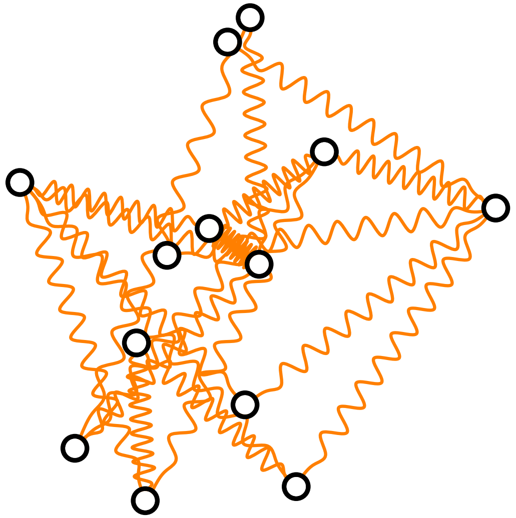
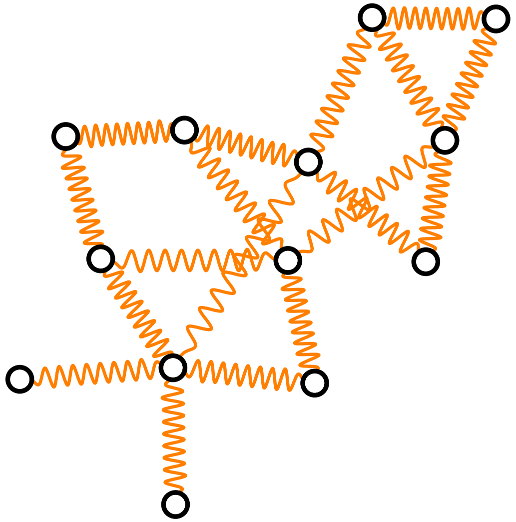
Visualization of Graphs

