Network Science

#21 Graph visualization

The layout problem

Algorithms for graph visualization

Before

- always based on some properties: tree, series-parallel graph, planar graph
- and on some additional information: ordering of the vertices, decompositions into SP-components

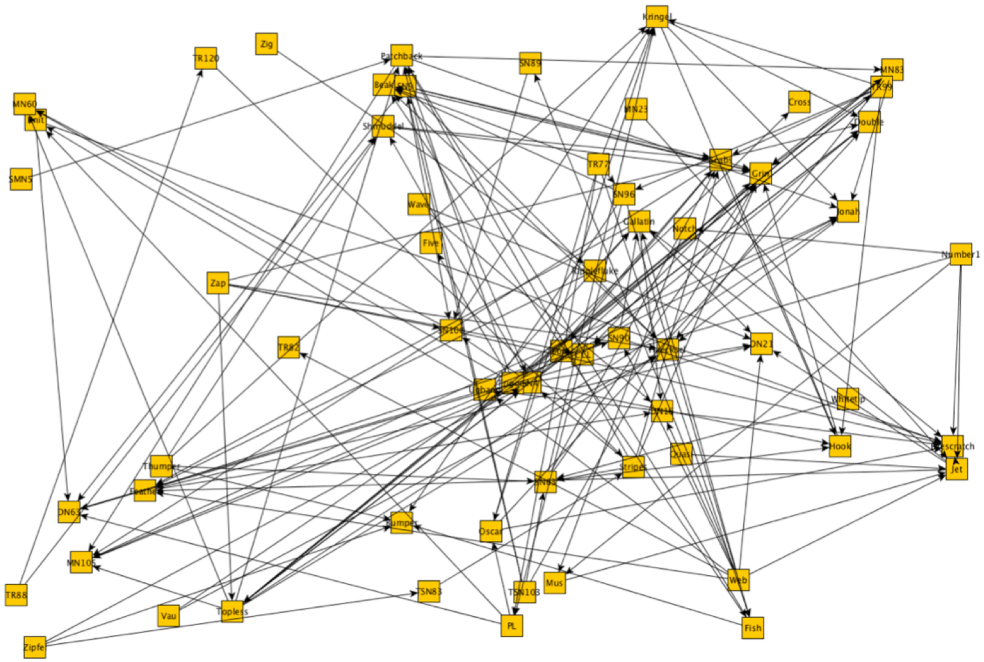
Today

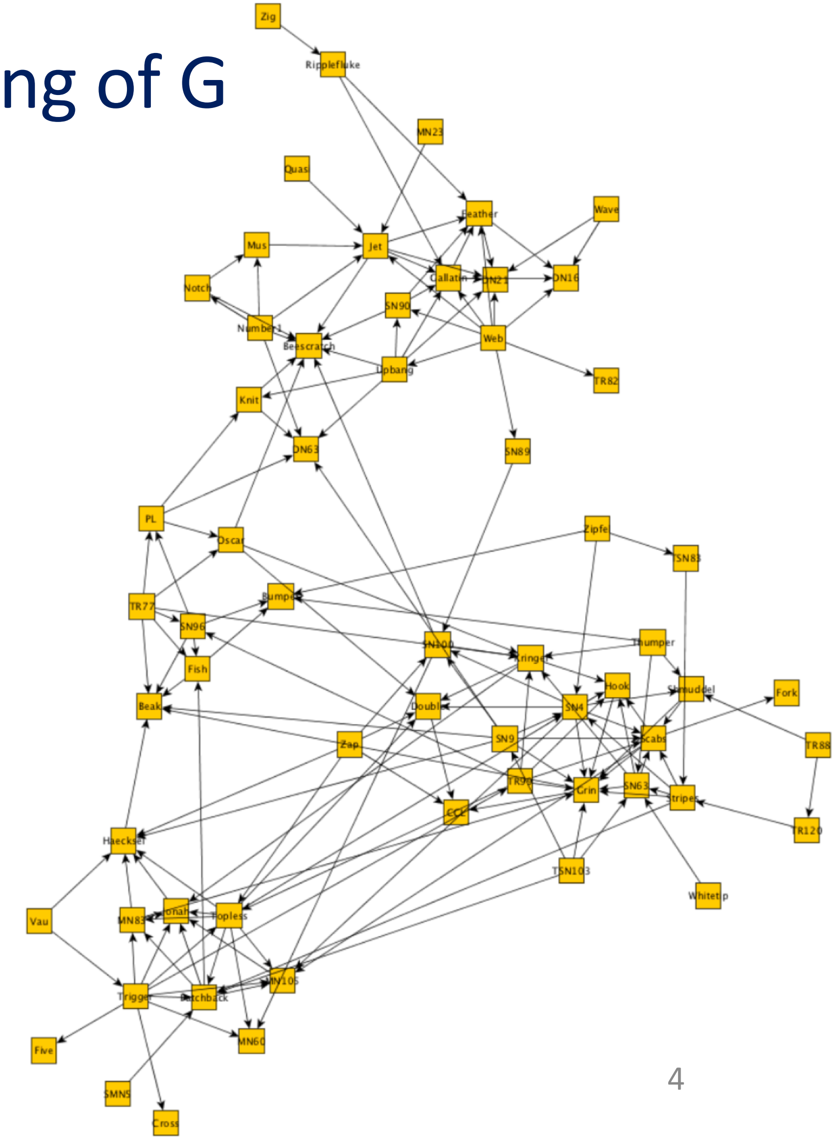
- more direct and intuitive method based on **physical analogies**
- The methods are very popular: intuitiveness, easy to program, generality, fairly satisfactory results,...

General layout problem

Given a graph G=(V,E)

Find a clear and readable drawing of G





Which aesthetic criteria would you optimize?

Aesthetic criteria

- adjacent nodes are close
- non-adjacent far apart
- edges short, straight-line, similar length
- densely connected parts (clusters) form **communities**
- as few crossings as possible
- nodes distributed evenly

… but optimization criteria partially contradict each other

Examples

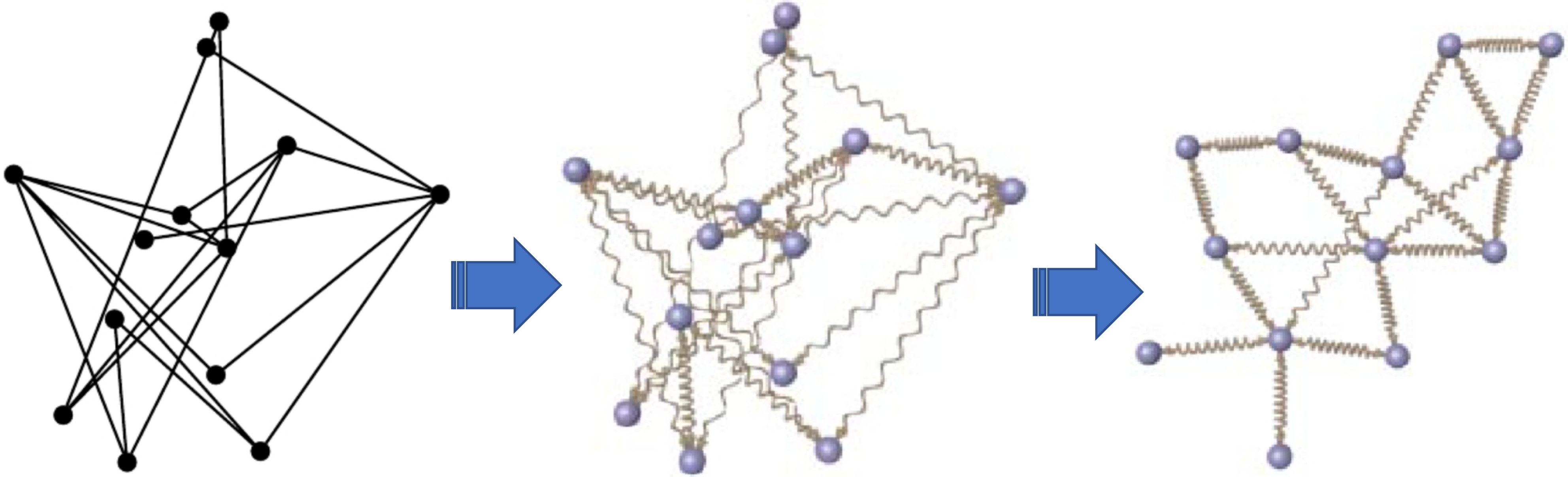
Given graph $G = (V, E)$, required edge length $l(e)$
Find drawing of G which realizes all the edge lengths

NP-hard for
- edge lengths $\{1, 2\}$ [Saxe, '80]
- planar drawing with unit edge lengths [Eades, Wormald, '90]

