1 Graph layout

Force-directed methods use analogies from physics to compute the positions of a graph's vertices. We consider an undirected graph $G = (V, E)$ as a system of bodies (V), influenced by forces (E). The layout algorithm's goal is to find a configuration of those bodies where the sum of all forces is minimized, i.e. it assigns every vertex to a position such that the sum of forces which influence this vertex is 0.

In general, force-directed layouts consist of two parts: a *model* and an *algorithm*.

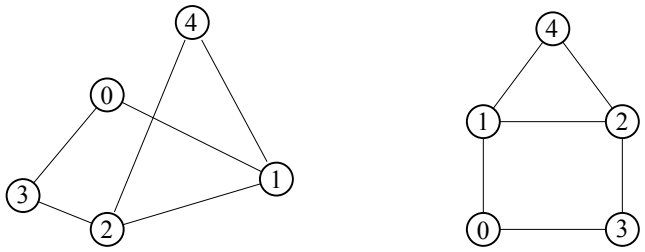

Figure 1: Example

2 Model: spring layout

The model in the spring layout uses a combination of springs and electrical forces. The edges correspond to springs. The vertices correspond to equally charged bodies which experience mutually repulsive forces in between them.

In the following, we denote vertices by $u, v \in V$, and by $\vec{u}, \vec{v} \in \mathbb{R}^2$ their position vectors in the two-dimensional space. The length of the difference vector $\vec{v} - \vec{u}$ is denoted by

$$||\vec{v} - \vec{u}|| = \sqrt{(x_v - x_u)^2 + (y_v - y_u)^2},$$

i.e. the euclidian distance of the points (x_u, y_u) and (x_v, y_v). Moreover, $\vec{e}_{uv}$ denotes the unit vector

$$\vec{e}_{uv} = \frac{\vec{v} - \vec{u}}{||\vec{v} - \vec{u}||}$$

which points from $\vec{u}$ to $\vec{v}$.

The repulsive force that v experiences from u obeys the equation[1]

$$\vec{F}_0(u, v) = \frac{c_0}{||\vec{v} - \vec{u}||^2} \cdot \vec{e}_{uv},$$

where $c_0 \in \mathbb{R}$ is a constant which determines the strength of the repulsive force at either one of the vertices u or v.

[1] Coulomb's law: $\vec{F} = \frac{1}{4\pi\epsilon_0} \cdot \frac{q_1 \cdot q_2}{r^2} \cdot \vec{e}_r$

The spring force experienced by v via the spring from u can be computed by the equation[2]

$$\vec{F}_1(u,v) = -c_1(||\vec{v} - \vec{u}|| - l) \cdot \vec{e}_{uv},$$

where the constant $c_1 \in \mathbb{R}$ denotes the strength of the spring and $l \in \mathbb{R}$ denotes its natural length.
The total force that is acting upon a vertex $v \in V$, can be computed as follows:

$$\vec{F}(v) = \sum_{u \in V} \vec{F}_0(u,v) + \sum_{(u,v) \in E} \vec{F}_1(u,v).$$

3 Algorithm

Vertices, which are not yet at equilibrium of forces, experience a force. To relax the system, vertices are iteratively moved towards the direction of their influencing force. At time t, a vertex v experiences the force $\vec{F}_t(v)$. After these forces are computed for all vertices at time t, each vertex v is moved by $\delta \cdot \vec{F}_t(v)$ in its respective direction. The constant $\delta,\ 0 < \delta < 1$, avoids that the vertices are moved too far.

Algorithm: spring embedder

```
WHILE (...)  DO
    FOR v ∈ V DO
        F⃗(v) = Σ_{u∈V} F⃗_0(u,v) + Σ_{(u,v)∈E} F⃗_1(u,v)
    FOR v ∈ V DO
        v⃗ ← v⃗ + δ · F⃗(v)
OD
```

3.1 Termination condition

The simplest option to terminate the algorithm is to stop after a certain number of iterations. Another option is to compute the global force of the system at a time t. This can be done as follows:

$$F_{\text{global}} = \sum_{v \in V} |\vec{F}_t(v)|.$$

If F_{global} reaches/falls below a certain value (ideally, $F_{\text{global}} = 0$), then the computation can be stopped.

4 Computation speed-up: clipping

In order to compute the force $\vec{F}(v)$ of a vertex $v \in V$, the sum $\sum_{u \in V} \vec{F}_0(v,u)$ is computed. This computation is done $n(n-1) = O(n^2)$ times per iteration. Vertices, which

[2] Hooke's law: $\vec{F} = -D \cdot \vec{s}$

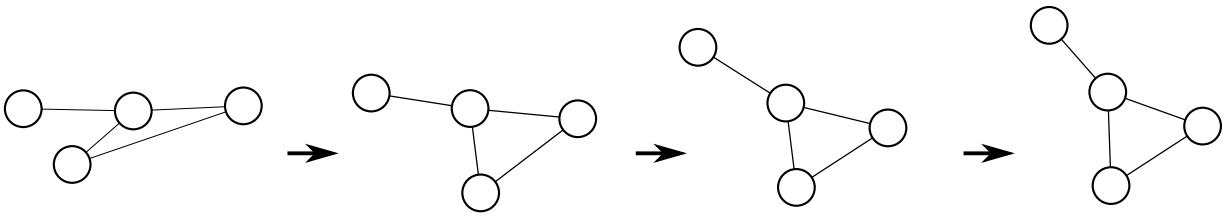

Figure 2: Process of the spring embedder algorithm

are far apart, only experience a very weak repulsive force from each other. Because of this, one can neglect these distant vertices in the computation of the repulsive force. The force $\vec{F_0}(v,u)$ is computed only for vertices u whose distance to v is $||\vec{v}-\vec{u}||^2 < d^2$ (for a proper constant $d \in \mathbb{R}$).

5 Non-connected graphs

The given algorithm only works for connected graphs G. If G contains several connected components $Z_1, \ldots, Z_n$, the components Z_i drift apart in every iteration and no equilibrium of forces is attained. This problem can be solved in two different ways:

- Insert another vertex w and either connect it to *all other vertices* $v \in V$, or connect it to *one vertex* v_i *in each connected component* Z_i. After termination of the algorithm, delete w and all its edges.
- Assign a charge to the border of the canvas such that vertices experience a repulsive force from it.

6 Computation

Instead of doing the calculations with vectors, it may be easier to divide the forces into the x and y components, which can be calculated separately. For example, the force F_0 in the x direction is

$$F_{0,x}(u,v) = \frac{c_0}{(x_v - x_u)^2 + (y_v - y_u)^2} \cdot \frac{x_v - x_u}{\sqrt{(x_v - x_u)^2 + (y_v - y_u)^2}}.$$

Analogously, values for the forces $F_{0,y}, F_{1,x}$ and $F_{1,y}$ can be computed.

References

[1] Ulrik Brandes. Drawing on Physical Analogies. Drawing Graphs. LNCS 2025: 71–86, 2001.

[2] Guiseppe di Battista and Peter Eades and Roberto Tamassia and Ioannis G. Tollis. Graph Drawing. Algorithms for the Visualizaition of Graphs. 303–325.