Lecture 2:
Force-Directed Drawing Algorithms



Part I:
Spring Embedders



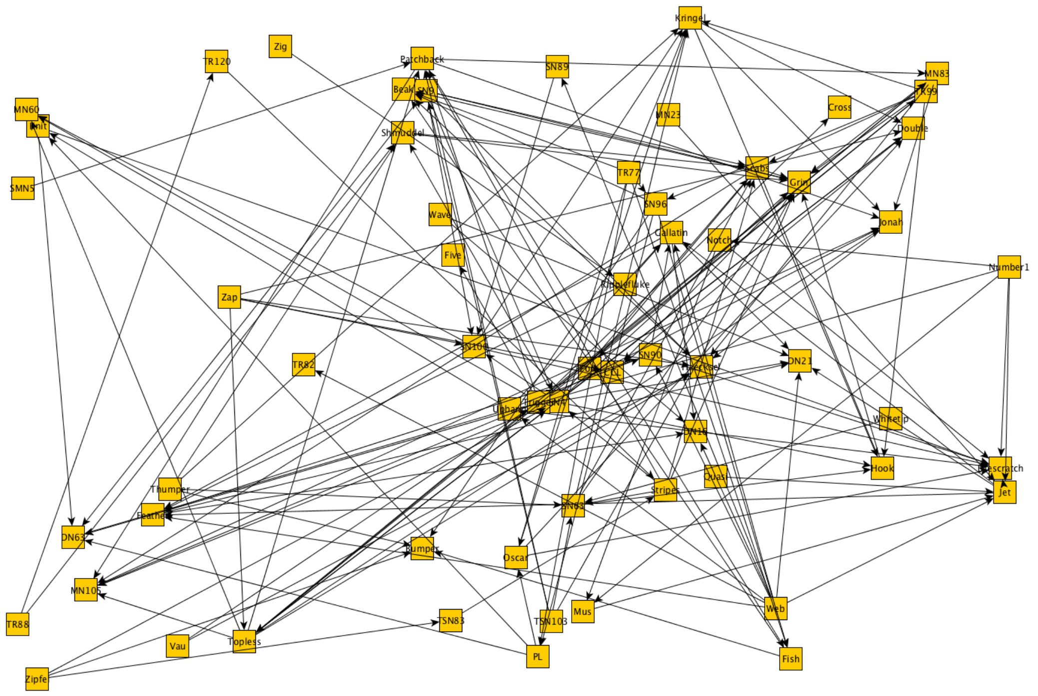
Alexander Wolff

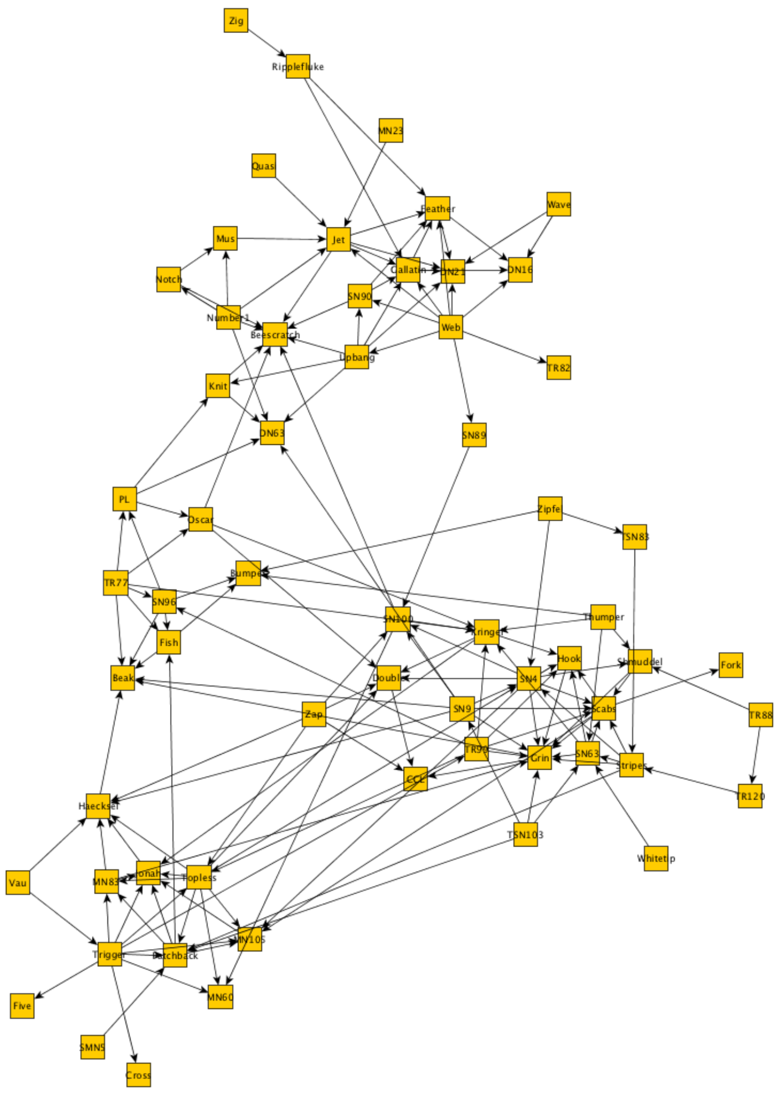
Summer term 2025

General Layout Problem

Input: Graph G

Output: Clear and readable straight-line drawing of G





General Layout Problem

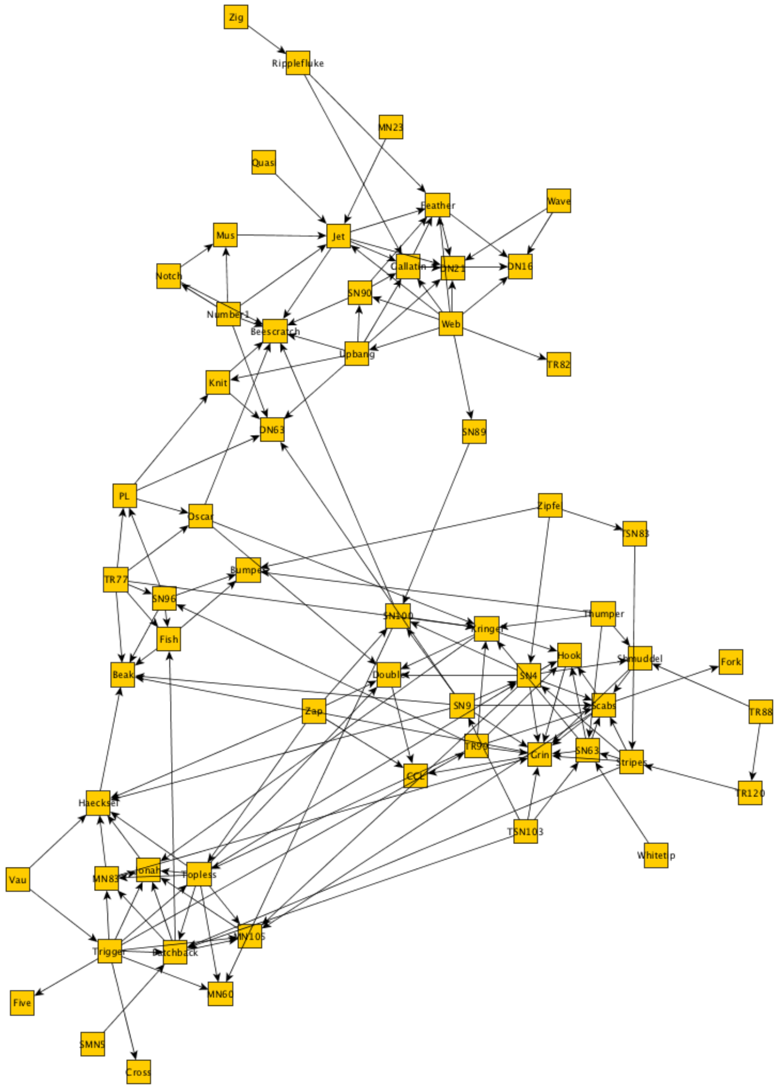
Input: Graph G

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