Spring-embedder algorithms

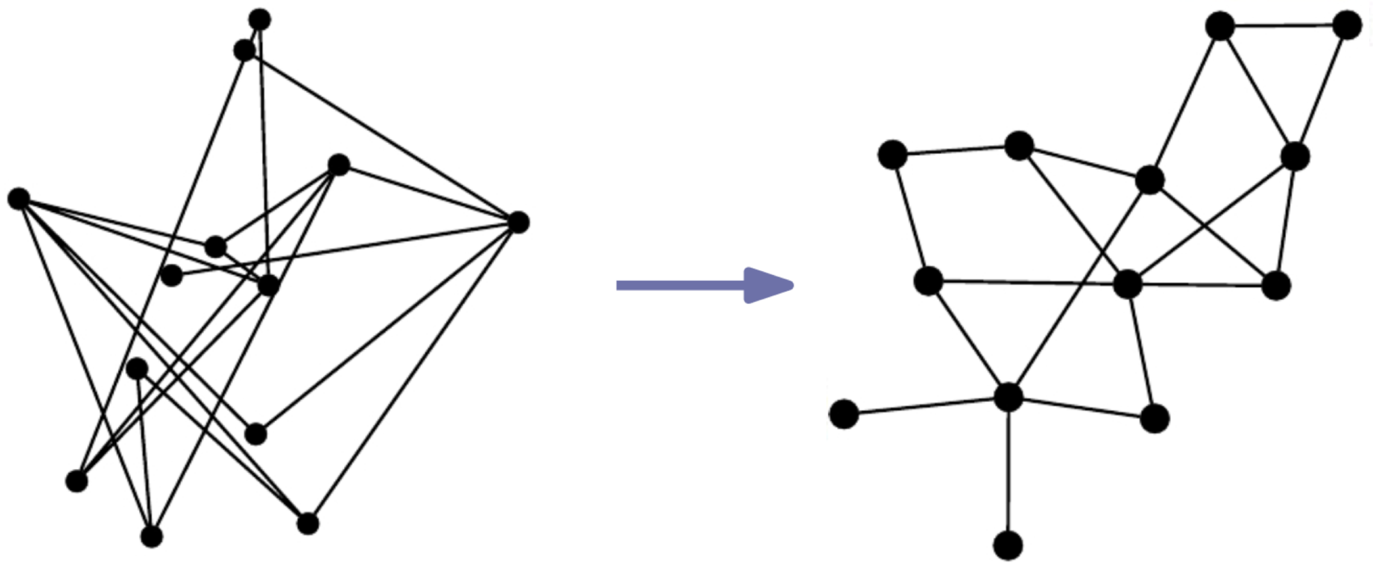
Eades, "A heuristic for graph drawing" (1984)

The physical model



"To embed a graph we replace the vertices by **steel rings** and replace each edge with a **spring** to form a mechanical system . . . The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a **minimal energy state**."

Spring-embedder algorithms



So-called **spring-embedder** algorithms that work according to this or similar principles are among the most frequently used graph-drawing methods in practice

Notation

- $l = l(e)$ ideal spring length for edge e
- $p_v = (x_v, y_v)$ position of node v
- $|p_v - p_u|$ Euclidean distance between u and v
- $\overrightarrow{p_v p_u}$ unit vector pointing from v to u
- 

Spring-embedder model (Eades, 1984)

- **repulsive** force between non-adjacent nodes u and v

$$f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{||p_v - p_u||^2} \cdot \overrightarrow{p_u p_v}$$

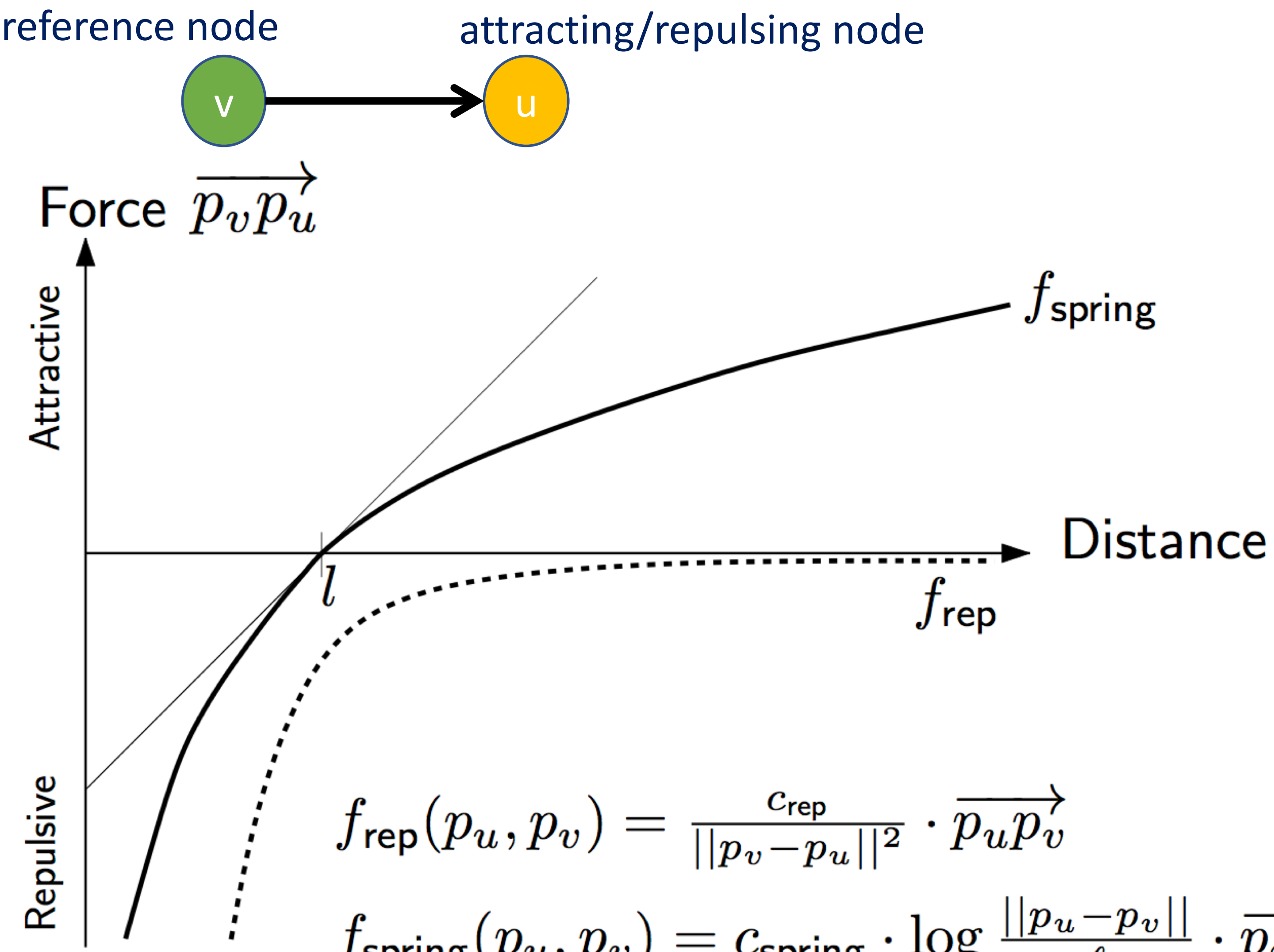
- **attractive** force between adjacent vertices u and v

$$f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \cdot \log \frac{||p_u - p_v||}{\ell} \cdot \overrightarrow{p_v p_u}$$

- resulting displacement vector for node v

$$F_v = \sum_{u:\{u,v\} \notin E} f_{\text{rep}}(p_u, p_v) + \sum_{u:\{u,v\} \in E} f_{\text{spring}}(p_u, p_v)$$

Diagram of repulsive/attractive forces



$$f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{||p_v - p_u||^2} \cdot \overrightarrow{p_u p_v}$$

$$f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \cdot \log \frac{||p_u - p_v||}{\ell} \cdot \overrightarrow{p_v p_u}$$

