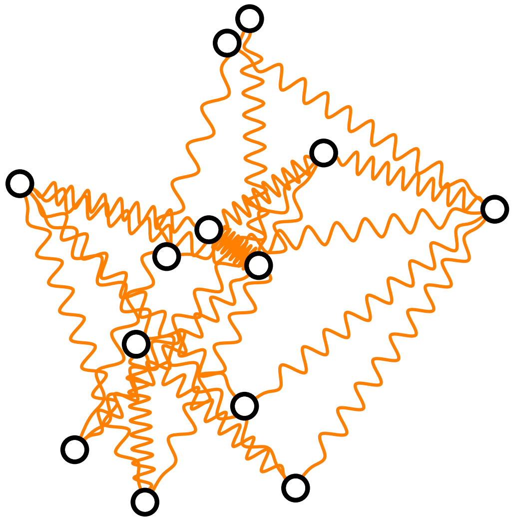
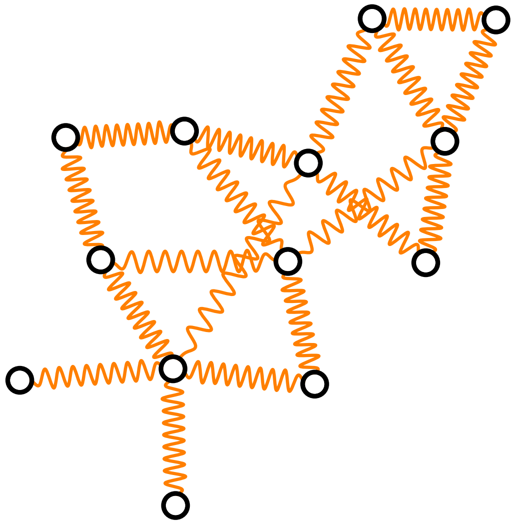
Visualization of Graphs

Lecture 2:
Force-Directed Drawing Algorithms



Part I:
Spring Embedders

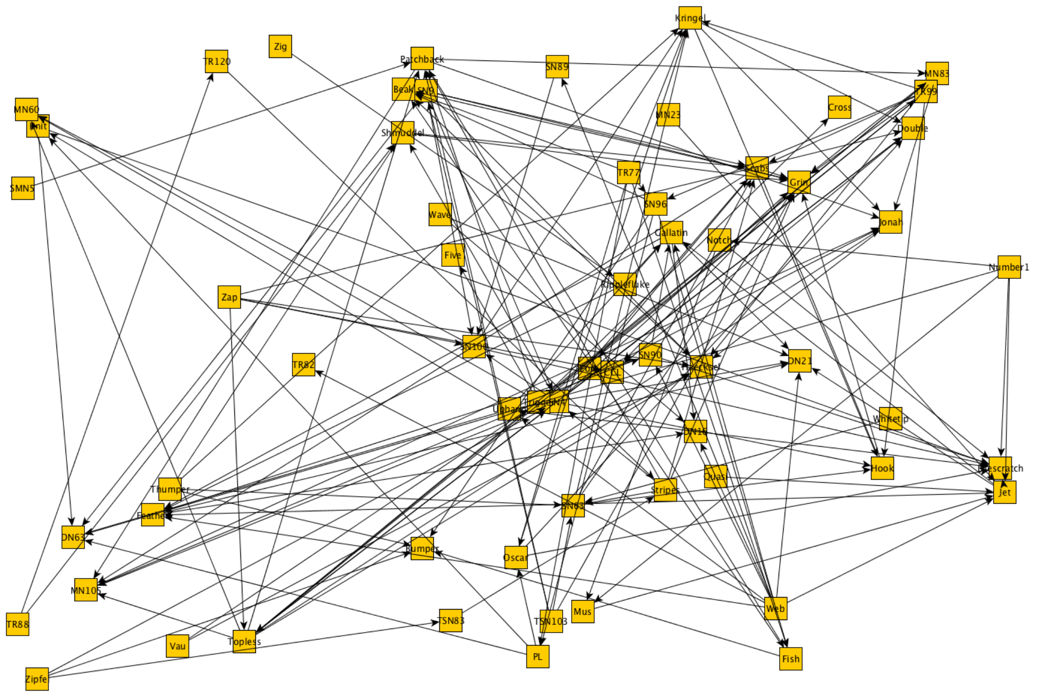


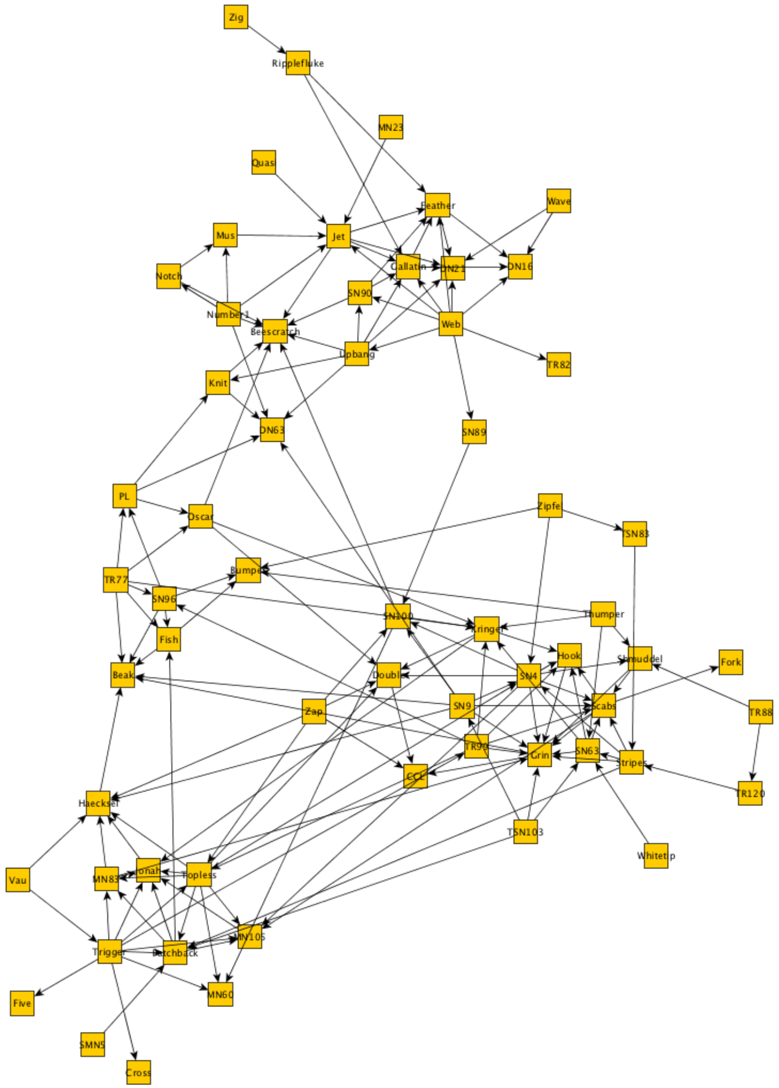
Johannes Zink

General Layout Problem

Input: Graph G

Output: Clear and readable straight-line drawing of G





General Layout Problem

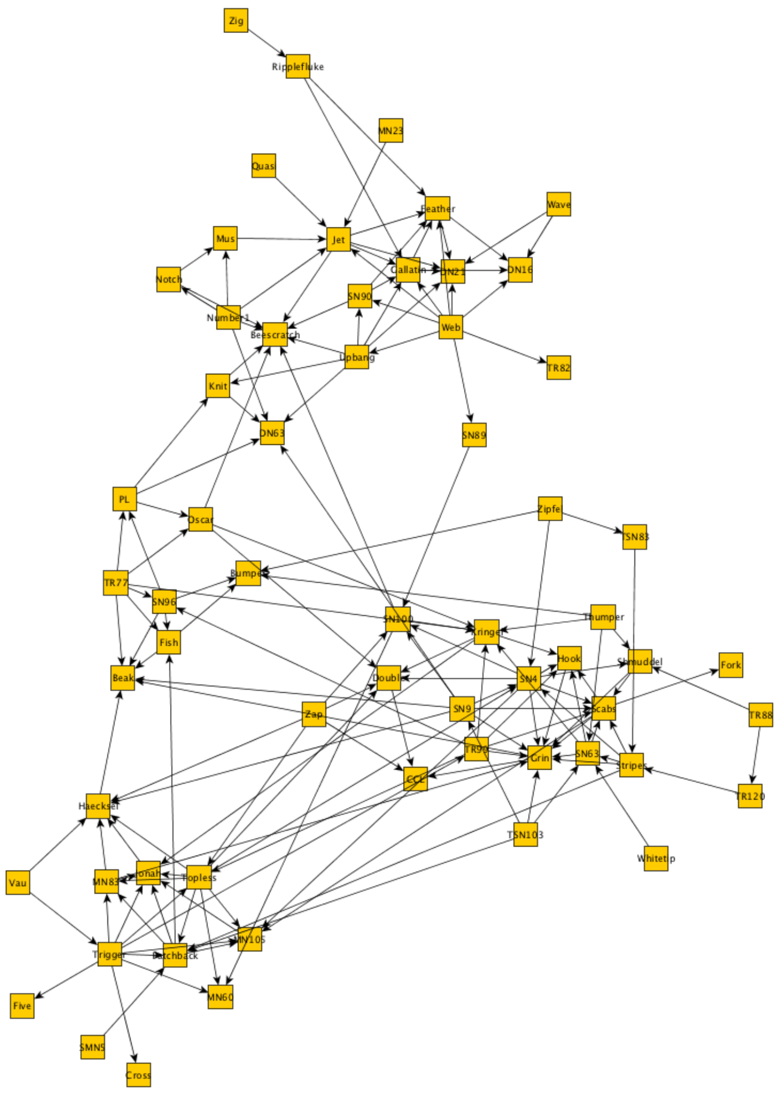
Input: Graph G

Output: Clear and readable straight-line drawing of G

Drawing aesthetics to optimize:

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

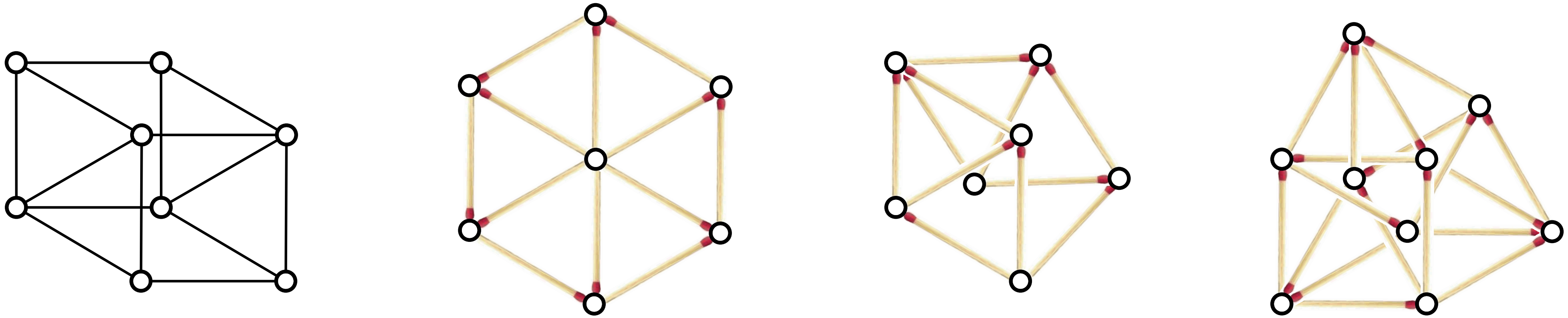
Optimization criteria partially contradict each other.