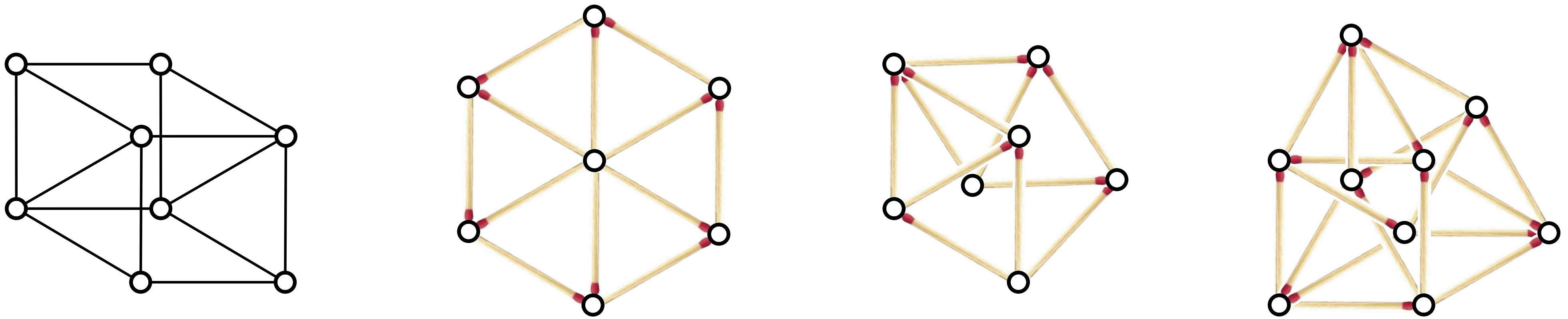
Optimization criteria partially contradict each other.



Fixed Edge Lengths?

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

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



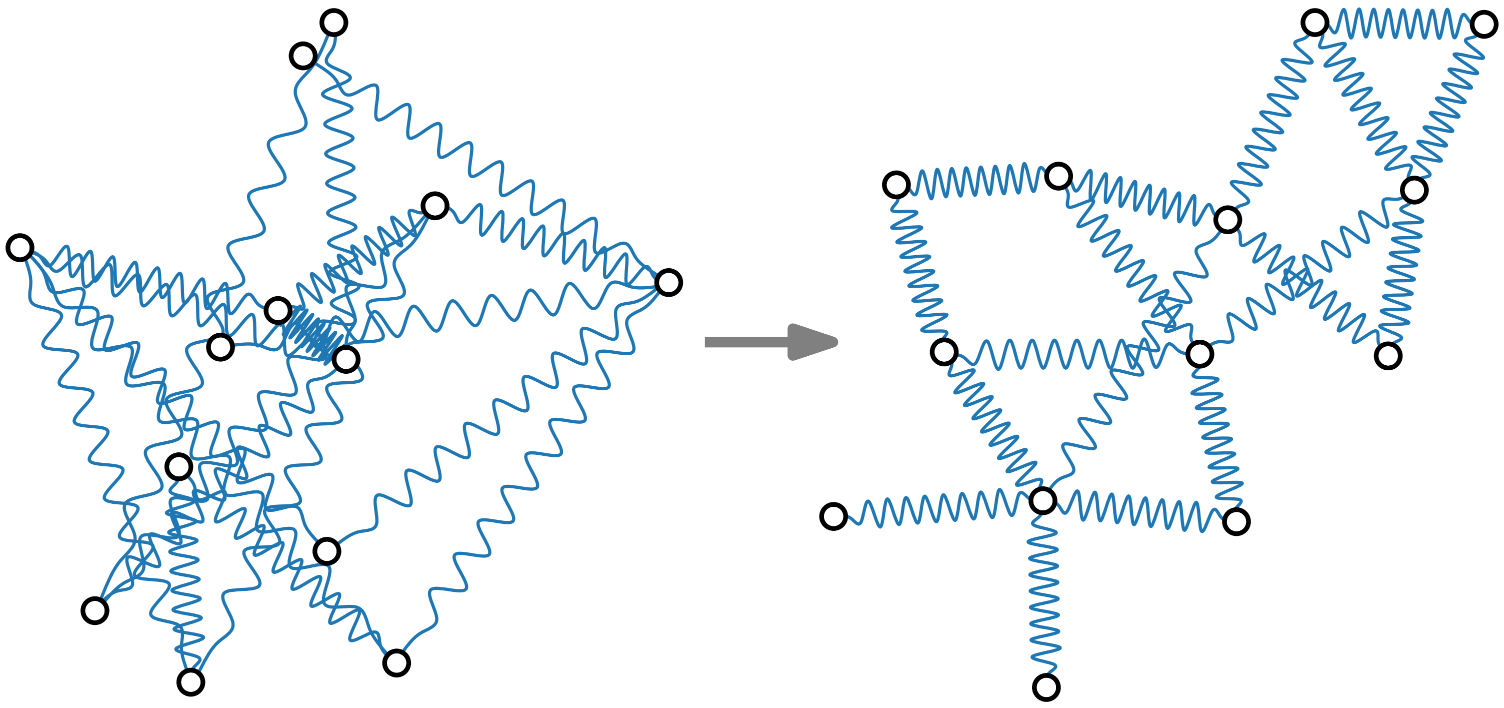
NP-hard for

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

Physical Analogy

Idea. [Eades '84]

