\boldsymbol{n}$-vertex graph, and $\frac{\mathbf{1}}{\sqrt{\boldsymbol{n}}} < \boldsymbol{\epsilon} < \mathbf{1}$.

Then with probability at least $\frac{1}{2}$, there is an $\boldsymbol{\epsilon}$-spectral approximation $\boldsymbol{G'}$ of $\boldsymbol{G}$ with $\boldsymbol{O}\left(\frac{\boldsymbol{n}\log \boldsymbol{n}}{\boldsymbol{\epsilon}^2}\right)$ edges.

# *Big Graph Drawing by Spectral Sparsification*




Step 1: sparsification algorithms $G \rightarrow G'$

- Derived from proofs of sparsification theorems

Step 2: layout algorithms $G' \rightarrow D'$

- Any layout algorithm is usable

*This paper*: is this a good way to draw big graphs?

## *Sparsification algorithms*

Two concepts for a sparsification $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}) \rightarrow \boldsymbol{G}' = (\boldsymbol{V}, \boldsymbol{E}')$:

*Relative density* $\boldsymbol{d} = \frac{|\boldsymbol{E}'|}{|\boldsymbol{E}|}$

*Effective resistance* $\boldsymbol{r}(\boldsymbol{e})$ of an edge $\boldsymbol{e}$

- Regard the graph as an electrical network where each edge is a 1-$\boldsymbol{\Omega}$ resistor.
- Voltage drop over an edge is the *effective resistance* $\boldsymbol{r}(\boldsymbol{e})$ of an edge $\boldsymbol{e}$ when a current is applied across $\boldsymbol{e}$.
- $\boldsymbol{r}(\boldsymbol{e})$ can be computed simply from the *Moore-Penrose inverse* (aka pseudo-inverse) of the Laplacian.

*Spectral sparsification algorithms*

Two specific spectral sparsification algorithms

*Input*: graph $\boldsymbol{G}$, relative density $\boldsymbol{d}$

*Output*: sparsification $\boldsymbol{G'}$ of $\boldsymbol{G}$ with relative density $\boldsymbol{d}$

***DSS***

(deterministic spectral sparsification)

Choose $\boldsymbol{E'}$ to be the $\boldsymbol{m'}$ edges with highest effective resistance, with $\boldsymbol{m'} = |\boldsymbol{E'}| = \lceil \boldsymbol{d}|\boldsymbol{E}| \rceil$.

***SSS***

(stochastic spectral sparsification)

*Repeat*

- Choose an edge $\boldsymbol{e}$ uniformly at random.
- Accept/reject $\boldsymbol{e}$ with probability proportional to its effective resistance.

*Until* $\boldsymbol{G'}$ has relative density $\boldsymbol{d}$

*Non-spectral sparsification algorithm*

Baseline sparsification algorithm, for comparison

***RES***

(random edge selection)

*Repeat*

- Choose an edge $e$ uniformly at random.

*Until* $G'$ has relative density $d$

The *absolute simplest* sparsification approach

Our hypothesis

- Both ***DSS*** and ***SSS*** are better than ***RES***

Our result

- On the whole,  the hypothesis holds true.

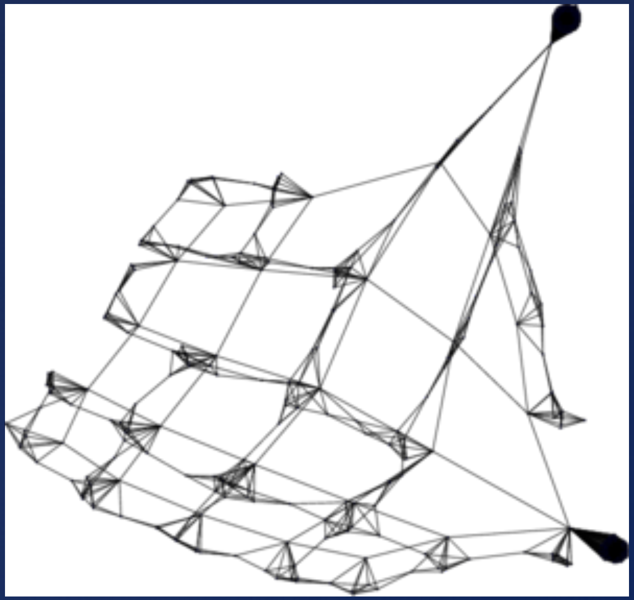
Typical example 1

`GridWithBlobs`

- $n = 733$
- $m = 62509$

- Most of the vertices and edges are in two blobs, but the global structure consists of a grid.