Fixed Edge Lengths?

Input: Graph $G = (V, E)$, required edge length $\ell(e)$ for each $e \in E$.

Output: Drawing of G that realizes the given edge lengths.



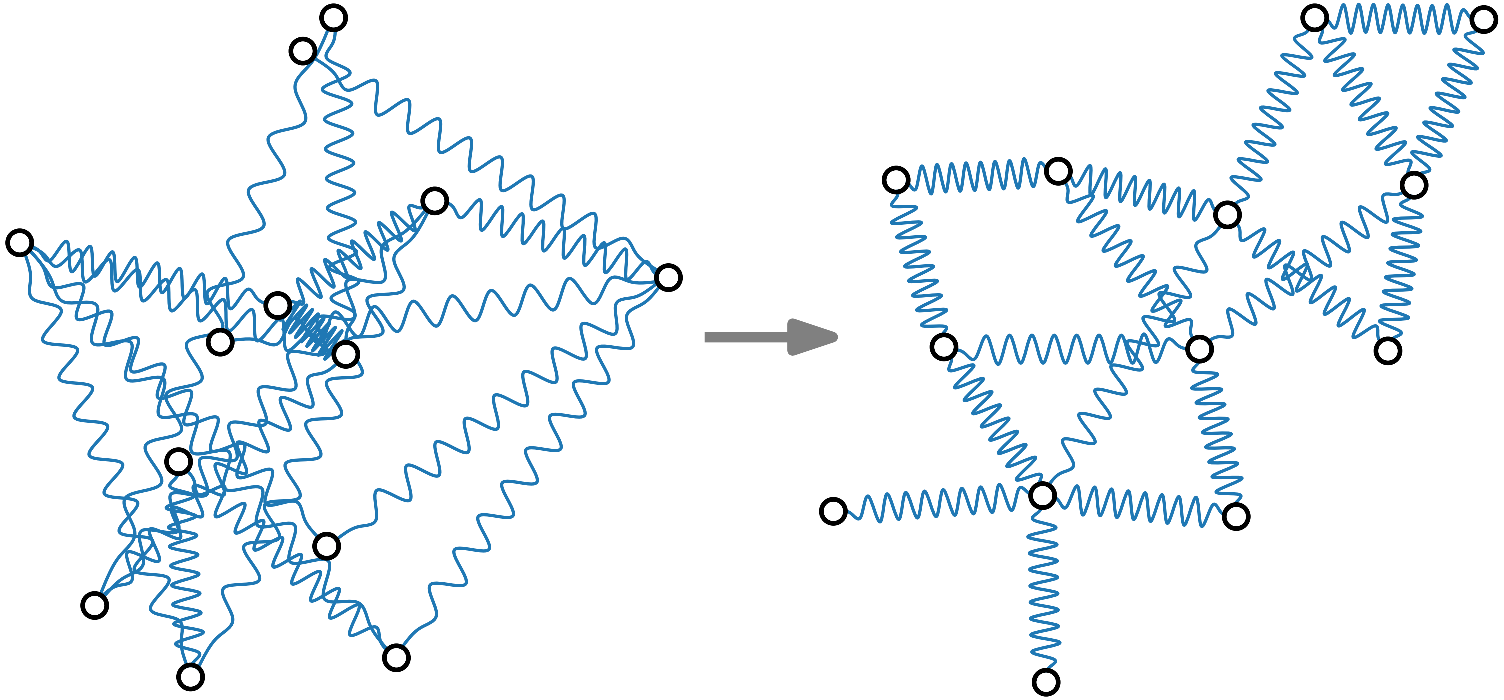
NP-hard for

- uniform edge lengths in any dimension [Johnson '82]
- uniform edge lengths in planar drawings [Eades, Wormald '90]
- edge lengths $\{1, 2\}$ [Saxe '80]

Physical Analogy

Idea. [Eades '84]

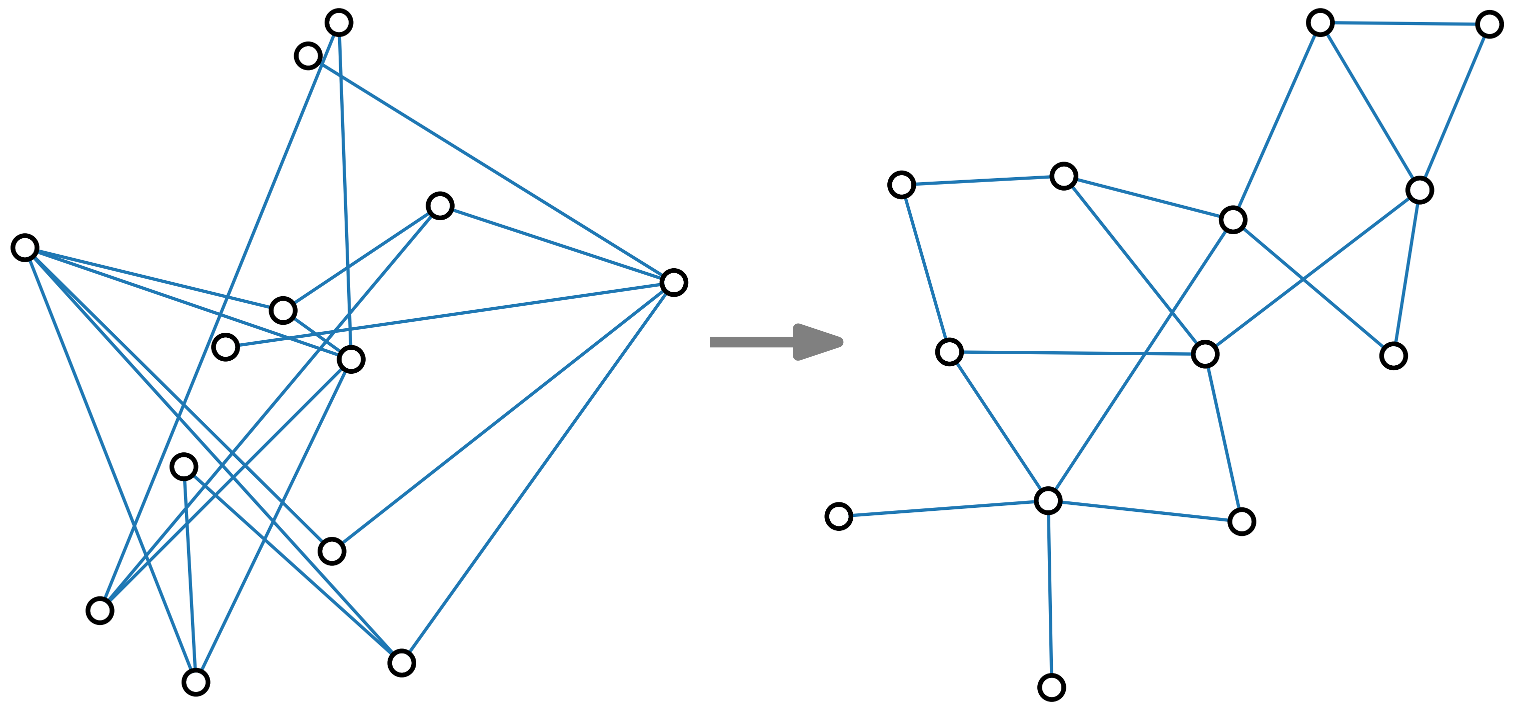
"To embed a graph we replace the vertices by steel rings and replace each edge with a **spring** to form a mechanical system . . . The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state."



Physical Analogy

Idea. [Eades '84]

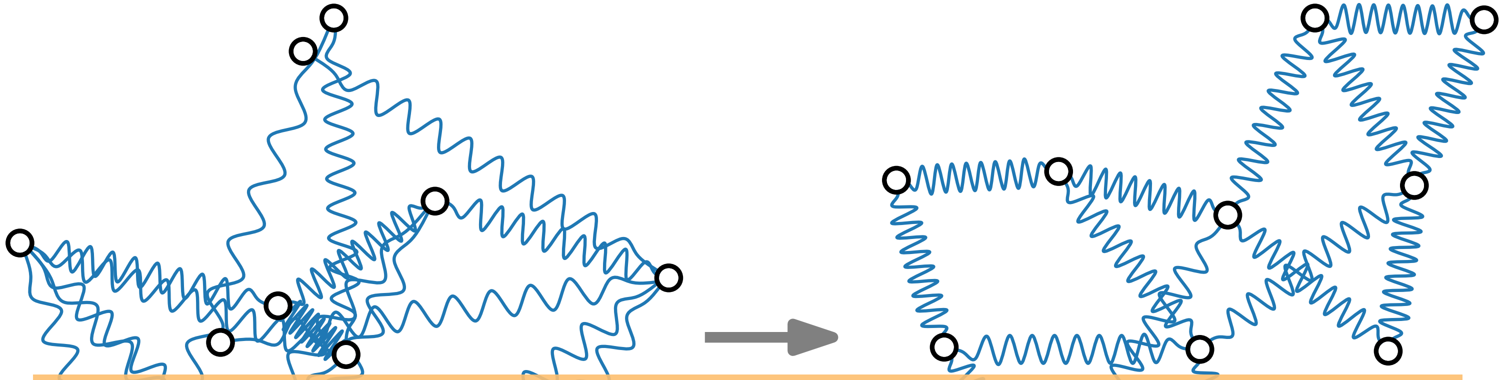
"To embed a graph we replace the vertices by steel rings and replace each edge with a **spring** to form a mechanical system . . . The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state."



Physical Analogy

Idea. [Eades '84]

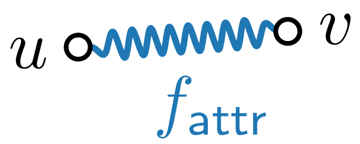
"To embed a graph we replace the vertices by steel rings and replace each edge with a **spring** to form a mechanical system . . . The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state."