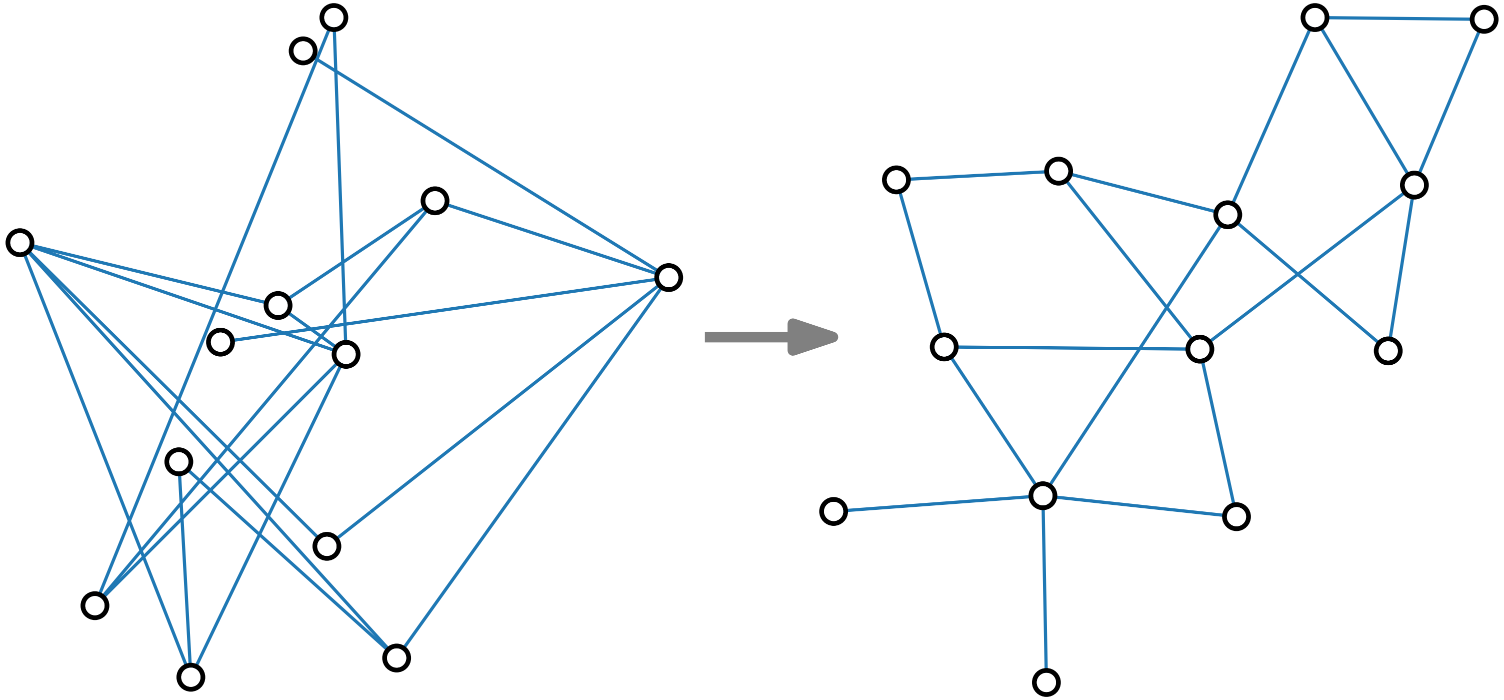
"To embed a graph we replace the vertices by steel rings and replace each edge with a **spring** to form a mechanical system... The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state."



Physical Analogy

Idea. [Eades '84]

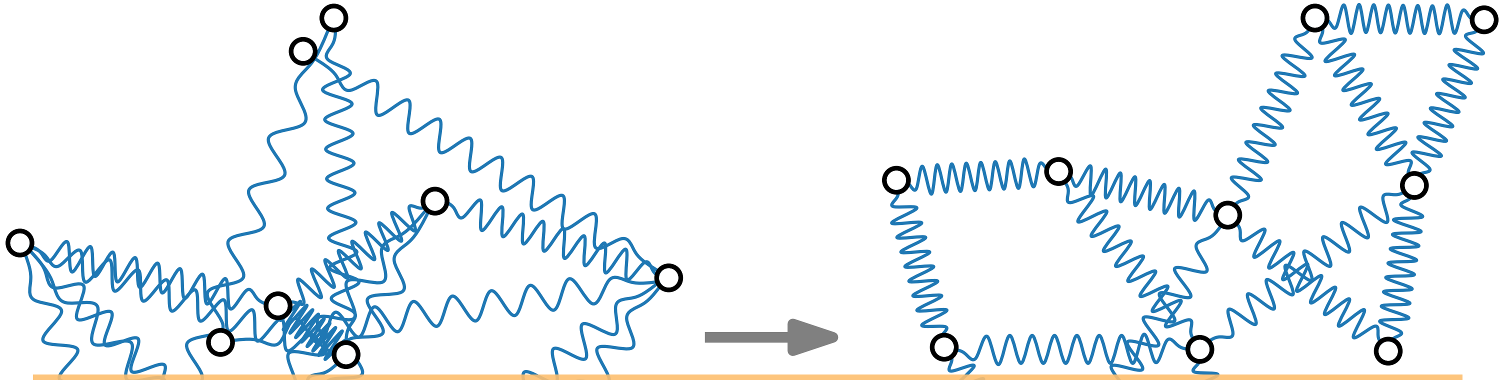
"To embed a graph we replace the vertices by steel rings and replace each edge with a **spring** to form a mechanical system... The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state."



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."



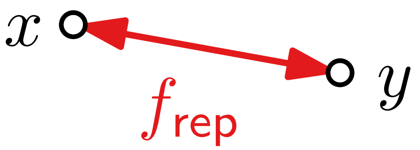
So-called **spring-embedder** algorithms that work according to this or similar principles are among the most frequently used graph-drawing methods in practice.