***FM³*** layout for the whole graph

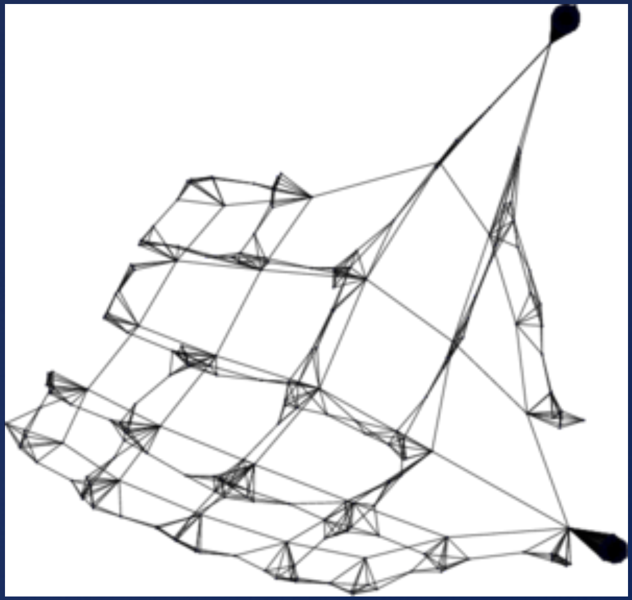


Typical example 1

GridWithBlobs

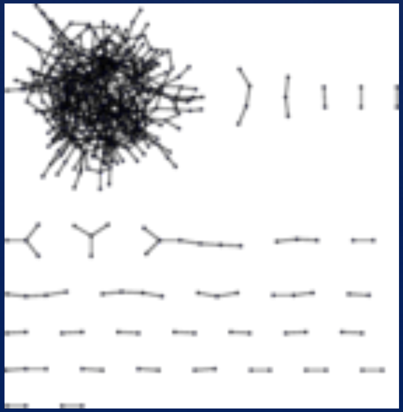
- $n = 733$
- $m = 62509$

***FM³***
layout for
the whole
graph

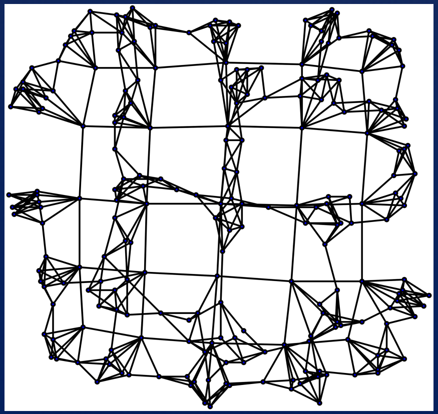


Sparsifications with relative density $d = 1\%$:

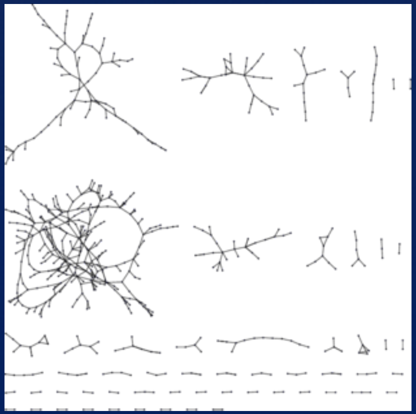
***RES***



***DSS***



***SSS***

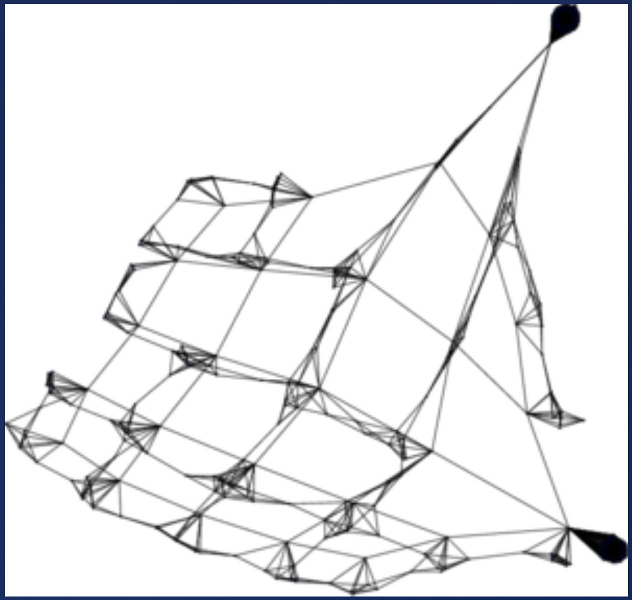


*Typical example 1*

GridWithBlobs

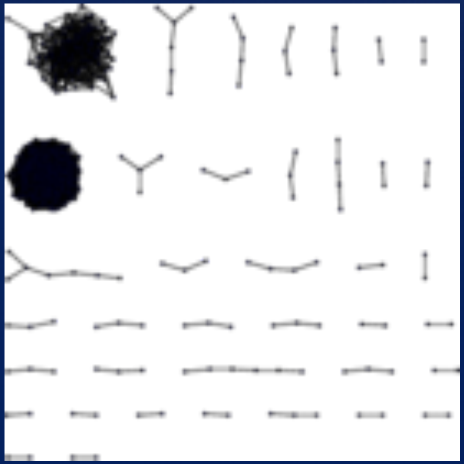
- $n = 733$
- $m = 62509$

***FM³*** **layout for the whole graph**

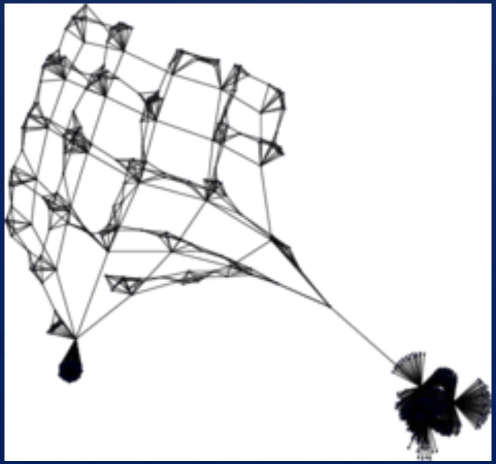


<u>*Sparsifications with relative density*</u> $\mathbf{d} = 10\%$:

***RES***



***DSS***



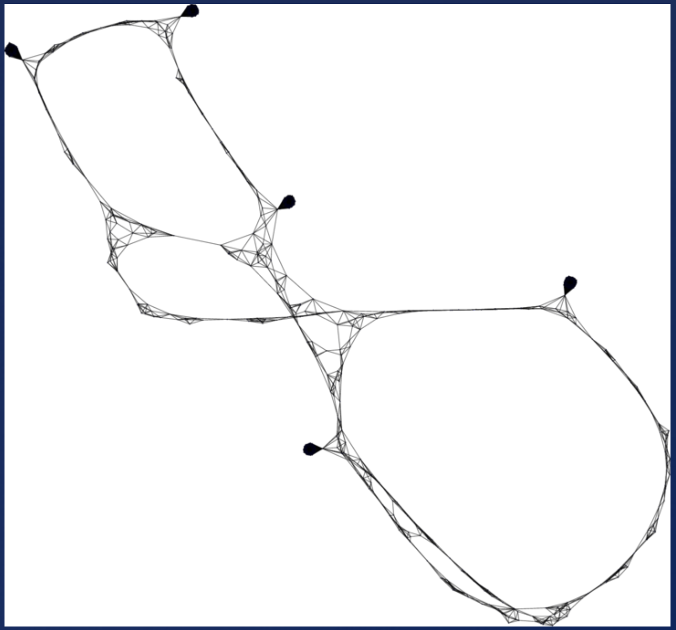
***SSS***

*Typical example 2*

`CyclesWithBlobs`

- $n = 1031$
- $m = 22638$

- Most of the vertices and edges are in five blobs, but the global structure consists of three cycles.



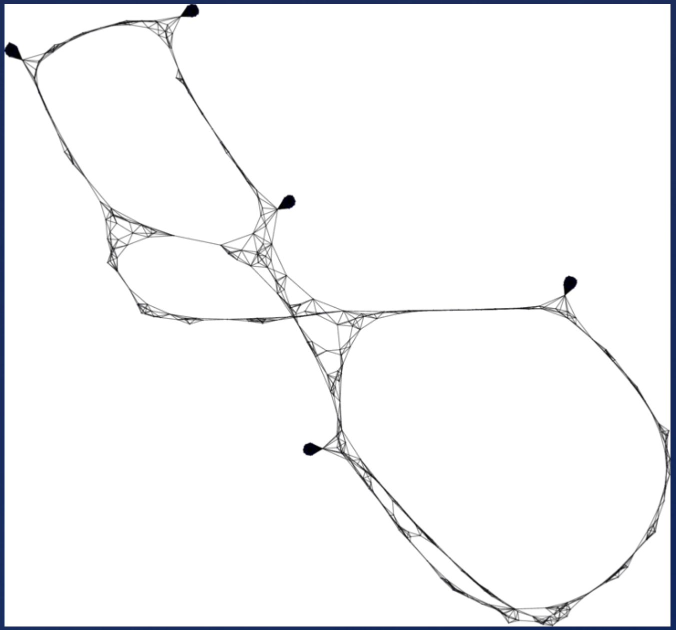
***FM*****[3]** layout for the whole graph