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

Attractive forces.

pairs $\{u, v\}$ of adjacent vertices:



Repulsive forces.

any pair $\{x, y\}$ of vertices:

Force-Directed Algorithms

initial layout; may be randomly chosen positions

threshold

max # iterations

ForceDirected($G = (V, E)$, $p = (p_v)_{v \in V}$, $\varepsilon > 0$, $K \in \mathbb{N}$)

$t \leftarrow 1$

while $t < K$ **and** $\max_{v \in V} \|F_v(t)\| > \varepsilon$ **do**

  foreach $u \in V$ **do**

    $F_u(t) \leftarrow \sum_{v \in V} f_{\mathsf{rep}}(u, v) + \sum_{v \in \mathsf{Adj}[u]} f_{\mathsf{attr}}(u, v)$

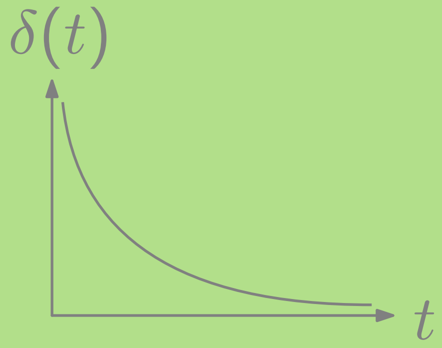
  foreach $u \in V$ **do**

    $p_u \leftarrow p_u + \delta(t) \cdot F_u(t)$

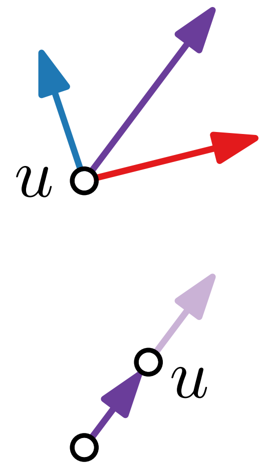
  $t \leftarrow t + 1$

return p

cooling factor



end layout



Spring Embedder by Eades – Model

```
ForceDirected(G = (V, E), p = (p_v)_{v∈V}, ε > 0, K ∈ ℕ)
  t ← 1
  while t < K and max_{v∈V} ||F_v(t)|| > ε do
      foreach u ∈ V do
          F_u(t) ← Σ_{v∈V} f_rep(u, v) + Σ_{v∈Adj[u]} f_attr(u, v)
      foreach u ∈ V do
          p_u ← p_u + δ(t) · F_u(t)
      t ← t + 1
  return p
```

Repulsive forces

repulsion constant (e.g., 2.0)

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

Attractive forces

spring constant (e.g., 1.0)

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

$$f_{\text{attr}}(u,v) = f_{\text{spring}}(u,v) - f_{\text{rep}}(u,v)$$

Resulting displacement vector

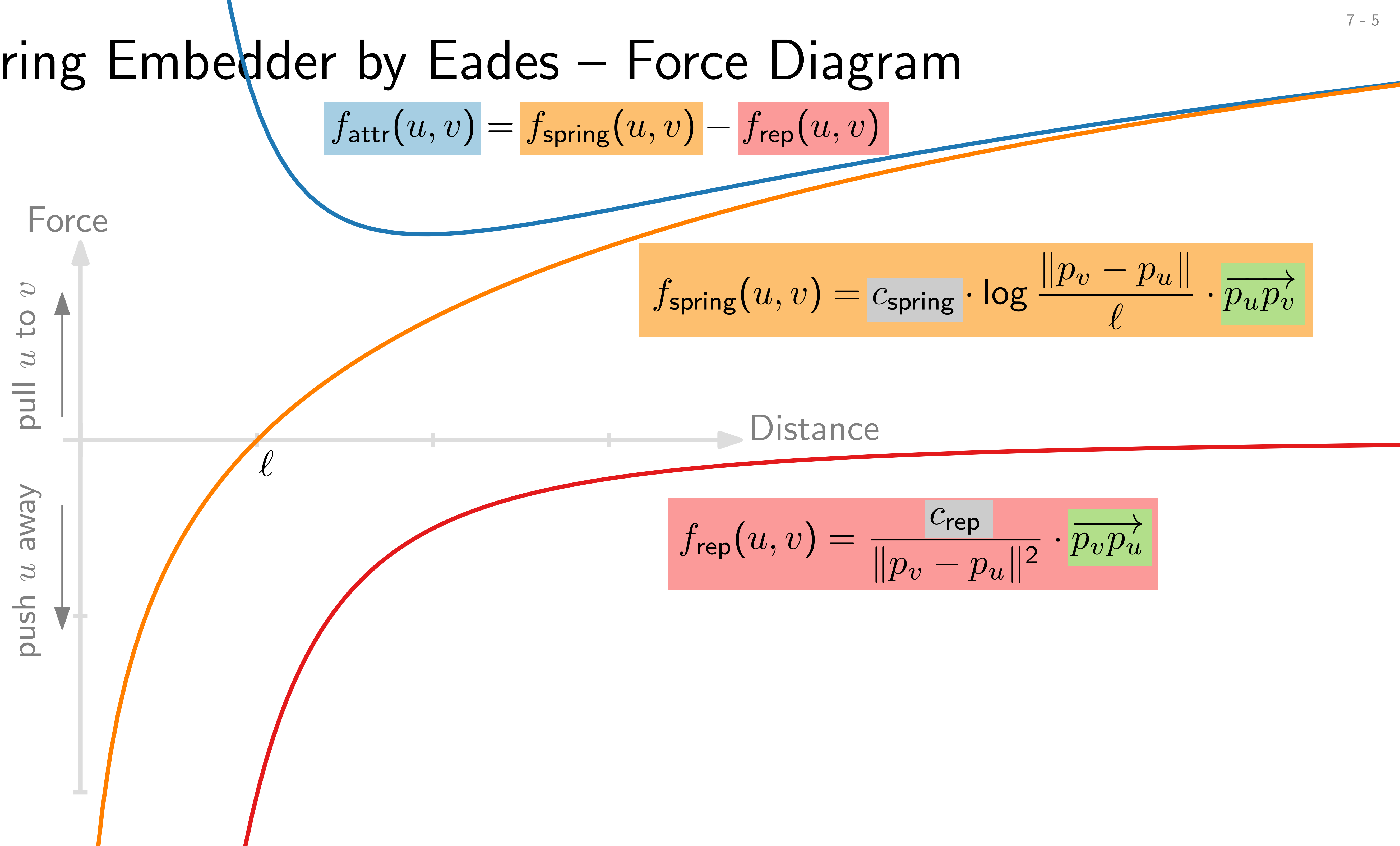
$$F_u = \sum_{v \in V} f_{\text{rep}}(u,v) + \sum_{v \in \text{Adj}[u]} f_{\text{attr}}(u,v)$$

Notation.

- $\overrightarrow{p_u p_v}$ = unit vector pointing from u to v
- $\|p_v - p_u\|$ = Euclidean distance between u and v
- ℓ = ideal spring length for edges

Spring Embedder by Eades – Force Diagram

$$f_{\text{attr}}(u,v) = f_{\text{spring}}(u,v) - f_{\text{rep}}(u,v)$$



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

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

Spring Embedder by Eades – Discussion

Advantages.

- very simple algorithm
- good results for small and medium-sized graphs
- empirically good representation of symmetry and structure

Disadvantages.

- System may not be stable at the end.
- Converges to local minima.
- Computing f_{spring} is in $\mathcal{O}(|E|)$ time and computing f_{rep} is in $\mathcal{O}(|V|^2)$ time.

Influence.

- original paper by Peter Eades [Eades '84] got $\approx$ 2000 citations
- basis for many further ideas

Variant by Fruchterman & Reingold

- **Repulsive forces**

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

- **Attractive forces**

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

- **Resulting displacement vector**

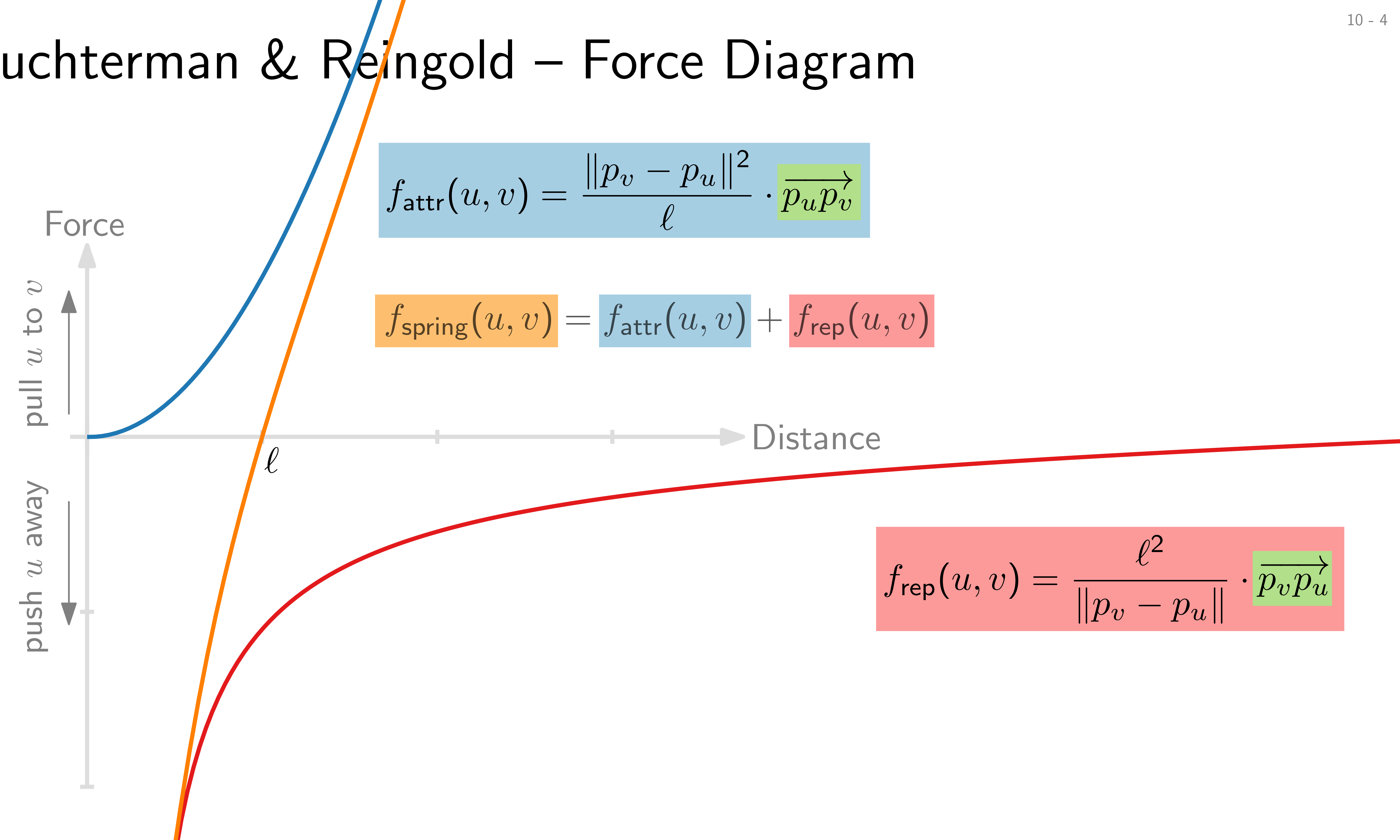
$$F_u = \sum_{v \in V} f_{\text{rep}}(u,v) + \sum_{v \in \text{Adj}[u]} f_{\text{attr}}(u,v)$$

ForceDirected($G = (V,E)$, $p = (p_v)_{v \in V}$, $\varepsilon > 0$, $K \in \mathbb{N}$)
 $t \leftarrow 1$
 while $t < K$ **and** $\max_{v \in V} \|F_v(t)\| > \varepsilon$ **do**
  foreach $u \in V$ **do**
   $F_u(t) \leftarrow \sum_{v \in V} f_{\text{rep}}(u,v) + \sum_{v \in \text{Adj}[u]} f_{\text{attr}}(u,v)$
  foreach $u \in V$ **do**
   $p_u \leftarrow p_u + \delta(t) \cdot F_u(t)$
  $t \leftarrow t + 1$
 return p

Notation.