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

max # iterations

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

$t \leftarrow 1$

while $t \leq K$ **and** $\max_{v \in V(G)} \|F_v(t-1)\| > \varepsilon$ **do**

  foreach $u \in V(G)$ **do**

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

  foreach $u \in V(G)$ **do**

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

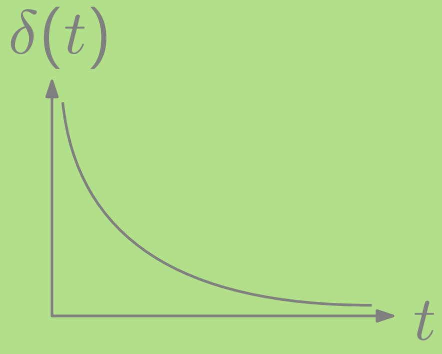
  $t \leftarrow t + 1$

return p

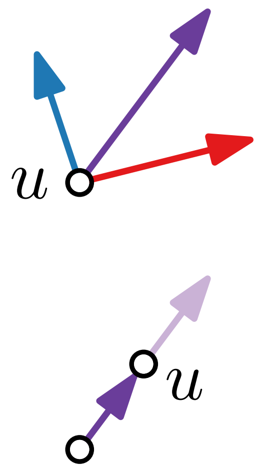
threshold (assume $F_v(0) = \infty$)

vertices adjacent to u

cooling factor



end layout



Spring Embedder by Eades – Model

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

- **Repulsive forces**

repulsion constant (e.g., 2.0)

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

- **Attractive forces**

spring constant (e.g., 1.0)

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

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

- **Resulting displacement vector**

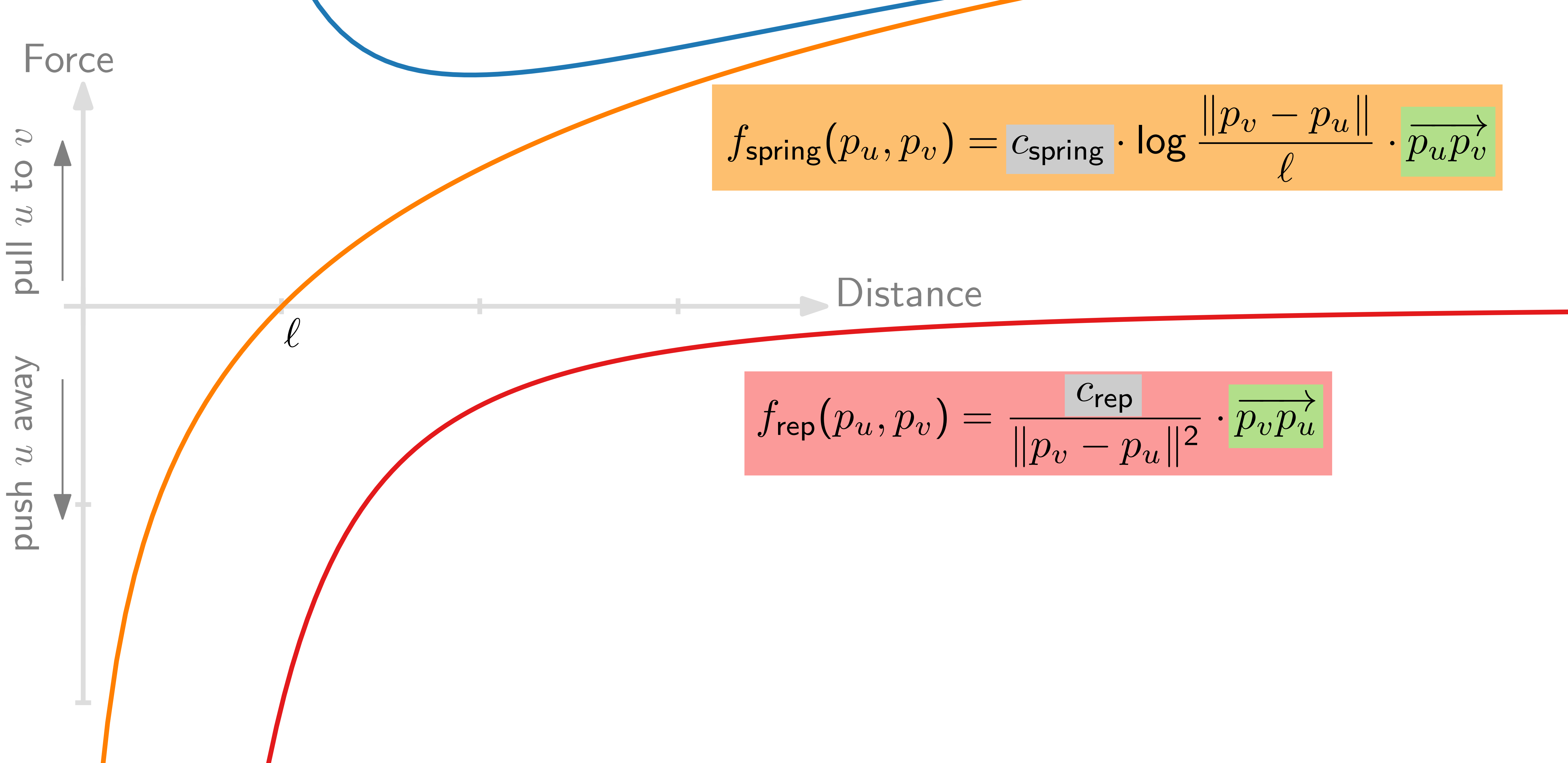
$$F_u = \sum_{v \in V(G)} f_{\text{rep}}(p_u, p_v) + \sum_{v \in \text{Adj}[u]} f_{\text{attr}}(p_u, p_v)$$