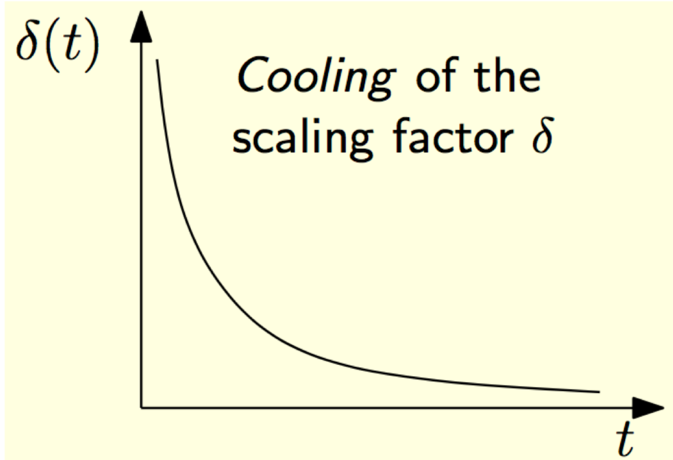
Algorithm

- While forces are sufficiently strong

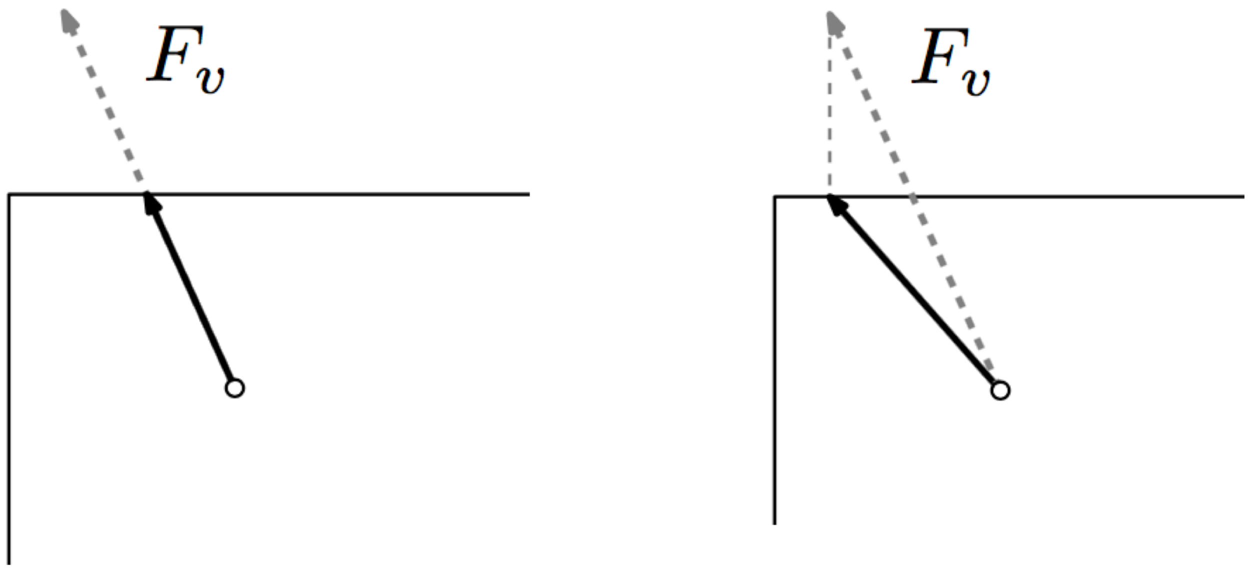
$$\max_{v \in V} \|F_v(t)\| > \varepsilon$$

- Update node position

$$p_v \leftarrow p_v + \delta \cdot F_v(t)$$



Bounded drawing area



If force F_v drives out of R, we adapt the vector appropriately within R

Discussion

Advantages

- very simple Algorithm
- good results for small and medium-sized graphs
- good representation of symmetry/structure

Disadvantages

- system is not stable at the end
- converging to local minima
- timewise f_{spring} in $O(|E|)$ and f_{rep} in $O(|V|^2)$

Influence

- Basis for many further ideas

Variants

Fruchterman and Reingold (1991)

Fruchterman & Reingold (1991). Graph drawing by force-directed placement
http://www.mathe2.uni-bayreuth.de/axel/papers/reingold:graph_drawing_by_force_directed_placement.pdf

- **repulsive** force between all node pairs

$$f_{\text{rep}}(p_u, p_v) = \frac{\ell^2}{||p_v - p_u||} \cdot \overrightarrow{p_u p_v}$$

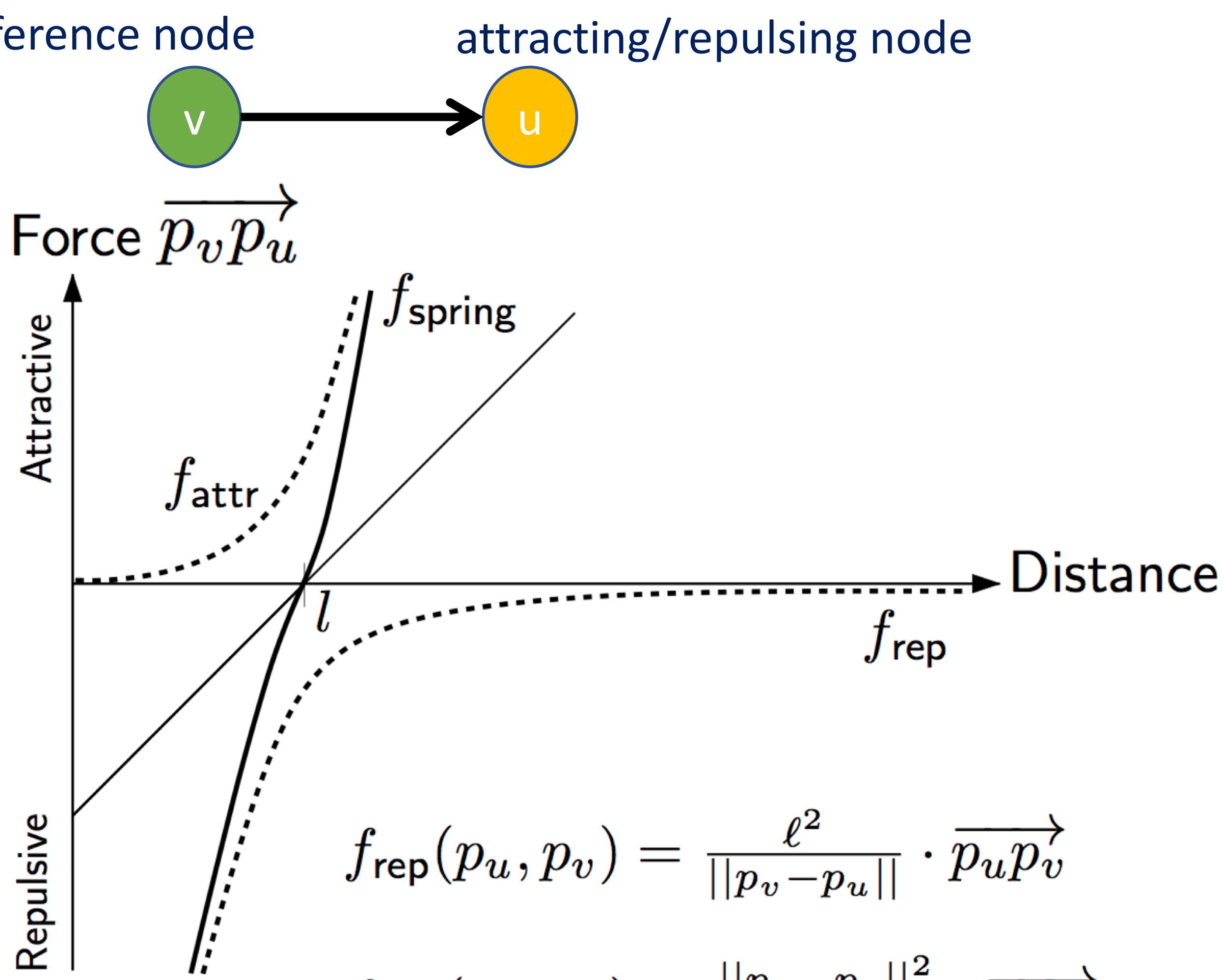
more repulsive

- **attractive** force between adjacent vertices u and v

more attractive

$$f_{\text{attr}}(p_u, p_v) = \frac{||p_u - p_v||^2}{\ell} \cdot \overrightarrow{p_v p_u}$$