- $\|p_u - p_v\|$ = Euclidean distance between u and v
- $\overrightarrow{p_u p_v}$ = unit vector pointing from u to v
- ℓ = ideal spring length for edges

Fruchterman & Reingold – Force Diagram



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

$$f_{\text{spring}}(u,v) = f_{\text{attr}}(u,v) + f_{\text{rep}}(u,v)$$

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

Adaptability

Inertia. ("Trägheit")

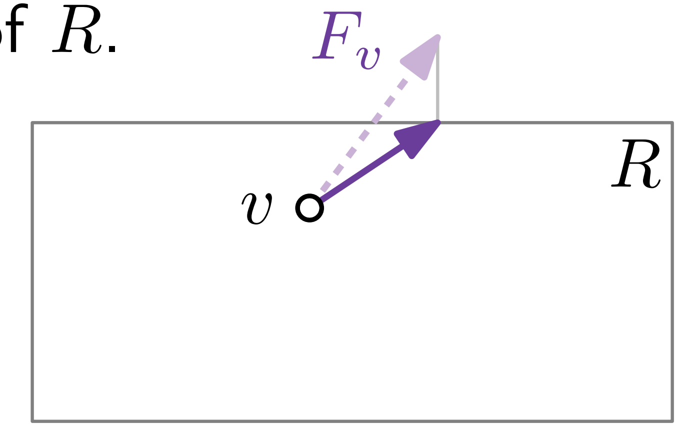
- Define vertex mass $\Phi(v) = 1 + \deg(v)/2$
- Set $f_{\text{attr}}(p_u, p_v) \leftarrow f_{\text{attr}}(p_u, p_v) \cdot 1/\Phi(v)$

Gravitation.

- Define centroid $\sigma_V = 1/|V| \cdot \sum_{v \in V} p_v$
- Add force $f_{\text{grav}}(p_v) = c_{\text{grav}} \cdot \Phi(v) \cdot \overrightarrow{p_v \sigma_V}$

Restricted drawing area.

If F_v points beyond area R, clip vector appropriately at the border of R.



And many more...

- magnetic orientation of edges [GD Ch. 10.4]
- other energy models
- planarity preserving
- speed-ups

Speeding up "Convergence" by Adaptive Displacement $\delta_v(t)$

ForceDirected($G = (V, E)$, $p = (p_v)_{v \in V}$, $\varepsilon > 0$, $K \in \mathbb{N}$)

$t \leftarrow 1$

while $t < K$ **and** $\max_{v \in V} \|F_v(t)\| > \varepsilon$ **do**

  foreach $u \in V$ **do**

    $F_u(t) \leftarrow \sum_{v \in V} f_{\mathsf{rep}}(u, v) + \sum_{v \in \mathsf{Adj}[u]} f_{\mathsf{attr}}(u, v)$

  foreach $u \in V$ **do**

    $p_u \leftarrow p_u + \cancel{\delta(t)} \cdot F_u(t)$

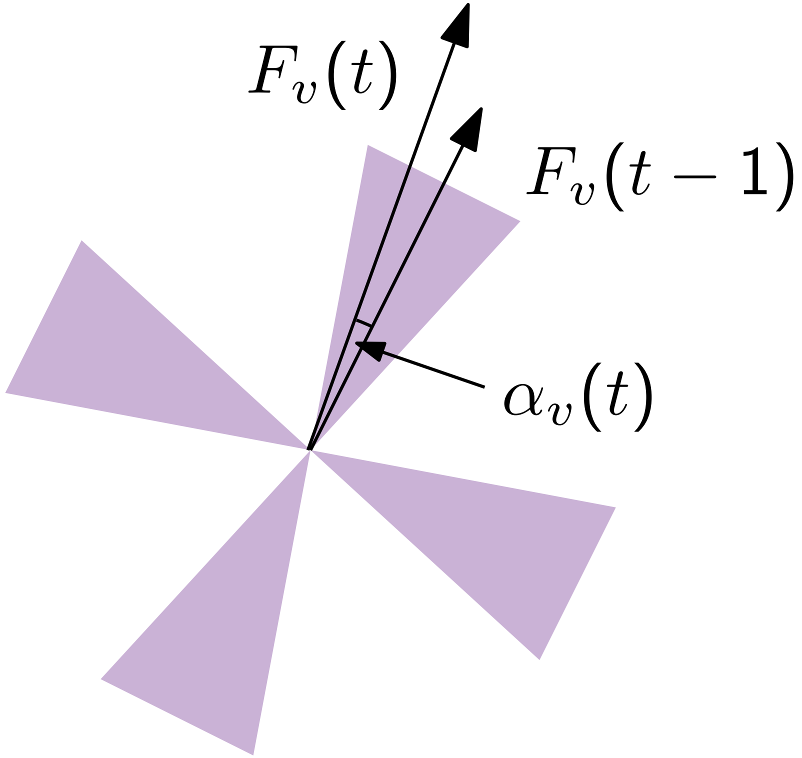
    $\delta_v(t)$

  $t \leftarrow t + 1$

return p

Speeding up "Convergence" by Adaptive Displacement $\delta_v(t)$

[Frick, Ludwig, Mehldau '95]

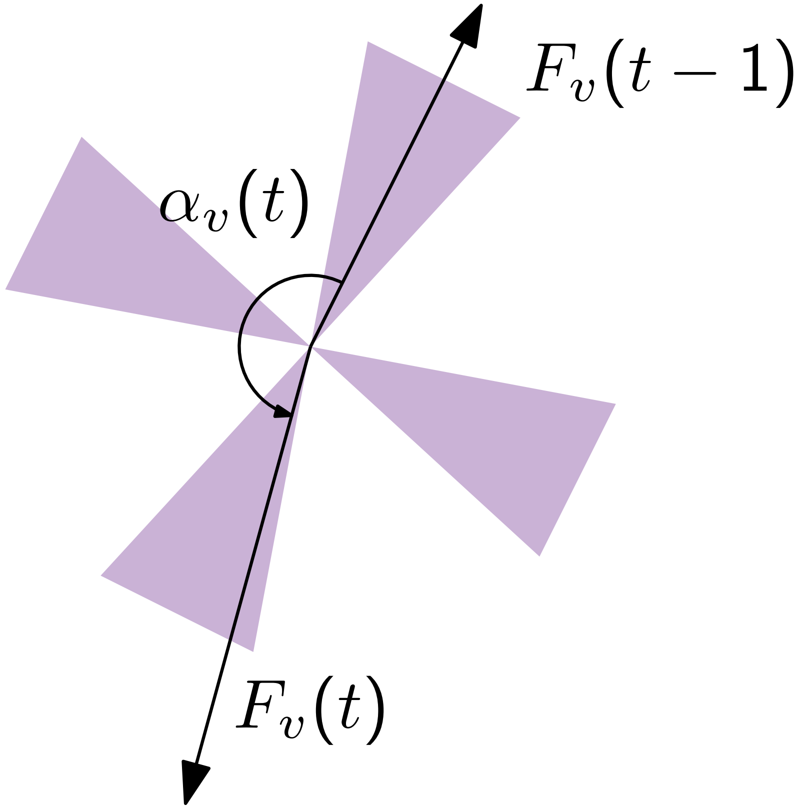


Same direction.

$\rightarrow$ increase temperature $\delta_v(t)$

Speeding up “Convergence” by Adaptive Displacement $\delta_v(t)$

[Frick, Ludwig, Mehldau ’95]

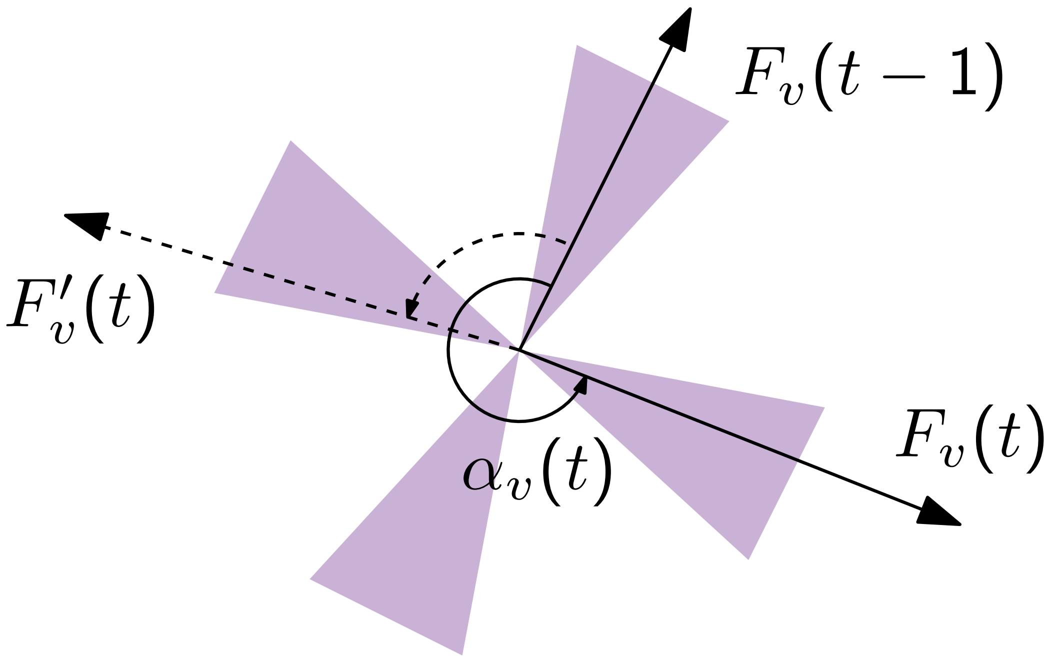


Same direction.
$\rightarrow$ increase temperature $\delta_v(t)$

Oscillation.
$\rightarrow$ decrease temperature $\delta_v(t)$

Speeding up "Convergence" by Adaptive Displacement $\delta_v(t)$

[Frick, Ludwig, Mehldau '95]



Same direction.
$\rightarrow$ increase temperature $\delta_v(t)$

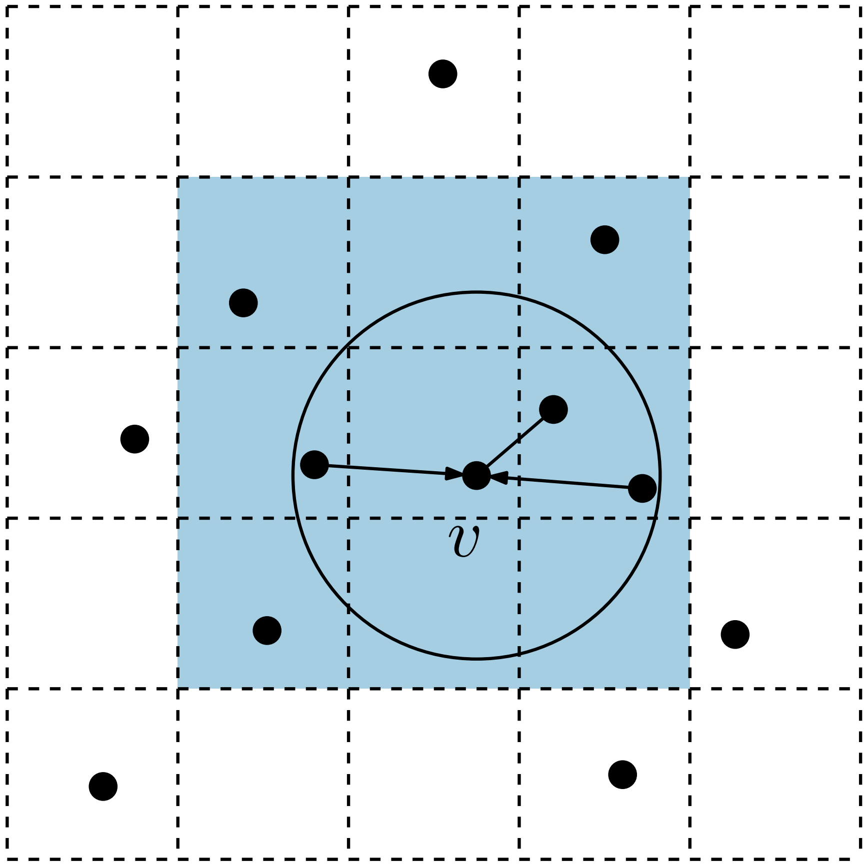
Oscillation.
$\rightarrow$ decrease temperature $\delta_v(t)$