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

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



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

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

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.
- May converge to a local minimum that is not a global minimum.
- Computing f_{spring} takes $\mathcal{O}(|E(G)|)$ time; computing f_{rep} takes $\mathcal{O}(|V(G)|^2)$ time.

Influence.

- The original paper by Peter Eades [Eades '84] has been cited more than 2000 times.
- It has become the basis for many further ideas.

Variant by Fruchterman & Reingold

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

- **Repulsive forces**

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

- **Attractive forces**

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

- **Resulting displacement vector**

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

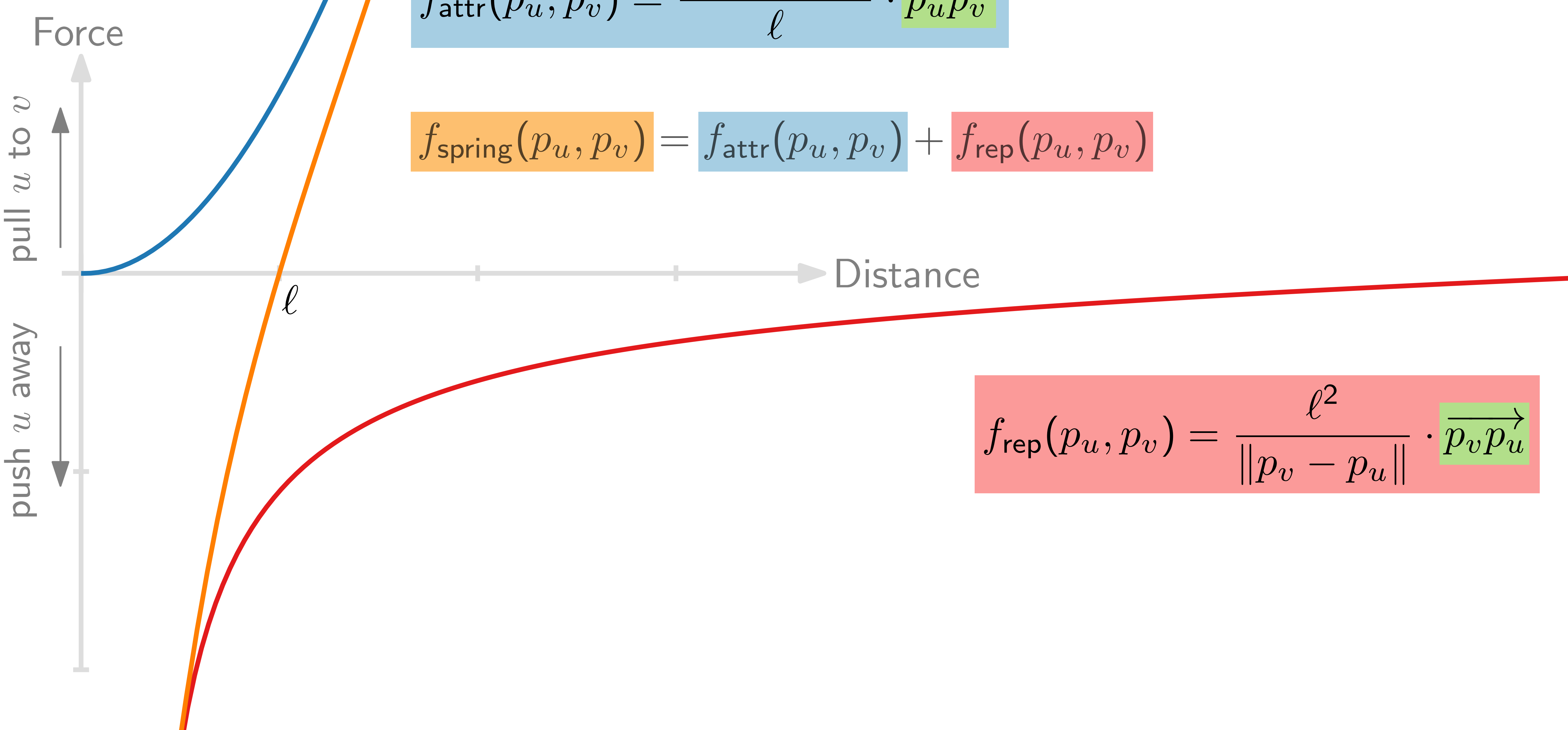
Notation.