# Typical example 2

`CyclesWithBlobs`

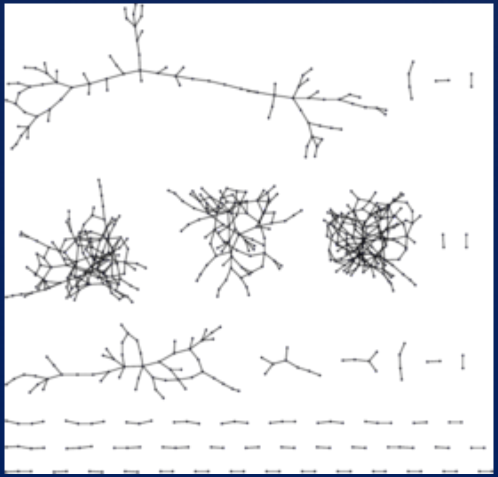
- $n = 1031$
- $m = 22638$

***FM³*** **layout for the whole graph**

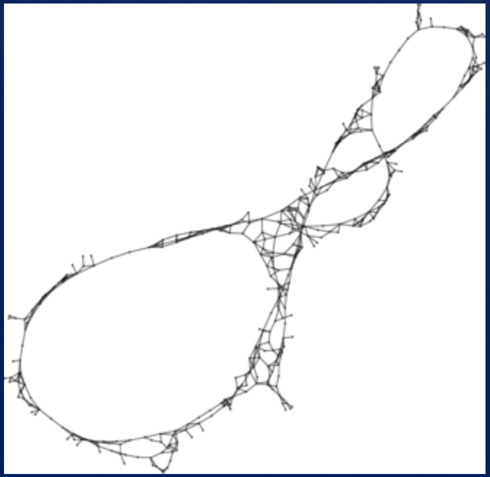


Sparsifications with relative density $d = 3\%$:

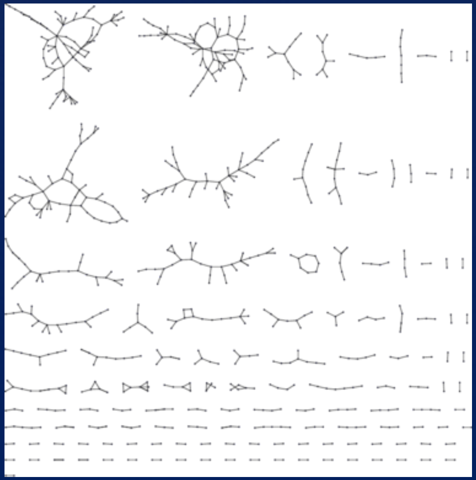
***RES***



***DSS***



***SSS***

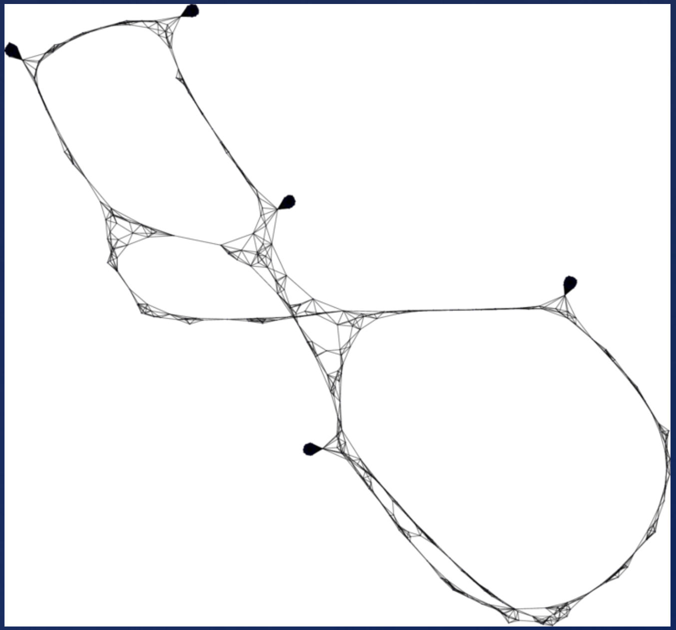


## Typical example 2

`CyclesWithBlobs`

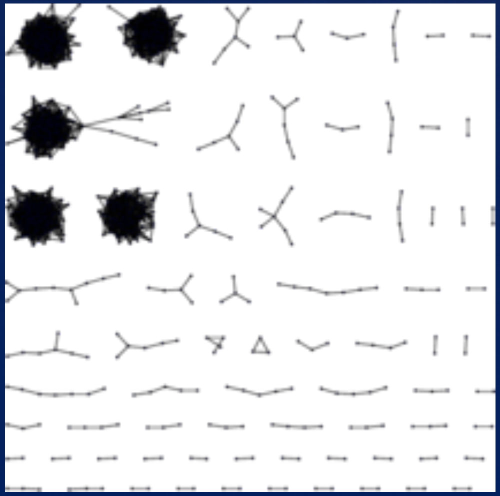
- $n = 1031$
- $m = 22638$

***FM³* layout for the whole graph**

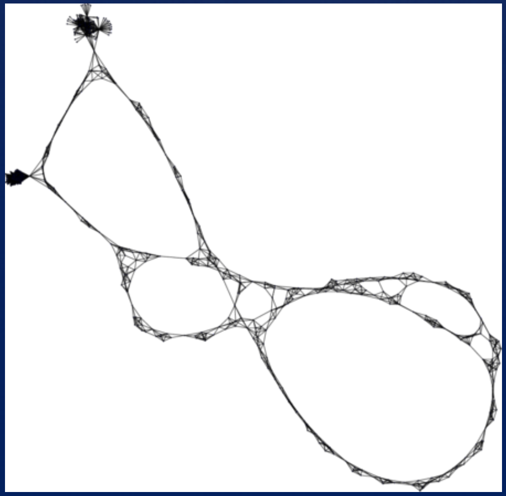


Sparsifications with relative density $d = 15\%$:

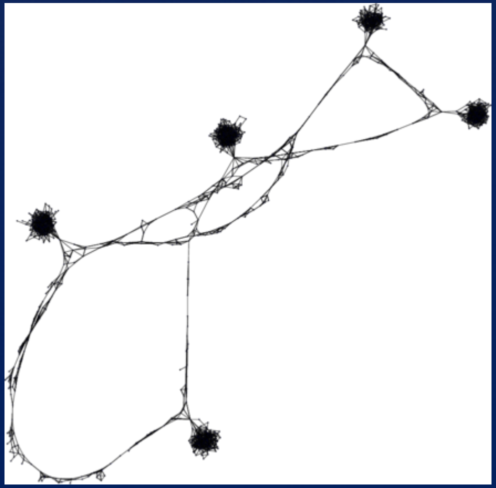
**RES**



**DSS**



**SSS**

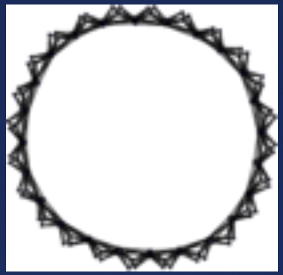


## *Atypical example*

`Can_144`

- $n = 144$
- $m = 576$

- Globally a cycle.



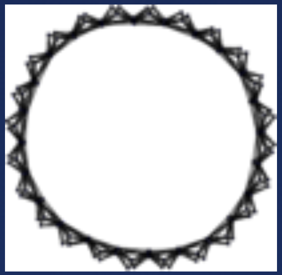
***FM³*** layout for the whole graph

Atypical example

`Can_144`

- $n = 144$
- $m = 576$

- Globally a cycle.



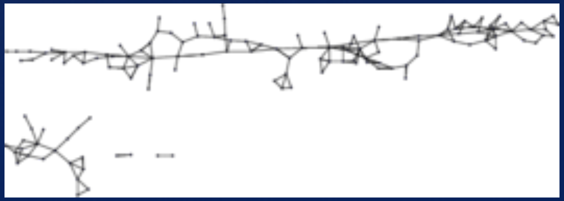
***FM***[3] layout for the whole graph

*Sparsifications with relative density* $d = 40\%$:

***RES***

***DSS***

***SSS***

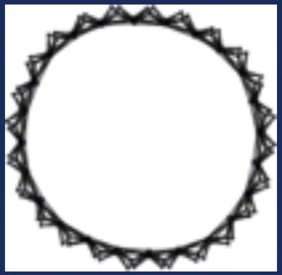


Atypical example

Can_144

- $\boldsymbol{n = 144}$
- $\boldsymbol{m = 576}$

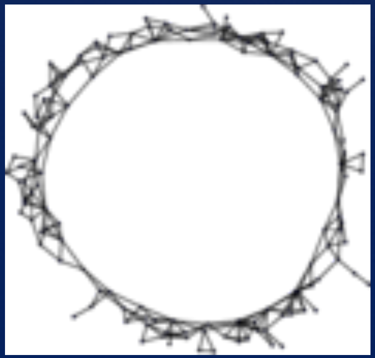
- Globally a cycle.



***FM[3]*** layout for the whole graph

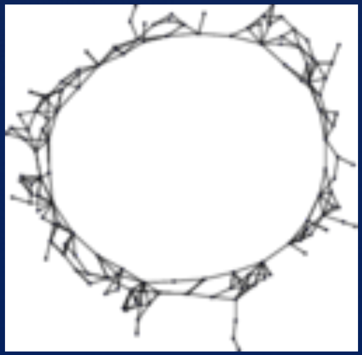
*Sparsifications with relative density* $\boldsymbol{d} = \mathbf{50\%}$:

***RES***

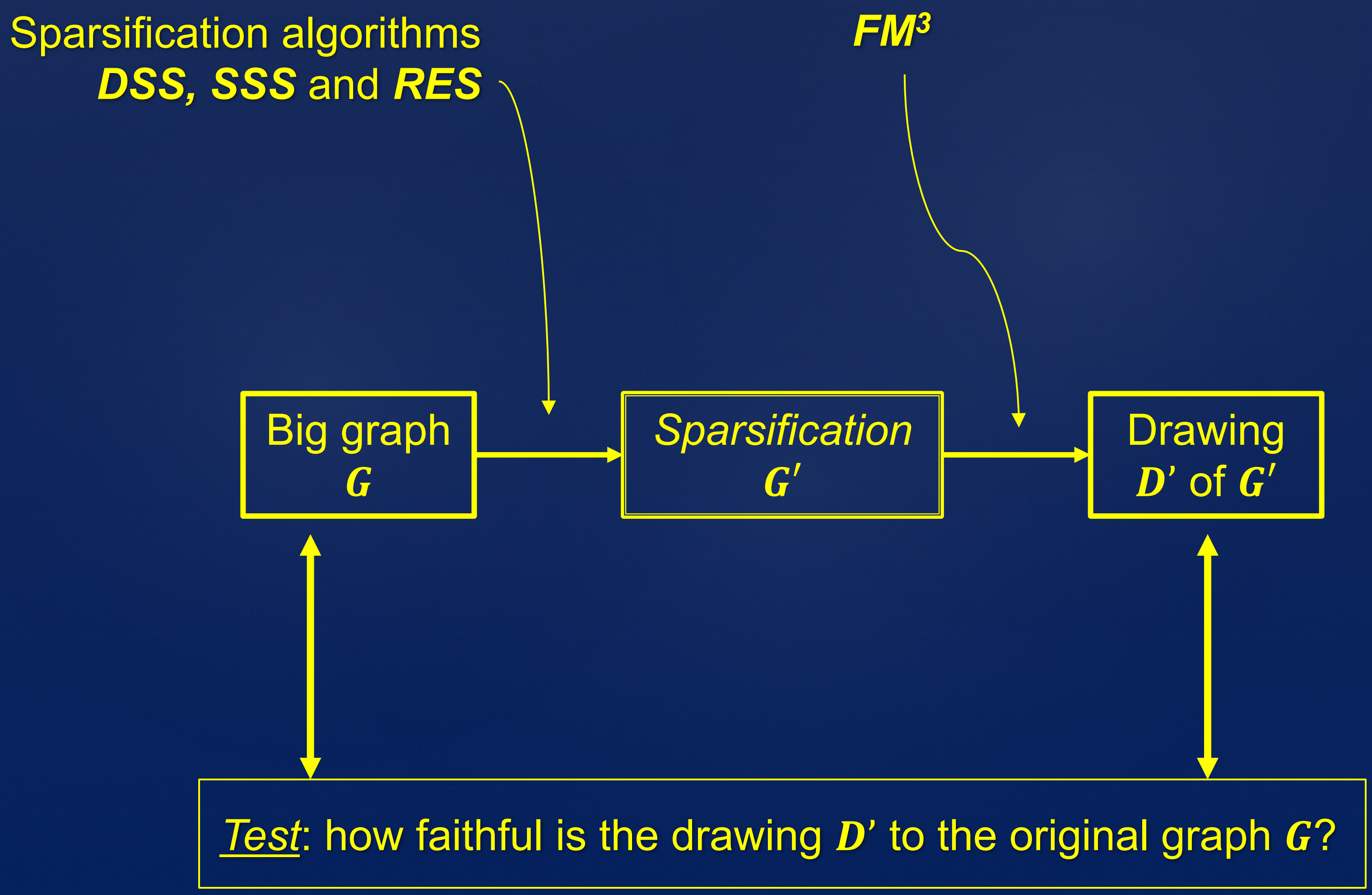


***DSS***

***SSS***



# *Experiments*



*Experiments*

*Data*

Some “*defacto-benchmark*” graphs: from

- Hachul library
- Walshaw’s Graph Partitioning Archive
- Sparse matrices collection
- The network repository

Some *GION* graphs:

- RNA sequence graphs
- Locally dense and globally sparse
- Generally have distinctive ”stringy” shapes

Some *blobby* graphs:

- Randomly generated
- Contain structures that are difficult to model with sparsification
- Some large and dense parts (blobs) connected by a few edges

## *Experiments*

*<u>Implementation</u>*

- Java and OpenIMAJ
- Moore-Penrose inverse computed by OpenIMAJ, with Java 8, 16GB heap, multiple threads
- Dell XPS13, i7, 16GB memory, 512GB SSD
- Ubuntu 16.04, 20GB swap memory

Density

- ***DSS***, ***SSS***, and ***RES*** were run for relative density values from 1% to 100%