Rotation.
- count rotations
- if applicable

$\rightarrow$ decrease temperature $\delta_v(t)$

Speeding up "Convergence" via Grids

[Fruchterman & Reingold '91]



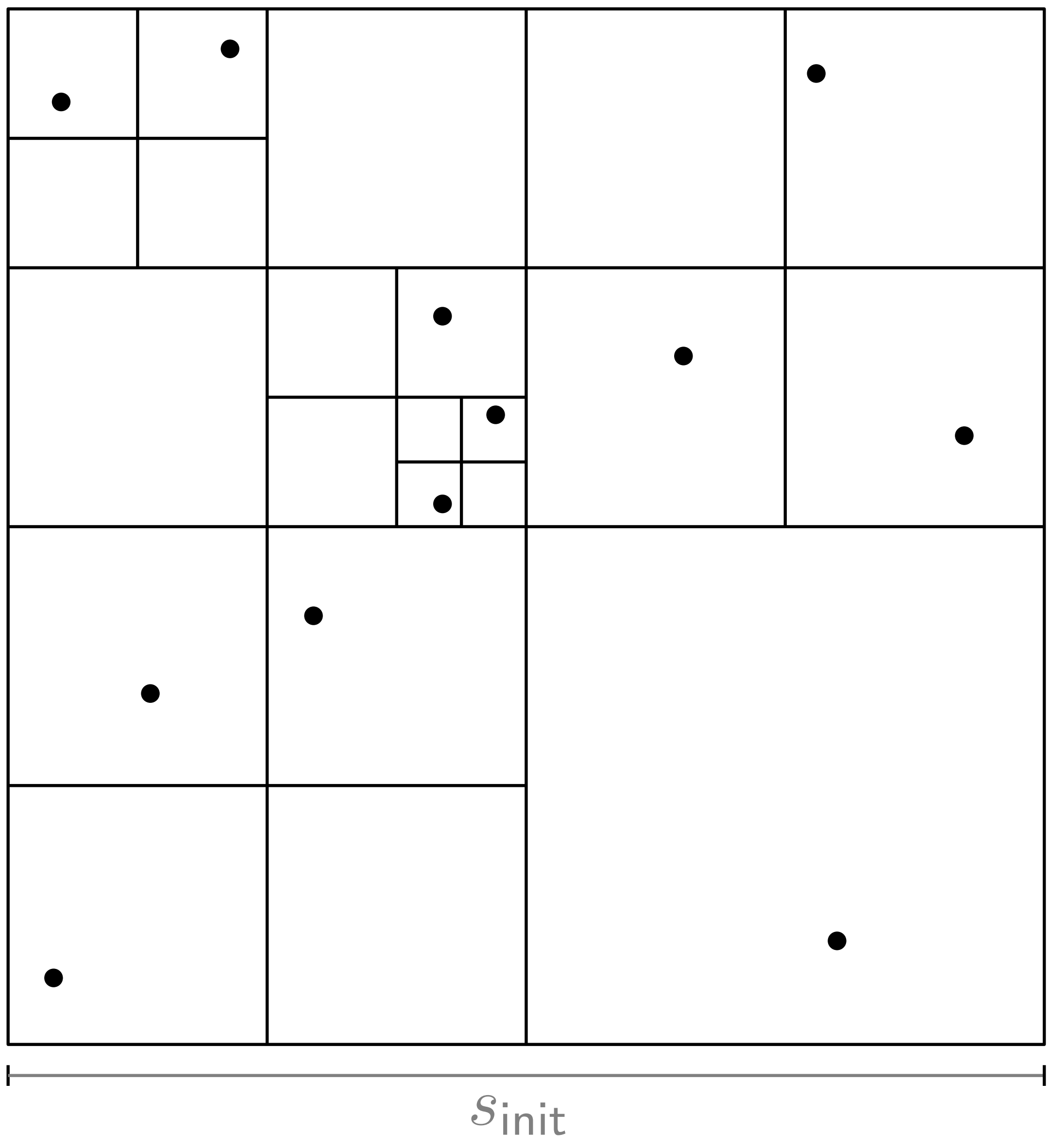
- divide plane into a grid
- consider repulsive forces only to vertices in neighboring cells
- and only if the distance is less than some threshold

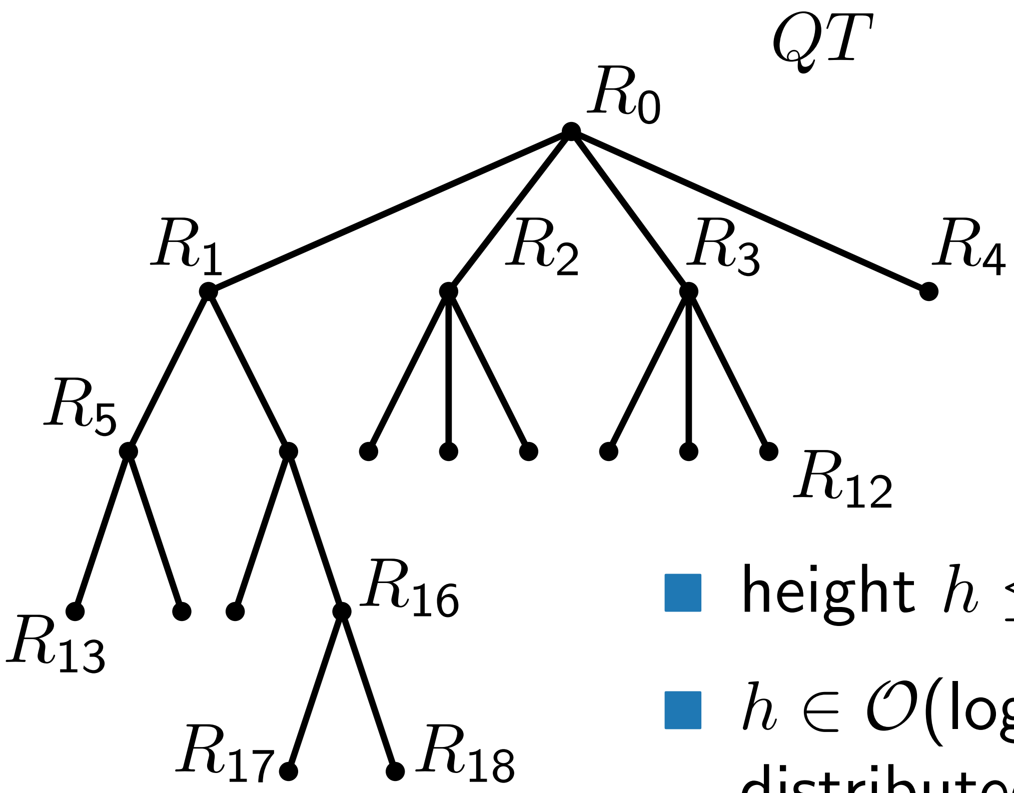
Discussion.

- good idea to improve actual runtime
- asymptotic runtime does not improve
- might introduce oscillation and thus a quality loss

Speeding up with Quad Trees

[Barnes, Hut '86]