- $\|p_u - p_v\|$ = 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

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

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



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

Adaptability

degree of vertex u, i.e., $|\mathrm{Adj}[u]|$

Inertia. ("Trägheit")

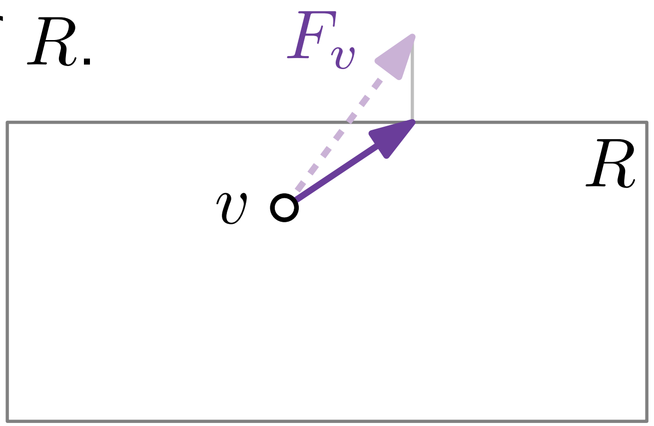
- Define vertex mass $\Phi(u) = 1 + \deg(u)/2$
- Set $f_{\text{attr}}^{\text{new}}(p_u, p_v) = f_{\text{attr}}^{\text{old}}(p_u, p_v)/\Phi(u)$

Gravitation.

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

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

$t \leftarrow 1$

while $t \leq K$ **and** $\max_{v \in V(G)} \|F_v(t-1)\| > \varepsilon$ **do**

  foreach $u \in V(G)$ **do**

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

  foreach $u \in V(G)$ **do**

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

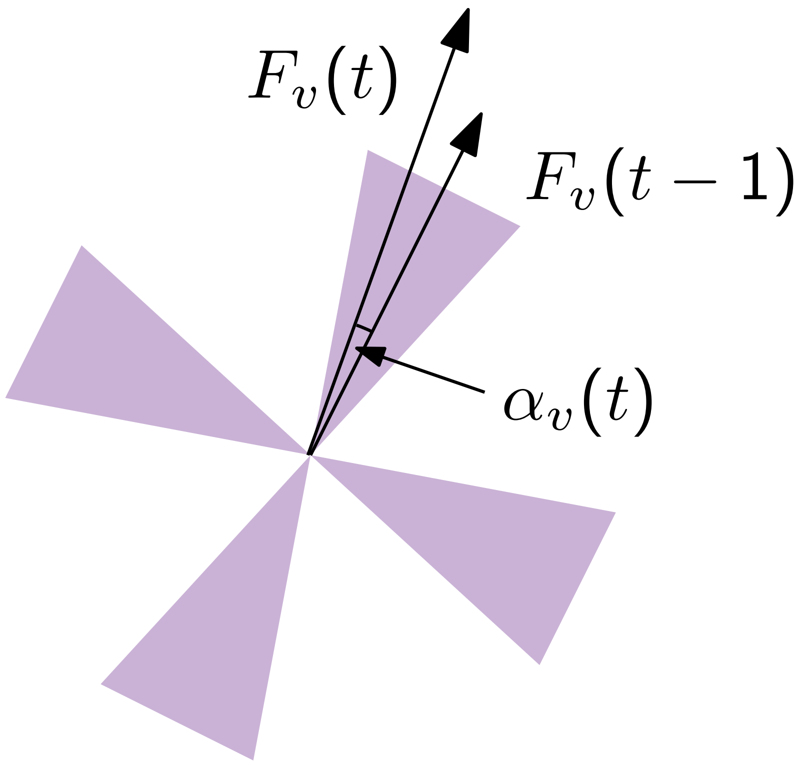
    $\delta_v(t)$

  $t \leftarrow t + 1$

return p

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

[Frick, Ludwig, Mehldau '95]

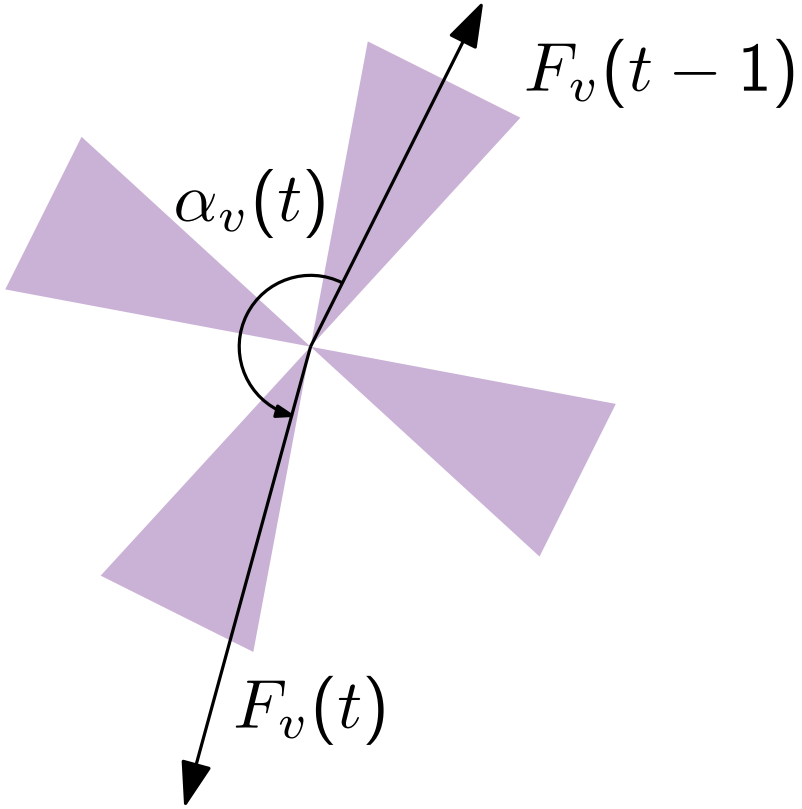


Same direction.