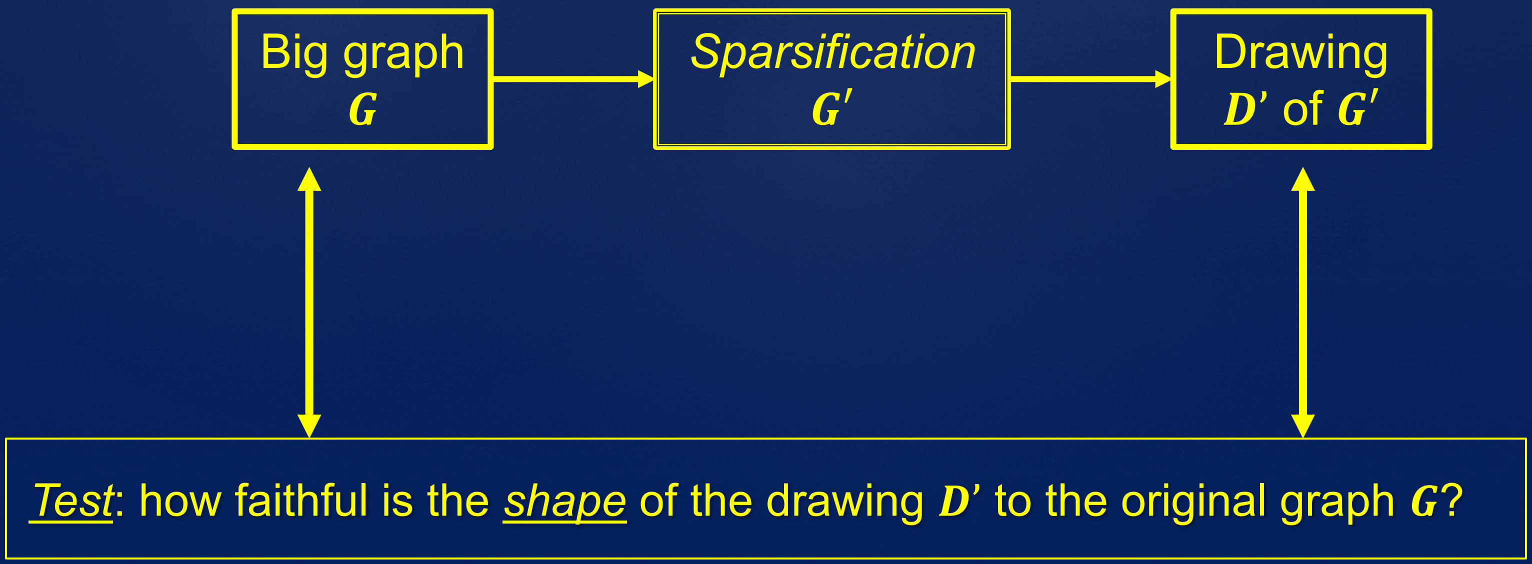
Layout

- Tested with several layout algorithms, but reported results are for ***FM***$^3$.
- Results consistent across layout algorithms.