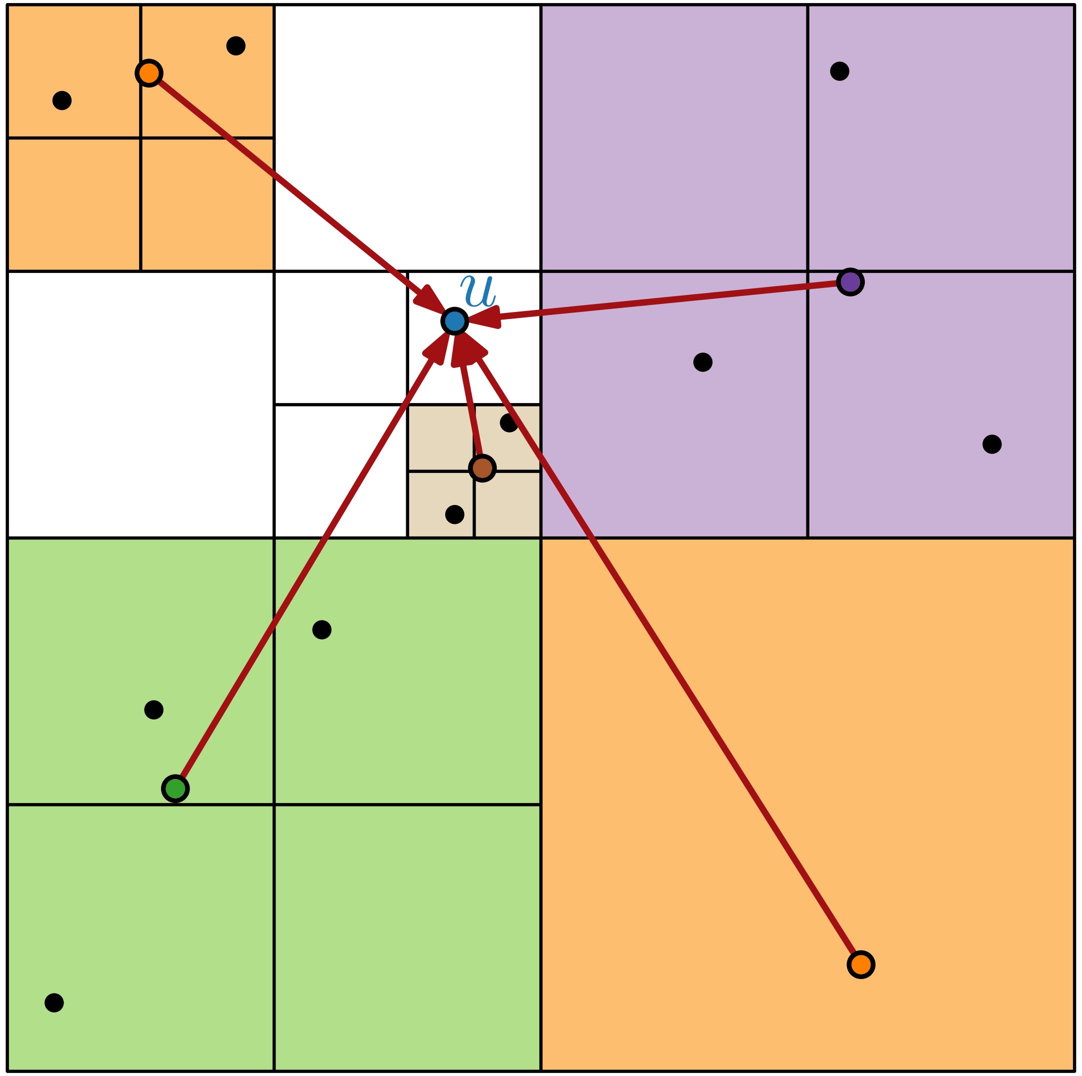


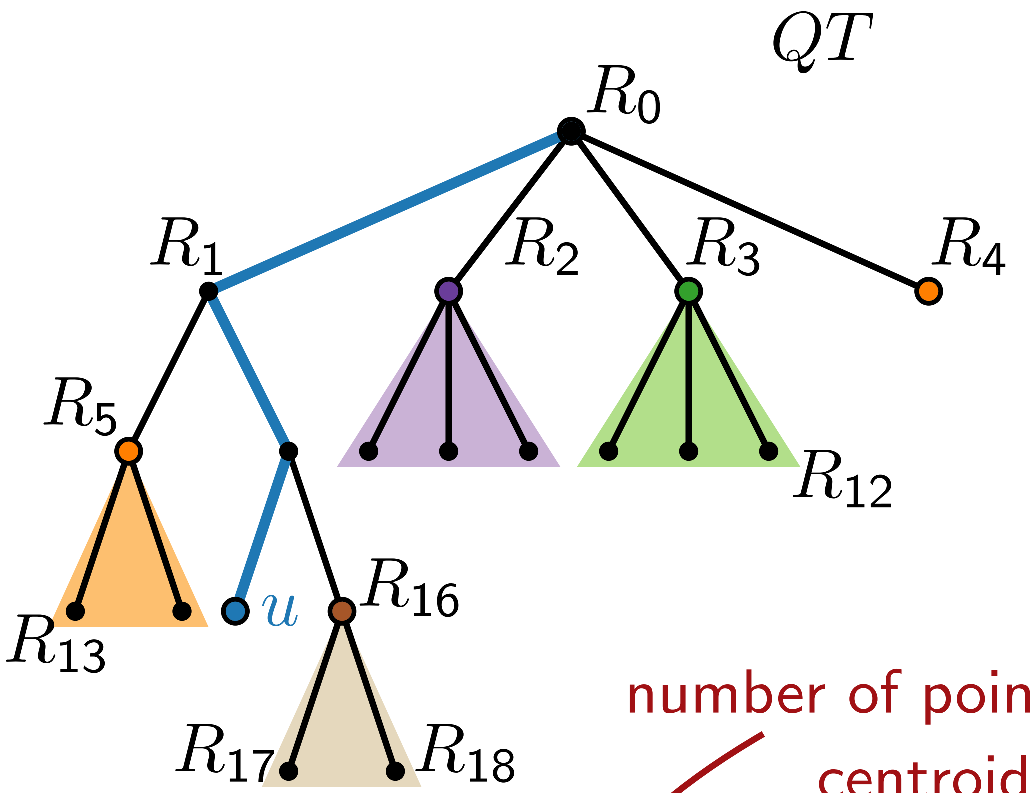


- height $h \leq \log \frac{s_{\text{init}}}{d_{\text{min}}} + \frac{3}{2}$
- $h \in \mathcal{O}(\log n)$ if vertices evenly distributed in the initial box
- time/space in $\mathcal{O}(hn)$
- compressed quad tree can be computed in $\mathcal{O}(n \log n)$ time

Speeding up with Quad Trees

[Barnes, Hut ’86]





number of points in the subtree R_i

centroid of R_i (pre-computed)

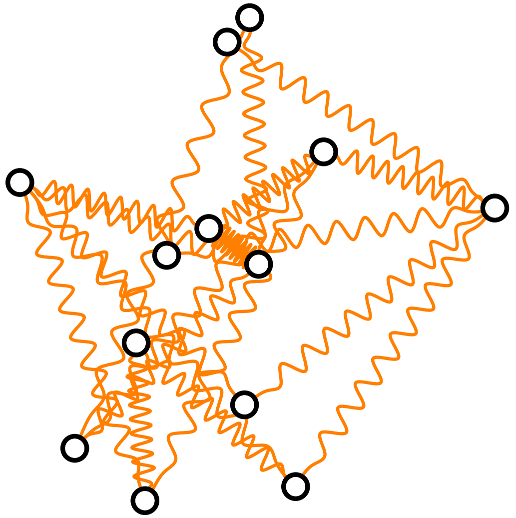
$$f_{\text{rep}}(R_i, p_u) = \boxed{|R_i|} \cdot f_{\text{rep}}(\boxed{\sigma_{R_i}}, p_u)$$

for each child R_i of a vertex on path from u to root.

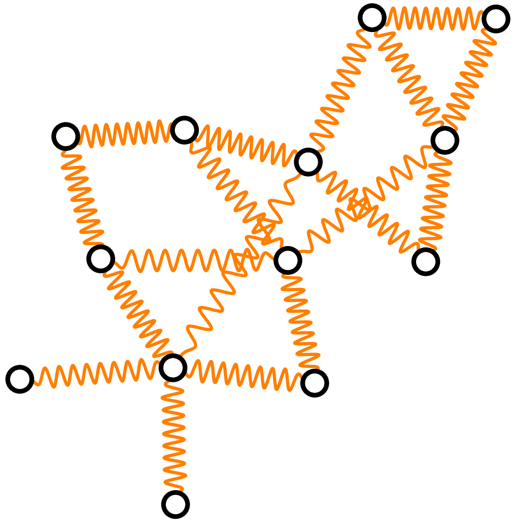
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Visualization of Graphs

Lecture 2:
Force-Directed Drawing Algorithms



Part II:
Tutte Embeddings

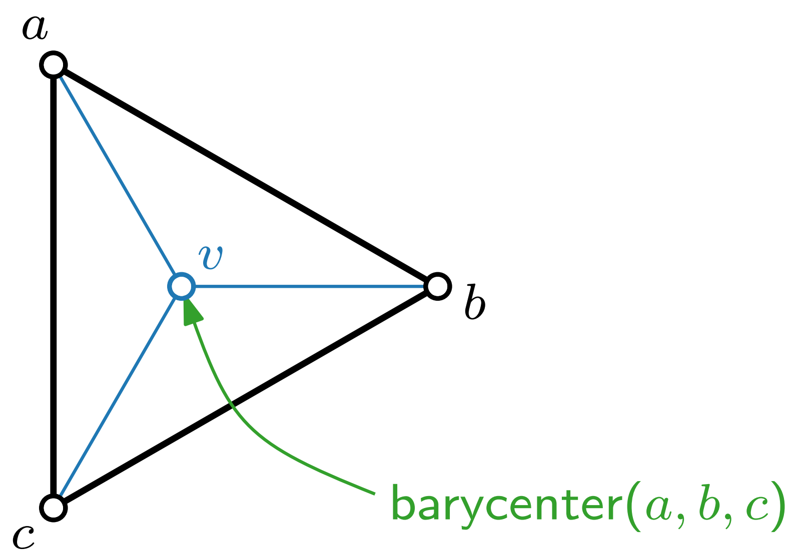


Johannes Zink

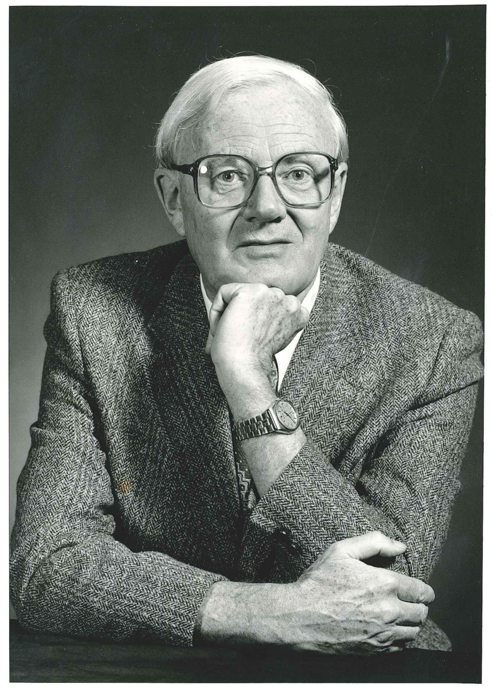
Idea

Consider a fixed triangle (a, b, c)
with a common neighbor v

Where would you place v?



barycenter$(x_1, \ldots, x_k) = \sum_{i=1}^{k} x_i / k$



William T. Tutte
1917 – 2002

Idea.
Repeatedly place every vertex at barycenter of neighbors.

Tutte's Forces

Goal.

$p_u = \text{barycenter}(\text{Adj}[u])$
$\quad = \sum_{v \in \text{Adj}[u]} p_v / \deg(u)$

$F_u(t) = \sum_{v \in \text{Adj}[u]} p_v / \deg(u) - p_u$
$\quad = \sum_{v \in \text{Adj}[u]} (p_v - p_u) / \deg(u)$
$\quad = \displaystyle\sum_{v \in \text{Adj}[u]} \frac{\|p_u - p_v\|}{\deg(u)} \overrightarrow{p_u p_v}$

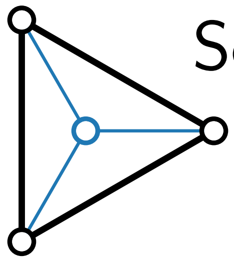
- **Repulsive forces** $\quad f_{\text{rep}}(u, v) = 0$
- **Attractive forces**

$$f_{\text{attr}}(u, v) = \begin{cases} 0 & \text{if } u \text{ fixed,} \\ \frac{\|p_u - p_v\|}{\deg(u)} \overrightarrow{p_u p_v} & \text{otherwise.} \end{cases}$$

ForceDirected($G = (V, E)$, $p = (p_v)_{v \in V}$, $\varepsilon > 0$, $K \in \mathbb{N}$)
$\quad t \leftarrow 1$
$\quad$ **while** $t < K$ **and** $\max_{v \in V} \|F_v(t)\| > \varepsilon$ **do**
$\quad\quad$ **foreach** $u \in V$ **do**
$\quad\quad\quad F_u(t) \leftarrow \sum_{v \in V} f_{\text{rep}}(u, v) + \sum_{v \in \text{Adj}[u]} f_{\text{attr}}(u, v)$
$\quad\quad$ **foreach** $u \in V$ **do**
$\quad\quad\quad p_u \leftarrow p_u + \cancel{\delta(t)}\; 1 \cdot F_u(t)$
$\quad\quad t \leftarrow t + 1$
$\quad$ **return** p

barycenter$(x_1, \ldots, x_k) = \sum_{i=1}^{k} x_i / k$

Global minimum: $p_u = (0, 0)\ \forall u \in V$



Solution: fix coordinates of outer face!