Diagram of repulsive/attractive forces



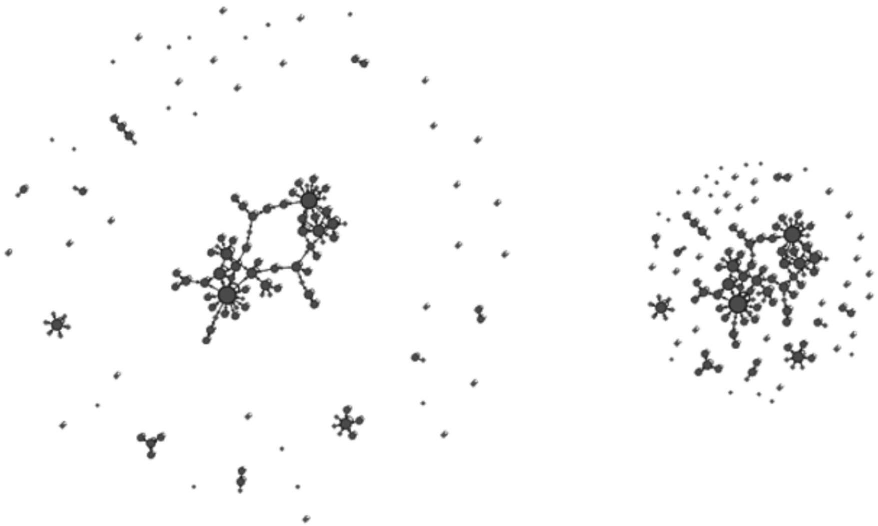
$$f_{\text{rep}}(p_u, p_v) = \frac{\ell^2}{||p_v - p_u||} \cdot \overrightarrow{p_u p_v}$$

$$f_{\text{attr}}(p_u, p_v) = \frac{||p_u - p_v||^2}{\ell} \cdot \overrightarrow{p_v p_u}$$

$$f_{\text{spring}}(p_u, p_v) = f_{\text{rep}}(p_u, p_v) + f_{\text{attr}}(p_u, p_v)$$

MIME

Gravity



$$\Phi(v) = 1 + \deg(v)/2$$

node mass

baricenter

$$p_{\text{bary}} = 1/|V| \cdot \sum_{v \in V} p_v$$

$$f_{\text{grav}}(p_v) = c_{\text{grav}} \cdot \Phi(v) \cdot \overrightarrow{p_v p_{\text{bary}}}$$

prevents disconnected components (islands) from drifting away; attracts nodes to the centre of the spatialisation. Its main purpose is to compensate repulsion for nodes that are far away from the centre.

Force atlas 2 (2014)

Jacomy, Venturini, Heymann, Bastian (2014).
ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098679

- **repulsive** force between all node pairs

correction

$$f_{\text{rep}}(p_u, p_v) = k_r \frac{(deg(u)+1)(deg(v)+1)}{||p_u - p_v||}$$

as F&R

- **attractive** force between adjacent vertices u and v

less attractive

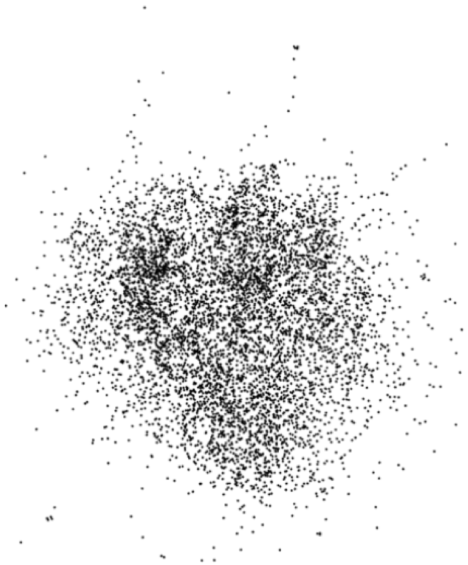
$$f_{\text{attr}}(p_u, p_v) = ||p_u - p_v|| \cdot \overrightarrow{p_v p_u}$$

in lin-log mode

$$log(1 + ||p_u - p_v||)$$

MIME.

Force atlas 2 (2014)



Fruchterman & Reingold

ForceAtlas2

ForceAtlas2 LinLog mode

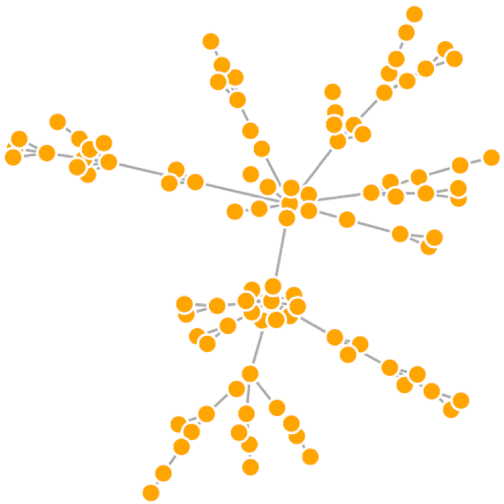
Comparison

	ATTRACTIVE	REPULSIVE
Spring model	$\log \frac{\|\|p_u - p_v\|\|}{\ell}$	$\frac{c_{\text{rep}}}{\|\|p_v - p_u\|\|^2}$
Fruch & Rein	$\frac{\|\|p_u - p_v\|\|^2}{\ell}$	$\frac{\ell^2}{\|\|p_v - p_u\|\|}$
Force Atlas 2	$\|\|p_u - p_v\|\|$	$\frac{(deg(u)+1)(deg(v)+1)}{\|\|p_u - p_v\|\|}$
LinLog mode	$log(1 + \|\|p_u - p_v\|\|)$	

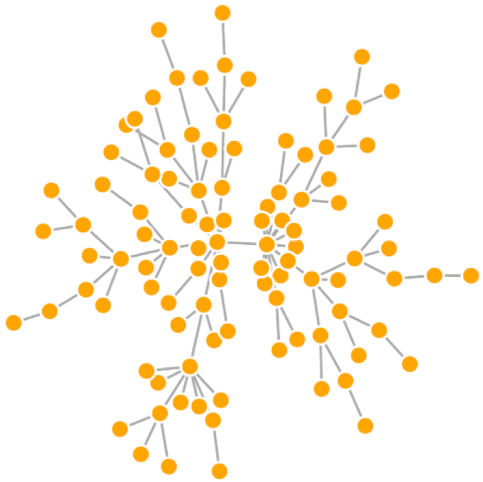
MIME.

Comparison (R igraph library)

