#21 Graph visualization

© 2020 T. Erseghe
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,...
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)
“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.”
So-called **spring-embedder** algorithms that work according to this or similar principles are among the most frequently used graph-drawing methods in practice.
- \( l = l(e) \) 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 \( u \)
- reference node
- attracting/repulsing node
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 $v$

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

- resulting displacement vector for node $v$

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

Mi ME.
Diagram of repulsive/attractive forces

Reference node \( v \) attracting/repulsing node \( u \)

\[
\begin{align*}
\text{Force} & \quad \frac{p_v - p_u}{p_v p_u} \\
\text{Attractive} & \quad f_{\text{spring}} \\
\text{Repulsive} & \quad f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{||p_v - p_u||^2} \cdot \frac{p_u p_v}{p_u p_v} \\
& \quad f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \cdot \log \frac{||p_u - p_v||}{\ell} \cdot \frac{p_v p_u}{p_v p_u}
\end{align*}
\]
Algorithm

- While forces are sufficiently strong
  \[ \max_{v \in V} \| F_v(t) \| > \varepsilon \]

- Update node position
  \[ p_v \leftarrow p_v + \delta \cdot F_v(t) \]
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 \( f_{\text{spring}} \) in \( O(|E|) \) and \( f_{\text{rep}} \) in \( O(|V|^2) \)

Influence
- Basis for many further ideas
Variants
Fruchterman and Reingold (1991)

Fruchterman & Reingold (1991). Graph drawing by force-directed placement

- **repulsive** force between all node pairs

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

more repulsive

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

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

more attractive
Diagram of repulsive/attractive forces

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

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

\[ f_{\text{spring}}(p_u, p_v) = f_{\text{rep}}(p_u, p_v) + f_{\text{attr}}(p_u, p_v) \]
Gravity

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.

\[
\Phi(v) = 1 + \frac{\text{deg}(v)}{2}
\]

\[
f_{\text{grav}}(p_v) = c_{\text{grav}} \cdot \Phi(v) \cdot \frac{p_v p_{\text{bary}}}{|V| \cdot \sum_{v \in V} p_v}
\]

node mass

baricenter
Force atlas 2 (2014)


- **repulsive** force between all node pairs

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

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

  \[ f_{\text{attr}}(p_u, p_v) = \frac{|p_u - p_v| \cdot \frac{p_v}{p_u}}{\log(1 + |p_u - p_v|)} \]
Force atlas 2 (2014)

Fruchterman & Reingold

ForceAtlas2

ForceAtlas2 LinLog mode
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>ATTRACTIVE</th>
<th>REPULSIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spring model</strong></td>
<td>$\log \frac{</td>
<td></td>
</tr>
<tr>
<td><strong>Fruch &amp; Rein</strong></td>
<td>$\frac{</td>
<td></td>
</tr>
<tr>
<td><strong>Force Atlas 2</strong></td>
<td>$</td>
<td></td>
</tr>
<tr>
<td><strong>LinLog mode</strong></td>
<td>$\log(1 +</td>
<td></td>
</tr>
</tbody>
</table>
Comparison (R igraph library)

- Fruchterman & Reingold
- Kamada Kawai
- Igl
- Graph opt
- 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
Gajer, Kobourov (2004). 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 $V = V_0 \supseteq V_1 \supseteq \cdots \supseteq V_k \supseteq \emptyset$
- 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?

Embed Documents

UMAP
Reduce dimensionality of embeddings

HDBSCAN
Cluster reduced embeddings

Although BERT is typically used for embedding documents, any embedding technique can be used.

Cluster Topics into semantically similar clusters

Create topic representations from clusters

c-TF-IDF
Generate candidates by extracting class-specific words

MMR
Maximize candidate relevance
# What is UMAP

**Idea**

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


<table>
<thead>
<tr>
<th></th>
<th>ATT</th>
<th>REP</th>
</tr>
</thead>
<tbody>
<tr>
<td>spring</td>
<td>$\log(d_{u,v}/\ell)$</td>
<td>$c_{rep}/d_{u,v}^2$</td>
</tr>
<tr>
<td>Fruch &amp; Rein</td>
<td>$d_{u,v}^2/\ell$</td>
<td>$\ell^2/d_{u,v}$</td>
</tr>
<tr>
<td>Force Atlas 2</td>
<td>$d_{u,v}$</td>
<td>$(1 + \text{deg}(u))(1 + \text{deg}(v))/d_{u,v}$</td>
</tr>
<tr>
<td>UMAP</td>
<td>$\frac{d_{u,v}^{2(b-1)}}{1 + d_{u,v}^2}\ell$</td>
<td>$\frac{1 - \ell}{(\epsilon + d_{u,v}^2)(1 + ad_{u,v}^{2b})}$</td>
</tr>
<tr>
<td>Method</td>
<td>UMAP</td>
<td>t-SNE</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
What is HDBSCAN

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

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

What is c-TF-IDF

\[ R_{w,c} = \frac{N_{w,c}}{\sum_w N_{w,c}} \cdot \log \left( \frac{\sum_d N_{w,d}}{\sum_d 1} \right), \quad N_{w,c} = \sum_{d \in D_c} N_{w,d} \]

- **Word**
- **Cluster (of documents)**
- **Document**
- **Probability of word w in cluster c**
- **Probability of appearance of word w in documents**

Words which occur in many documents are rated less important.
Intertopic distance map

model.visualize_topics()
Topic similarity

model.visualize_heatmap()
Visualize documents

topic_model.visualize_documents()
Questions ?