$\overrightarrow{p_u p_v}$ = unit vector pointing from u to v

$\|p_u - p_v\|$ = Euclidean distance between u and v

System of Linear Equations

Goal. $p_u = (x_u, y_u)$

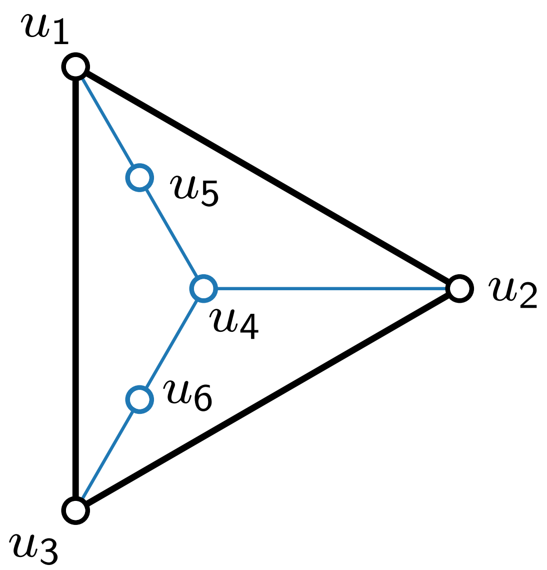
$p_u = \text{barycenter}(\text{Adj}[u]) \;= \sum_{v \in \text{Adj}[u]} p_v / \deg(u)$

$Ax = b \qquad Ay = b \qquad b = (0)_n$

Two systems of linear equations:

$$x_u = \sum_{v \in \text{Adj}[u]} x_v / \deg(u) \;\Leftrightarrow\; \deg(u) \cdot x_u = \sum_{v \in \text{Adj}[u]} x_v \;\Leftrightarrow\; \deg(u) \cdot x_u - \sum_{v \in \text{Adj}[u]} x_v = 0$$

$$y_u = \sum_{v \in \text{Adj}[u]} y_v / \deg(u) \;\Leftrightarrow\; \deg(u) \cdot y_u = \sum_{v \in \text{Adj}[u]} y_v \;\Leftrightarrow\; \deg(u) \cdot y_u - \sum_{v \in \text{Adj}[u]} y_v = 0$$



$$\begin{array}{c|c} & \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} & \begin{pmatrix} 3 & -1 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ 0 & -1 & 0 & 3 & -1 & -1 \\ -1 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 2 \end{pmatrix} \end{array} \quad A$$

Laplacian matrix of G

$A_{ii} = \deg(u_i)$

$$A_{ij, i \neq j} = \begin{cases} -1 & u_i u_j \in E \\ 0 & u_i u_j \notin E \end{cases}$$

n variables, n constraints, $\det(A) = 0$

$\Rightarrow$ no unique solution

System of Linear Equations

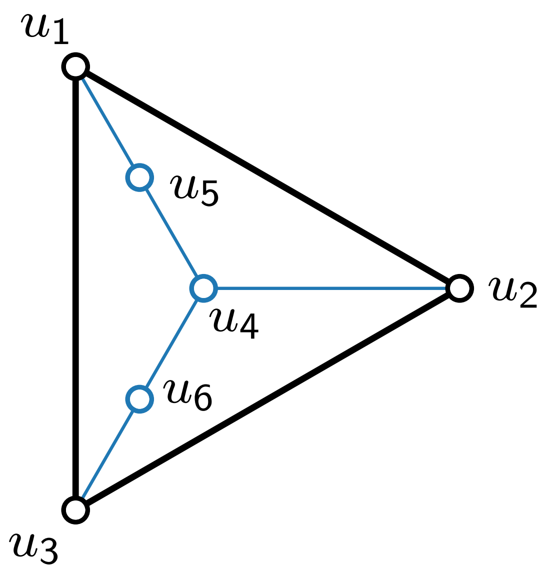
solve two systems of linear equations

Theorem. **Tutte drawing**
Tutte's barycentric algorithm admits a unique solution.
It can be computed in polynomial time.

Goal. $p_u = (x_u, y_u)$

$p_u = \text{barycenter}(\text{Adj}[u]) =$

$$x_u = \sum_{v \in \text{Adj}[u]} x_v / \deg(u) \Leftrightarrow \deg(u) \cdot x_u = \sum_{v \in \text{Adj}[u]} x_v \Leftrightarrow \deg(u) \cdot x_u - \sum_{v \in \text{Adj}[u]} x_v = 0$$
$$y_u = \sum_{v \in \text{Adj}[u]} y_v / \deg(u) \Leftrightarrow \deg(u) \cdot y_u = \sum_{v \in \text{Adj}[u]} y_v \Leftrightarrow \deg(u) \cdot y_u - \sum_{v \in \text{Adj}[u]} y_v = 0$$



$$A = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{pmatrix} 3 & -1 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ 0 & -1 & 0 & 3 & -1 & -1 \\ -1 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 2 \end{pmatrix} \end{array}$$

Laplacian matrix of G

$$A_{ii} = \deg(u_i)$$

$$A_{ij, i \neq j} = \begin{cases} -1 & u_i u_j \in E \\ 0 & u_i u_j \notin E \end{cases}$$

k variables, k constraints, $\det(A) > 0$

$k = \#$free vertices

$\Rightarrow$ unique solution ☺

Solution: we don't need to change the fixed vertices & constraints dependent on fixed vertices are constant and can be moved into b

3-Connected Planar Graphs

(up to the choice of the outer face and mirroring)

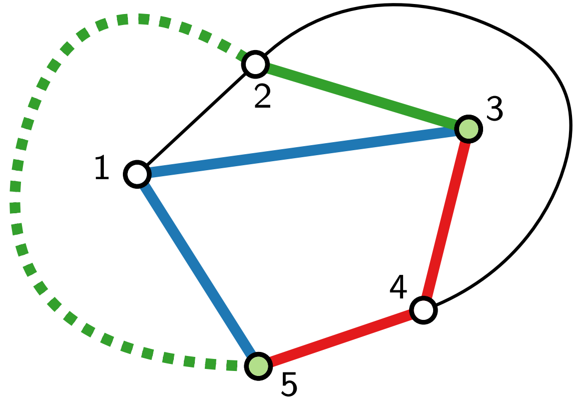
planar: G can be drawn in such a way that no edges cross each other

connected: $\exists$ u–v path for every vertex pair $\{u, v\}$.

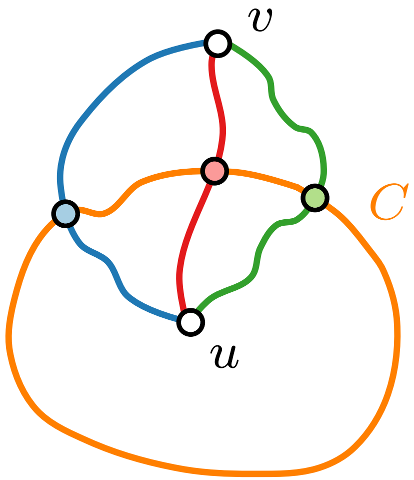
k-**connected:** $G - \{v_1, \dots, v_{k-1}\}$ is connected for **any** $k-1$ vertices $v_1 \dots, v_{k-1}$.
Or (equivalently if $G \neq K_k$):
There are at least k vertex-disjoint u–v paths for every vertex pair $\{u, v\}$.

Theorem. [Whitney 1933]
Every 3-connected planar graph has a unique planar embedding.

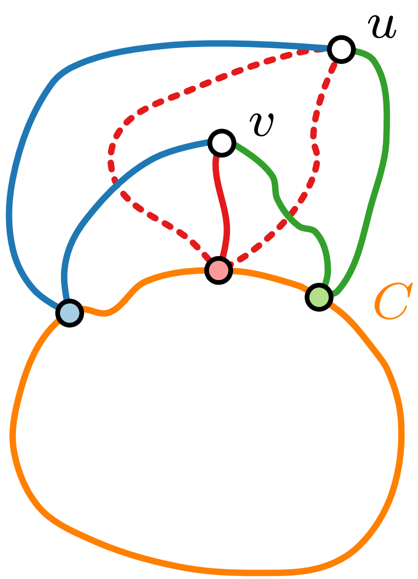
Proof sketch.
Γ_1, Γ_2 embeddings of G.
Let C be a face of Γ_2, but not of Γ_1.
u inside C in Γ_1, v outside C in Γ_1
both on same side in Γ_2



Γ_1

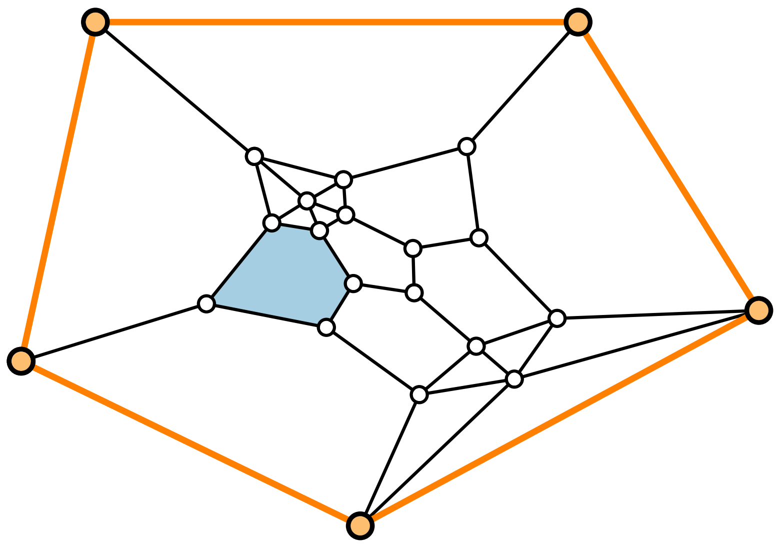


Γ_2



Tutte's Theorem

Theorem. [Tutte 1963]
Let G be a 3-connected planar graph, and
let C be a face of its unique embedding.
If we fix C on a strictly convex polygon, then the Tutte drawing of G is planar and all its faces are strictly convex.

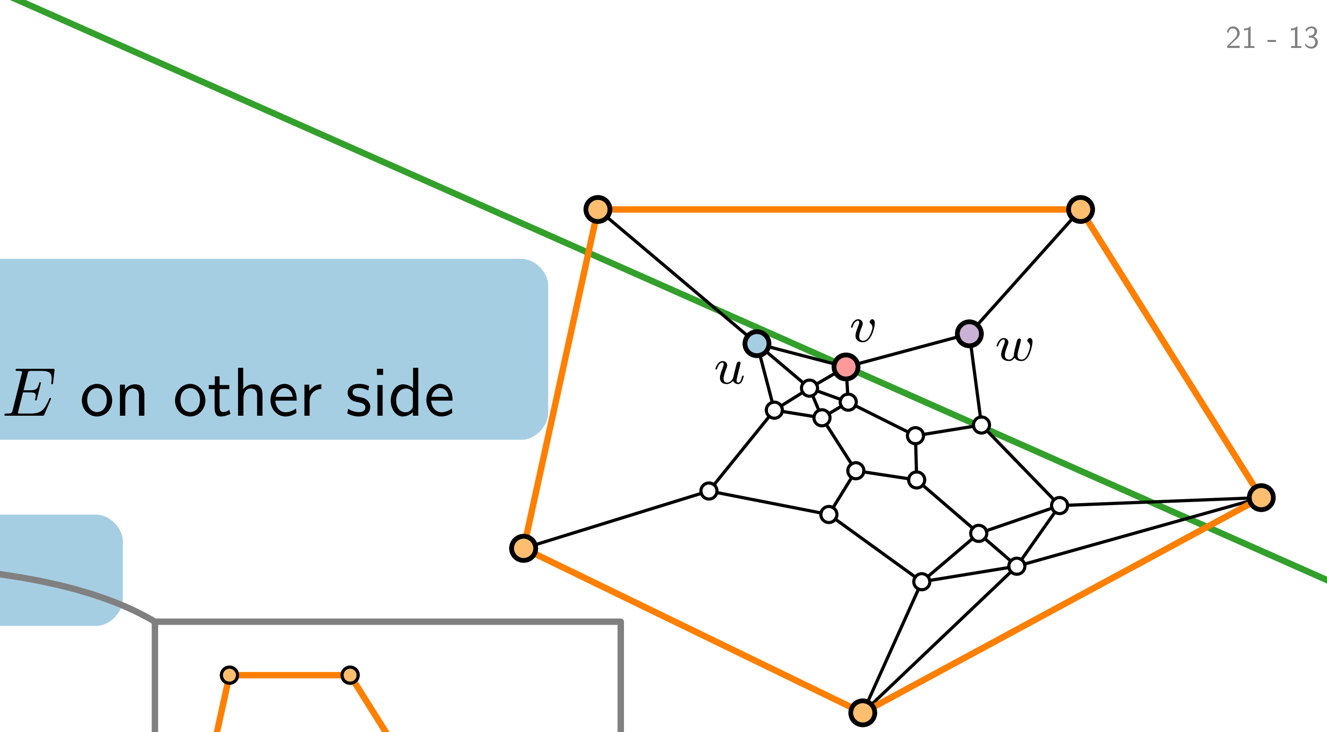


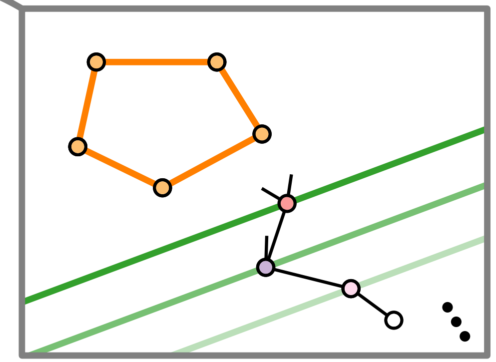
Properties of Tutte Drawings

Property 1. Let $v \in V$ free, ℓ line through v.
$\exists uv \in E$ on one side of $\ell \Rightarrow \exists vw \in E$ on other side

Otherwise, all forces to same side ...