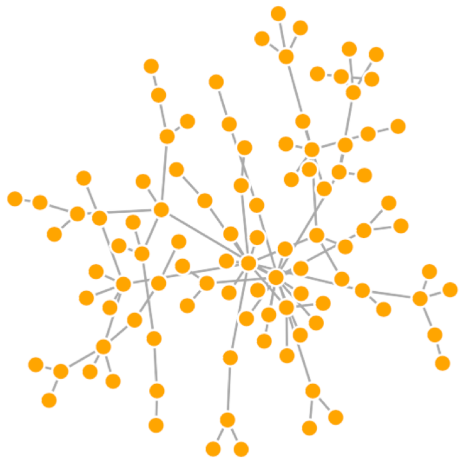
Fruchterman & Reingold



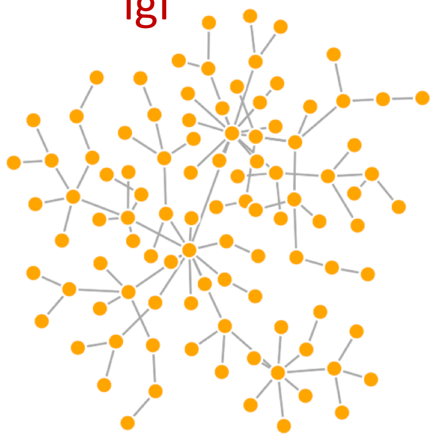
Kamada Kawai



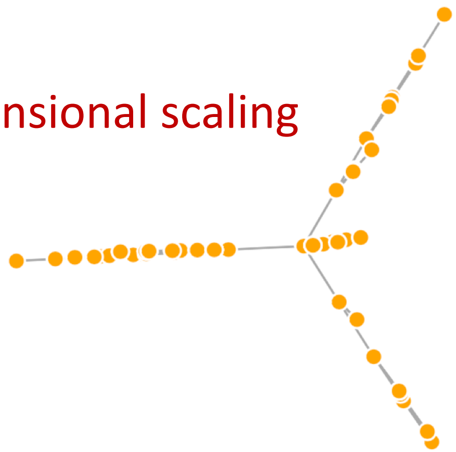
Graph opt



Igl



Multidimensional scaling



Discussion

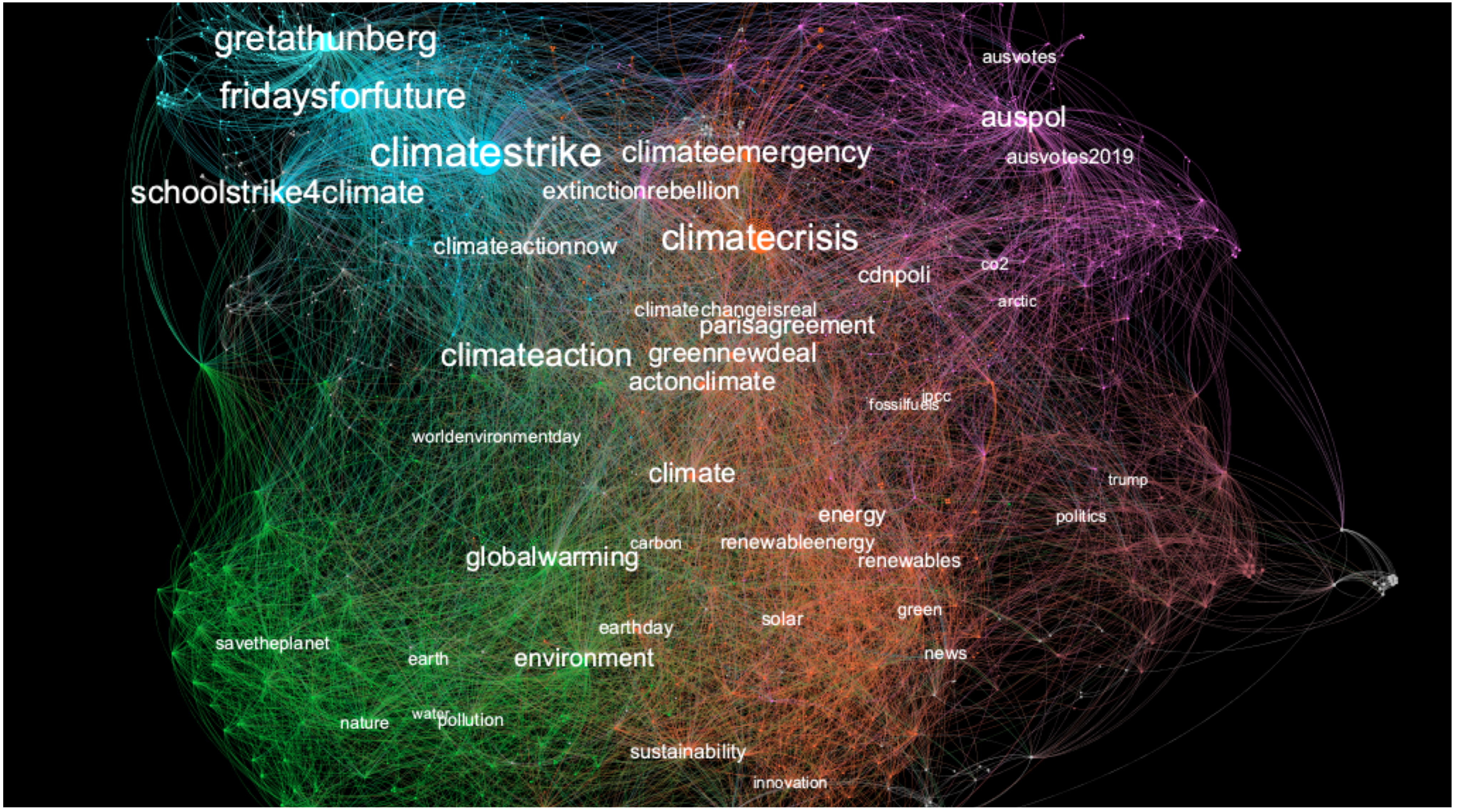
Force-based Approaches are

- easily understandable and implementable
- depending on the graphs (small and sparse)
- amazingly good layouts (Symmetries, Clustering)
- easily adaptable and configurable
- robust
- scalable

But...

- no quality and running time guarantees
- bad choice of starting layout → slow convergence
- possibly slow for large graphs
- fine-turning can be done by experts

Example


gretathunberg
fridaysforfuture
climatestrike
climateemergency
schoolstrike4climate
extinctionrebellion
climateactionnow
climatecrisis
cdnpoli
ausvotes
auspol
ausvotes2019
co2
arctic
climatechangeisreal
parisagreement
climateaction
greennewdeal
actonclimate
fossilfuels
ipcc
worldenvironmentday
climate
trump
politics
energy
carbon
renewableenergy
renewables
globalwarming
green
earthday
solar
savetheplanet
earth
environment
news
nature
water
pollution
sustainability
innovation

Grip

Gajer, Kobourov (2004). Grip: Graph drawing with intelligent placement
http://emis.um.ac.ir/journals/JGAA/accepted/2002/GajerKobourov2002.6.3.pdf

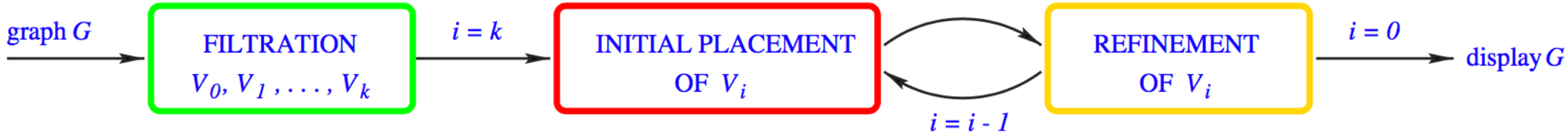
GRIP – Graph dRawing with Intelligent Placement

Motivation

- Spring-Embedder for large graphs are too slow
- sensitivity to initialisation of node positions

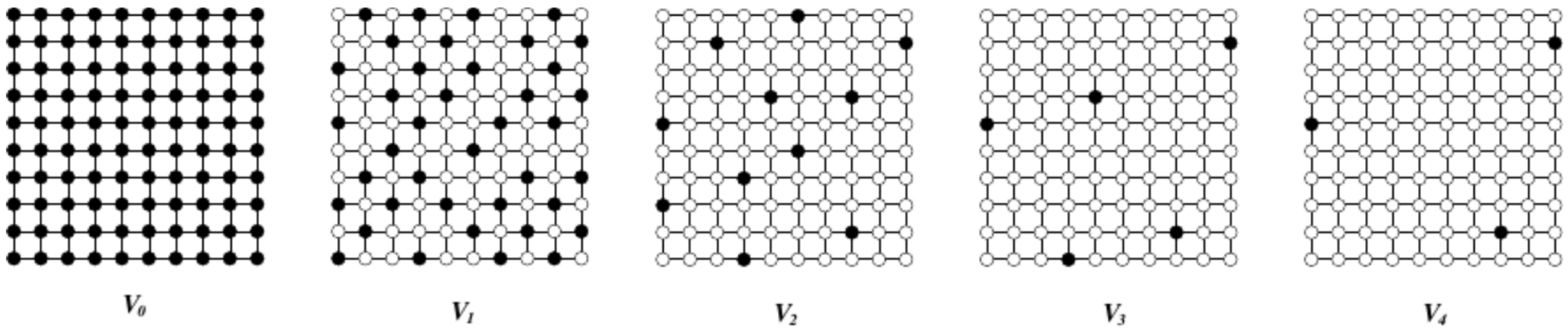
Approach

- top-down graph coarsening/filtration
- bottom-up calculation of the layout
- clever placement of new nodes
- force-based refinement of their positions



Maximal independent set (MIS) filtering

- Sequence of node sets $V=V_0 \supset V_1 \supset \cdots \supset V_k \supset \emptyset$
- distance in G between nodes in V_i is $\geq 2^{i-1} + 1$
- can be done by **BFS** by deleting nodes closer than bound
- good balance between size of a level and depth of decomposition



Level-based node placement

Step 1

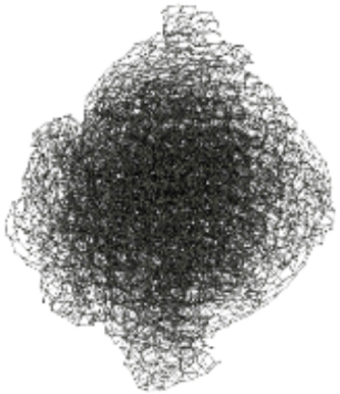
- for each node $v \in V_i \setminus V_{i+1}$ find optimal position with respect to three adjacent nodes V_{i+1}

Step 2

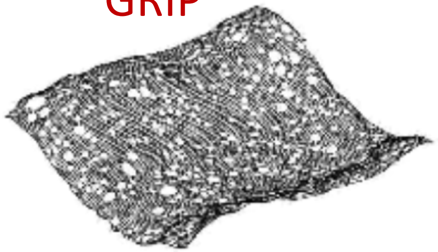
- perform force-based refinement, where forces are computed locally only to a constant number of nearest neighbours in V_i