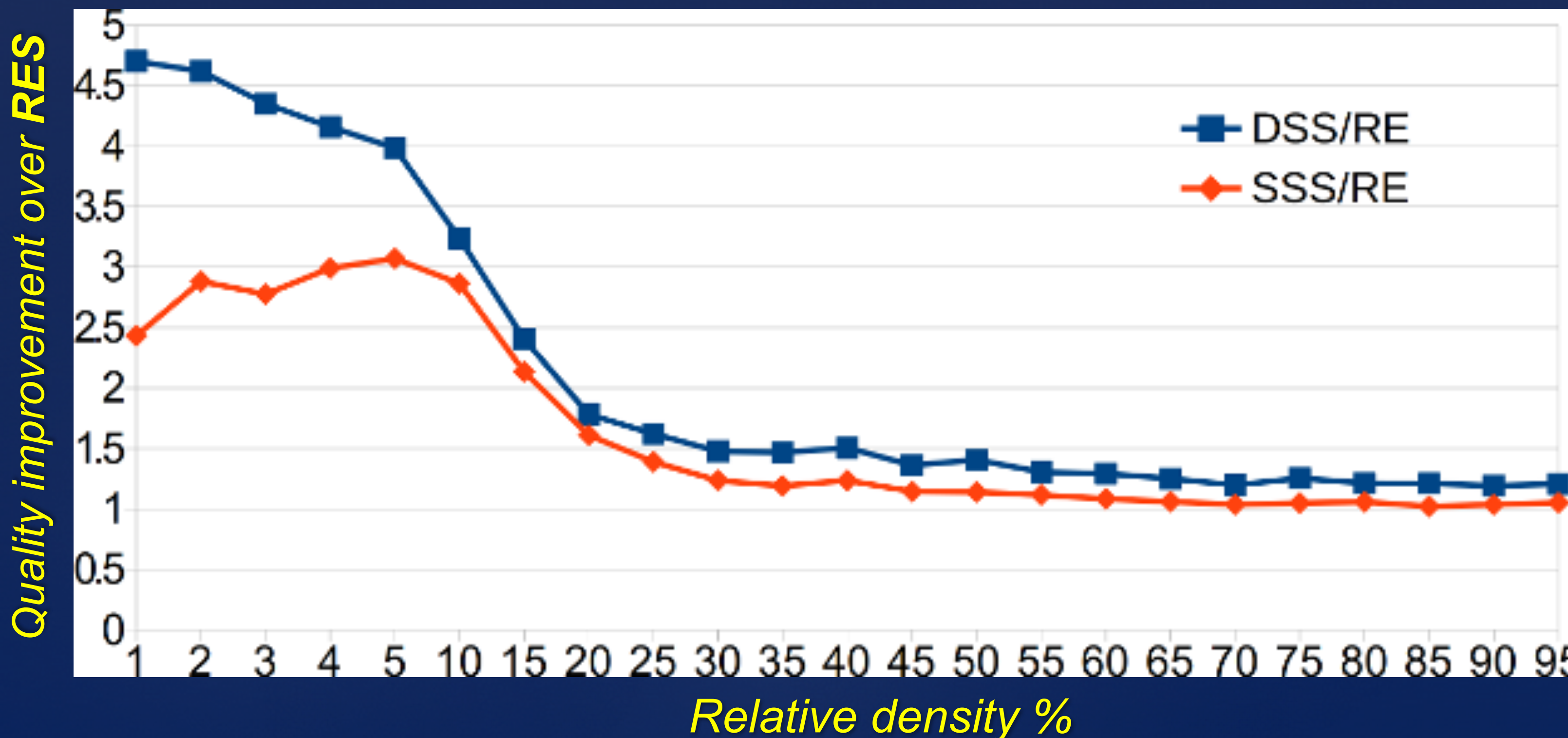
## *Experiments*

Quality metric

- Shape-based faithfulness
  - Measures how well the shape of the picture represents the graph.
  - Does not measure readability.



# *Main results*




1. For low relative density ($d < 20\%$): Both ***DSS*** and ***SSS*** are better than ***RES***
2. For very low relative density ($d < 10\%$): ***DSS*** is better than ***SSS***
3. For higher relative density ($d > 30\%$): all methods are about the same.

## *Runtime*

The dominant function is the Moore-Penrose inverse.

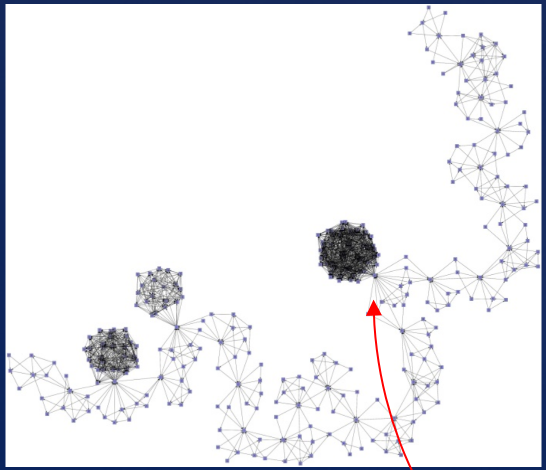
- Our experiments use standard off-the-shelf software to compute the Moore-Penrose inverse.
- Can take several minutes
- Stochastic algorithms that are theoretically fast $\left(O(m \log m)\right)$ are available, but untried in practice.

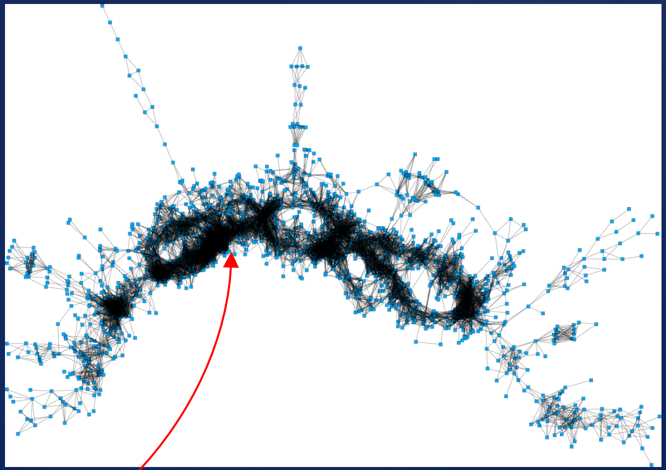
*Question*: What is a *big* graph?

*Answer*: A graph $G$ is *big* if

- Every drawing of $G$ has a blob, that is,
- Every drawing of $G$ is unfaithful.

*Depends on screen resolution*





Blobs

A *blob* occurs when

- The screen does not have enough pixels to represent the drawing precisely, that is,
- The visualization function is not 1-1, that is,
- The drawing is unfaithful

## *Conclusion*

Spectral sparsification has considerable potential for big graph drawing.