Property 2. All free vertices lie inside C.



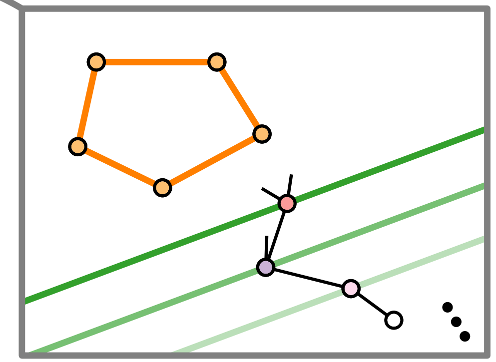


Properties of Tutte Drawings

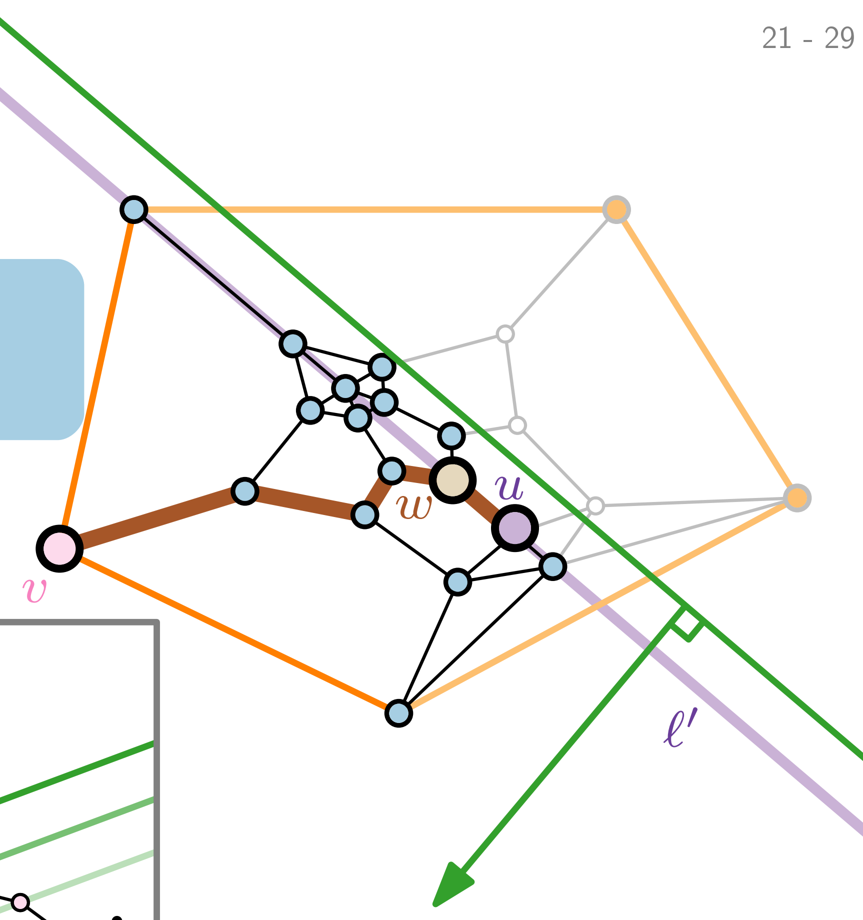
Property 1. Let $v \in V$ free, ℓ line through v.
$\exists uv \in E$ on one side of $\ell \Rightarrow \exists vw \in E$ on other side

Otherwise, all forces to same side . . .

Property 2. All free vertices lie inside C.

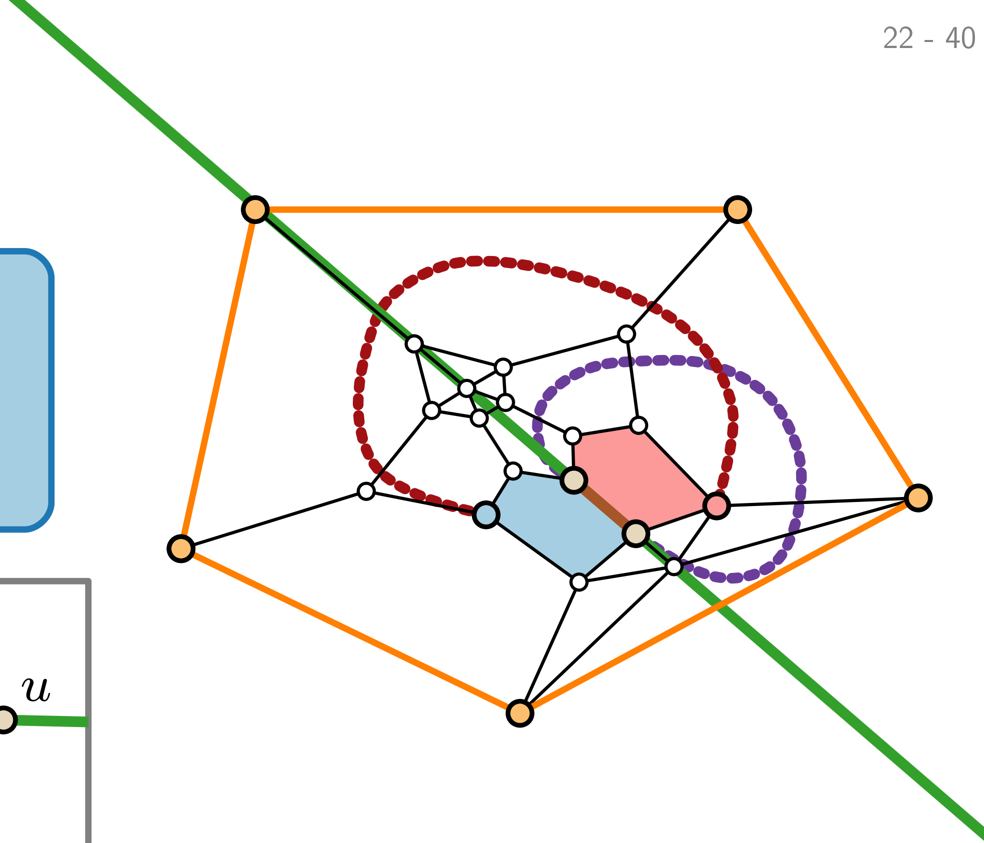


Property 3. Let ℓ be any line.
Let V_ℓ be all vertices on one side of ℓ.
Then $G[V_\ell]$ is connected.



v furthest away from ℓ
Pick any vertex $u \in V_\ell$, ℓ' parallel to ℓ through u
G connected, v not on $\ell' \Rightarrow \exists$ neighbor $w \in V_\ell$ of u on the same side of ℓ' as v
move ℓ' onto w and repeat $\Rightarrow \exists$ path from u to v

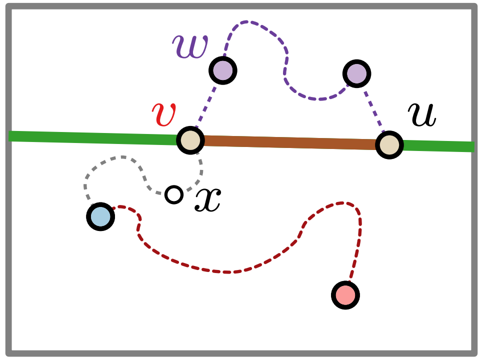
Proof of Tutte's Theorem



Lemma. Let uv be a non-boundary edge, ℓ line through uv. Then the two faces f_1, f_2 incident to uv lie completely on opposite sides of ℓ.

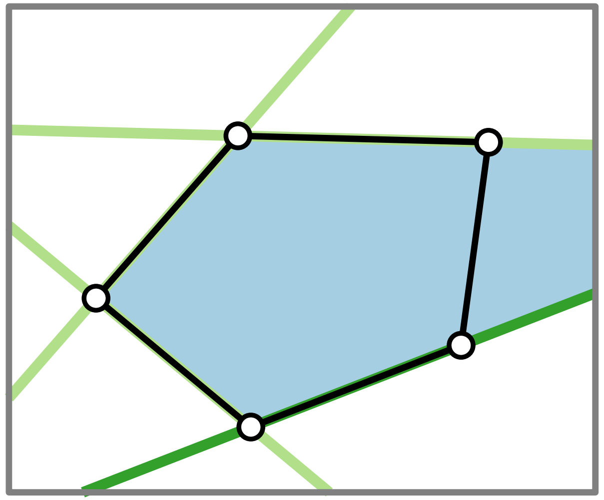
Property 1. Let $v \in V$ free, ℓ line through v.
$\exists xv \in E$ on one side of $\ell \Rightarrow \exists vw \in E$ on other side

Property 3. Let ℓ be any line.
Let V_ℓ be the set of vertices on one side of ℓ.
Then $G[V_\ell]$ is connected.



xv and vw on different sides of $\ell \Rightarrow f_1, f_2$ have angles $< \pi$ at v

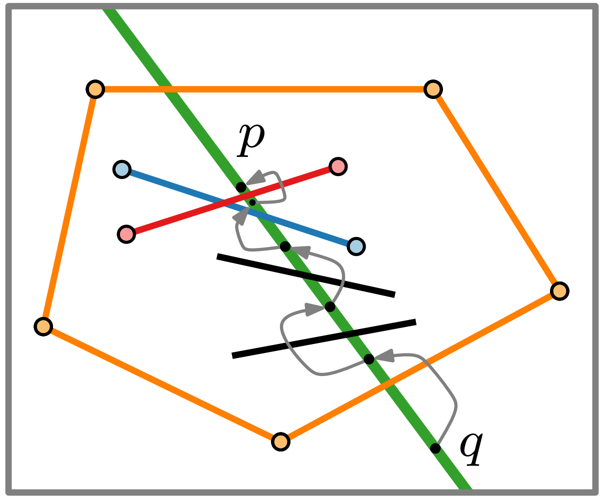
Lemma. All faces are strictly convex.



Property 2. All free vertices lie inside C.

Lemma. The drawing is planar.

p inside two faces
$\Rightarrow q$ in one face
jumping over edge
$\rightarrow$ #faces the same
$\Rightarrow p$ inside one face ⚡



Literature

Main sources:

- [GD Ch. 10] Force-Directed Methods
- [DG Ch. 4] Drawing on Physical Analogies

Original papers:

- [Eades 1984] A heuristic for graph drawing
- [Fruchterman, Reingold 1991] Graph drawing by force-directed placement
- [Tutte 1963] How to draw a graph