Experiments

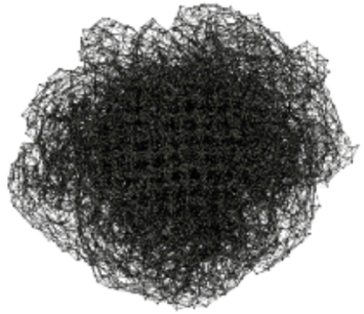
Fruchterman & Reingold



GRIP

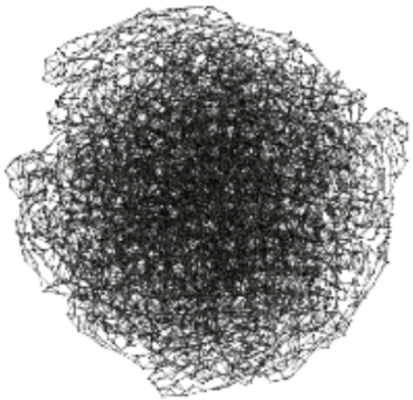


Fruchterman & Reingold

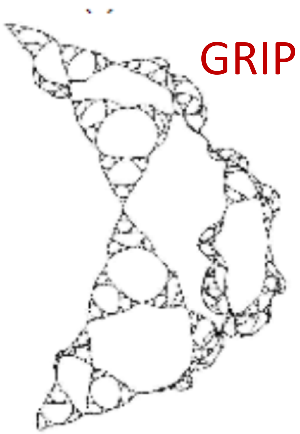


GRIP

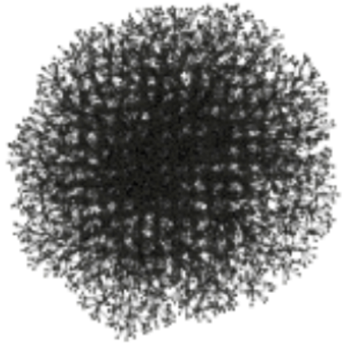
Fruchterman & Reingold



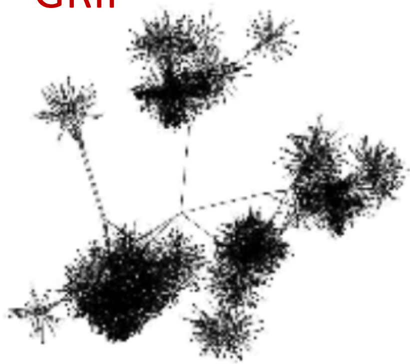
GRIP



Fruchterman & Reingold



GRIP

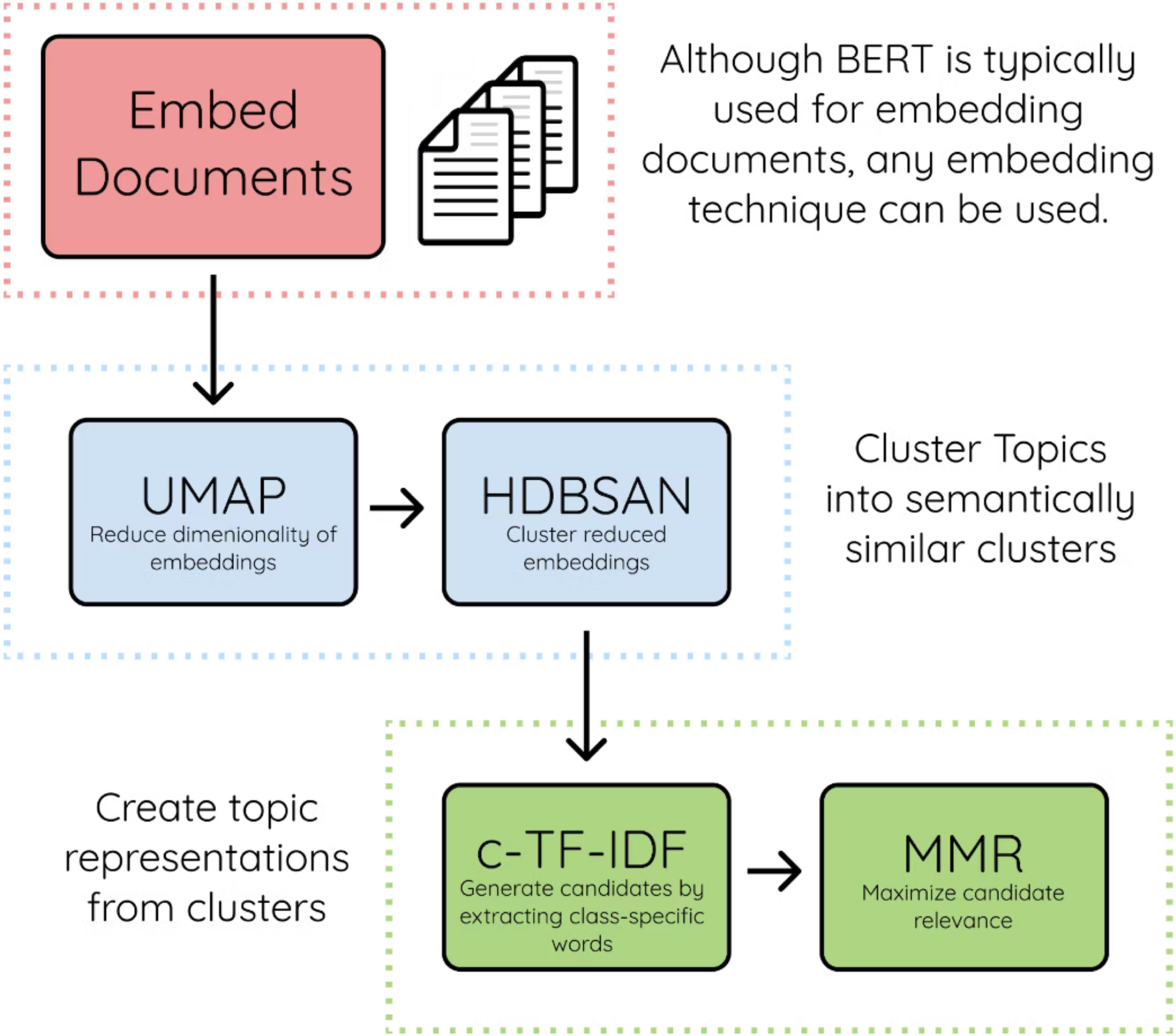


BERTopic

M. Grootendorst (2022)
BERTopic: Neural topic modeling with a class-based TF-IDF procedure
https://arxiv.org/abs/2203.05794

What is BERTopic?

from bertopic import BERTopic



MIME.

What is UMAP

https://arxiv.org/abs/1802.03426

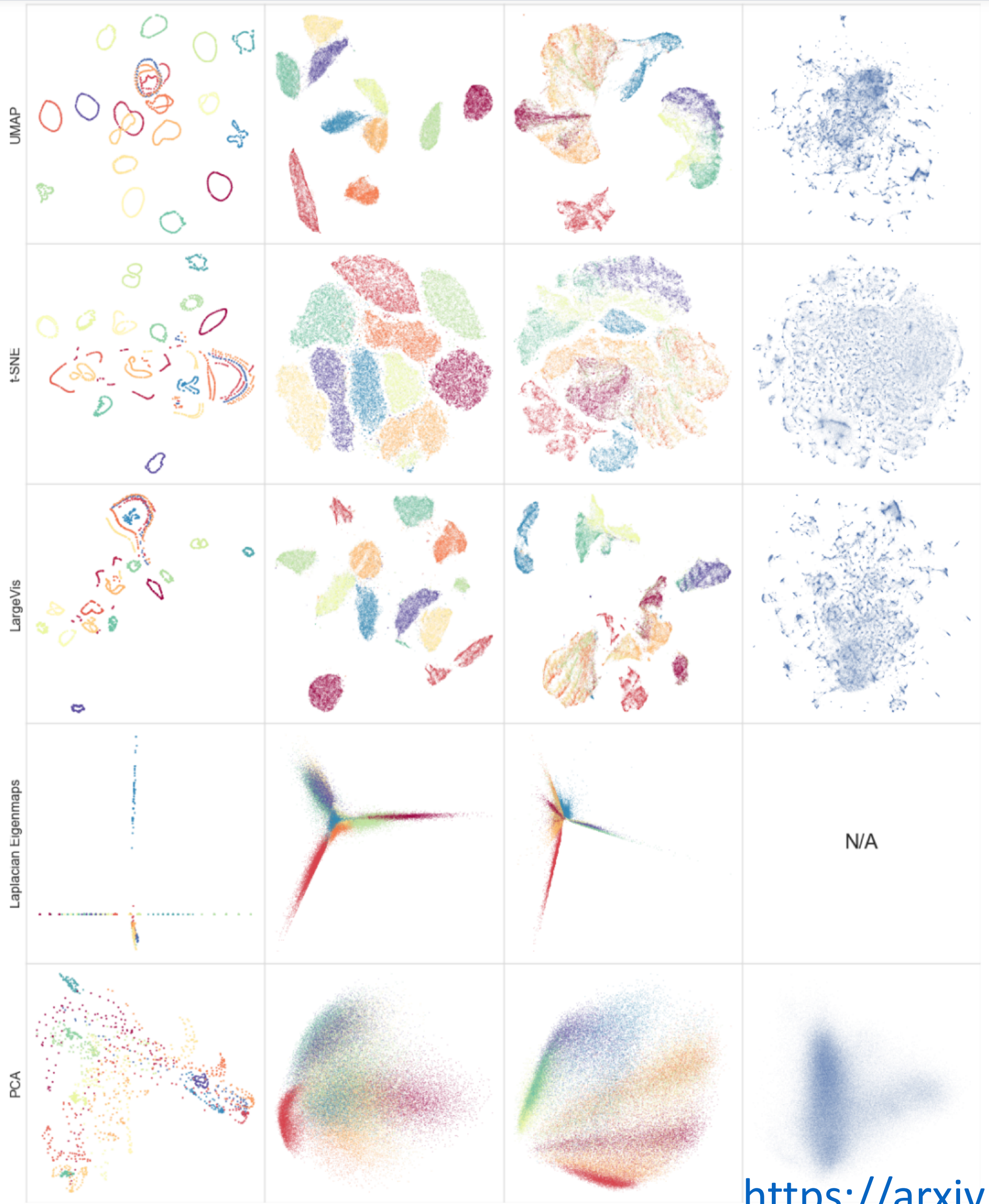
Idea

- Identify distances with k-nearest neighbours
- Apply a force-directed algorithm
- This keeps local info

	ATT	REP
spring	$\log(d_{u,v}/\ell)$	$\frac{c_{\text{rep}}}{d_{u,v}^2}$
Fruch & Rein	$\frac{d_{u,v}^2}{\ell}$	$\frac{\ell^2}{d_{u,v}}$
Force Atlas 2	$d_{u,v}$	$\frac{(1+\deg(u))(1+\deg(v))}{d_{u,v}}$
UMAP	$\frac{d_{u,v}^{2(b-1)}}{1+d_{u,v}^2}\ \ell$	$\frac{1-\ell}{(\epsilon+d_{u,v}^2)(1+ad_{u,v}^{2b})}$

MIME.

Comparison



UMAP

t-SNE

LargeVis

Laplacian eigenvector

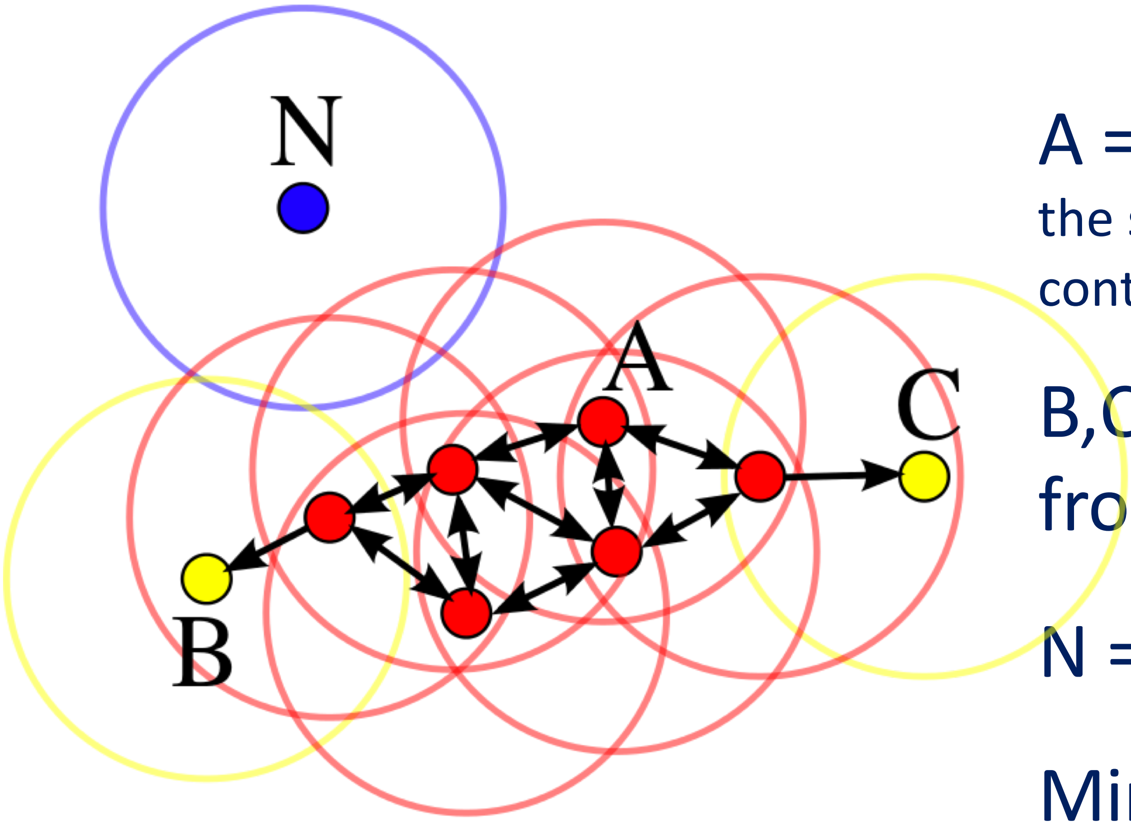
PCA

https://arxiv.org/abs/1802.03426

What is HDBSCAN

https://en.wikipedia.org/wiki/DBSCAN

Builds clusters by measuring distances



A = core points
the surrounding area
contains other 4 points

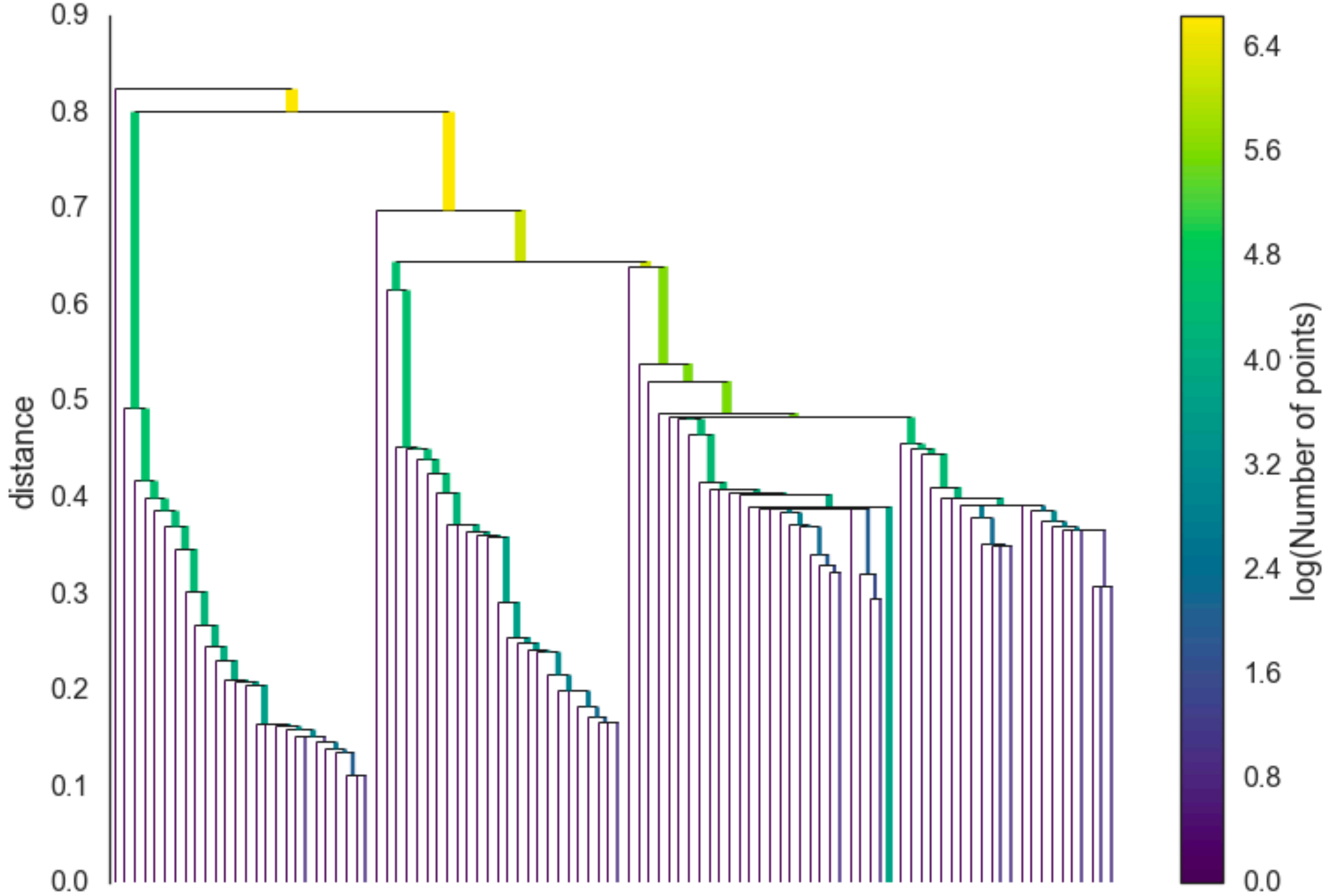
B,C = reachable
from A

N = noise point

Minimum radius
to be set

Hierarchical output

https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



What is c-TF-IDF

probability of word
w in cluster c

probability of appearance
of word w in documents

ranking

$$R_{w,c} = \frac{N_{w,c}}{\sum_w N_{w,c}} \cdot -\log\left(\frac{\sum_d N_{w,d}}{\sum_d 1}\right) \quad , \quad N_{w,c} = \sum_{d \in \mathcal{D}_c} N_{w,d}$$

word

cluster (of documents)

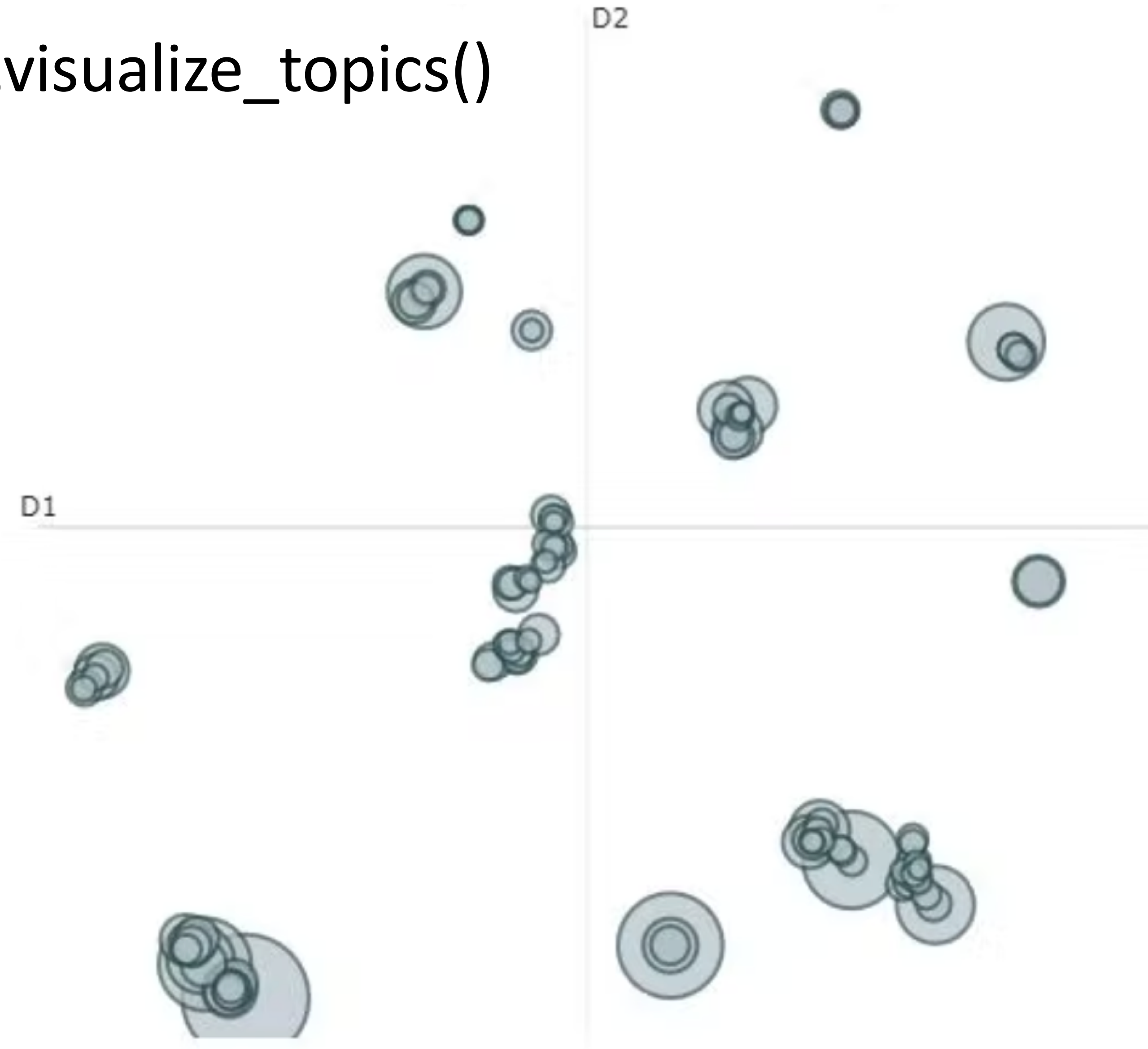
document

words which occur in many documents
are rated less important

MIME.

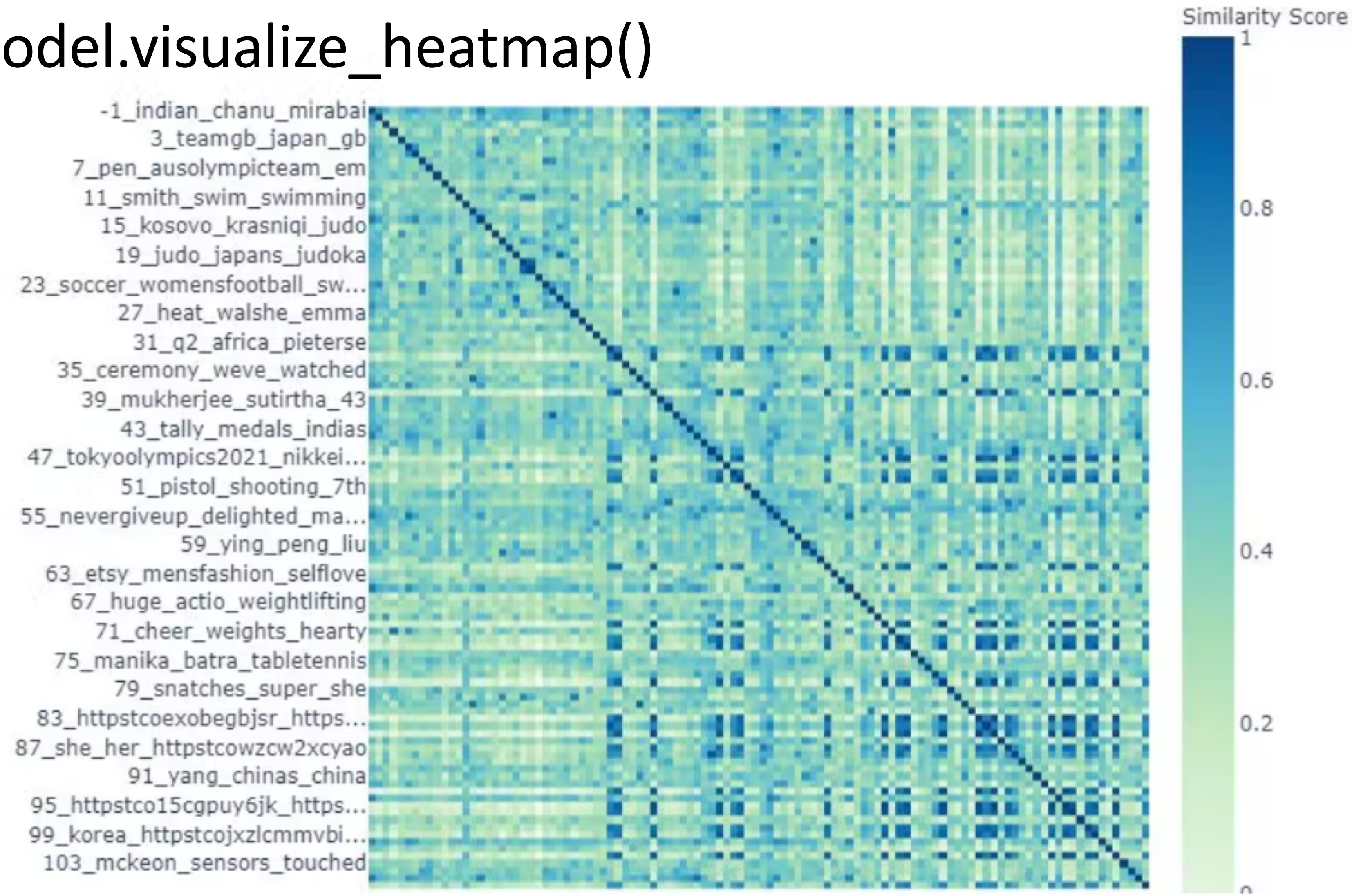
Intertopic distance map

model.visualize_topics()



Topic similarity

model.visualize_heatmap()



Visualize documents

topic_model.visualize_documents()