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

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

[Frick, Ludwig, Mehldau '95]



Same direction.

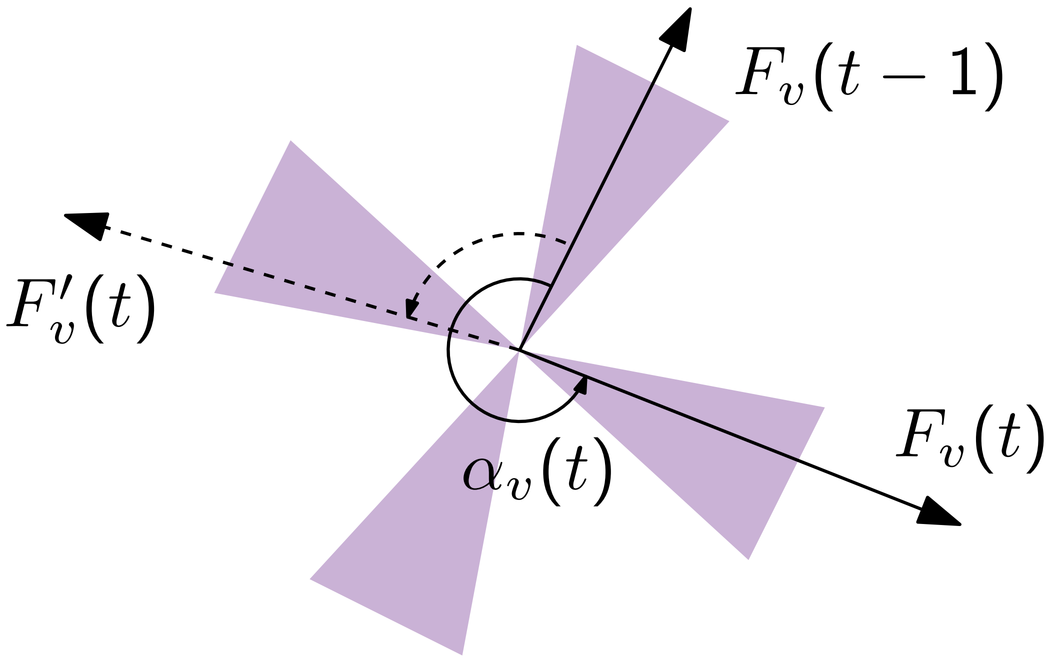
$\rightarrow$ increase temperature $\delta_v(t)$

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 $\delta_v(t)$

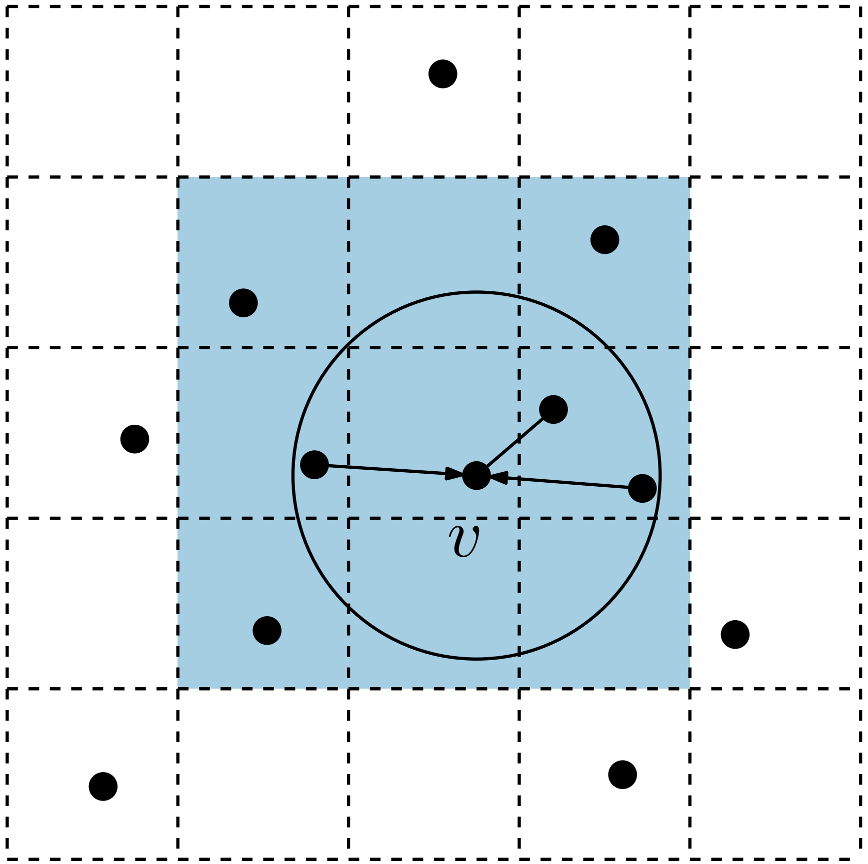
Rotation.

- count rotations
- if applicable

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

Speeding up "Convergence" via Grids

[Fruchterman & Reingold '91]



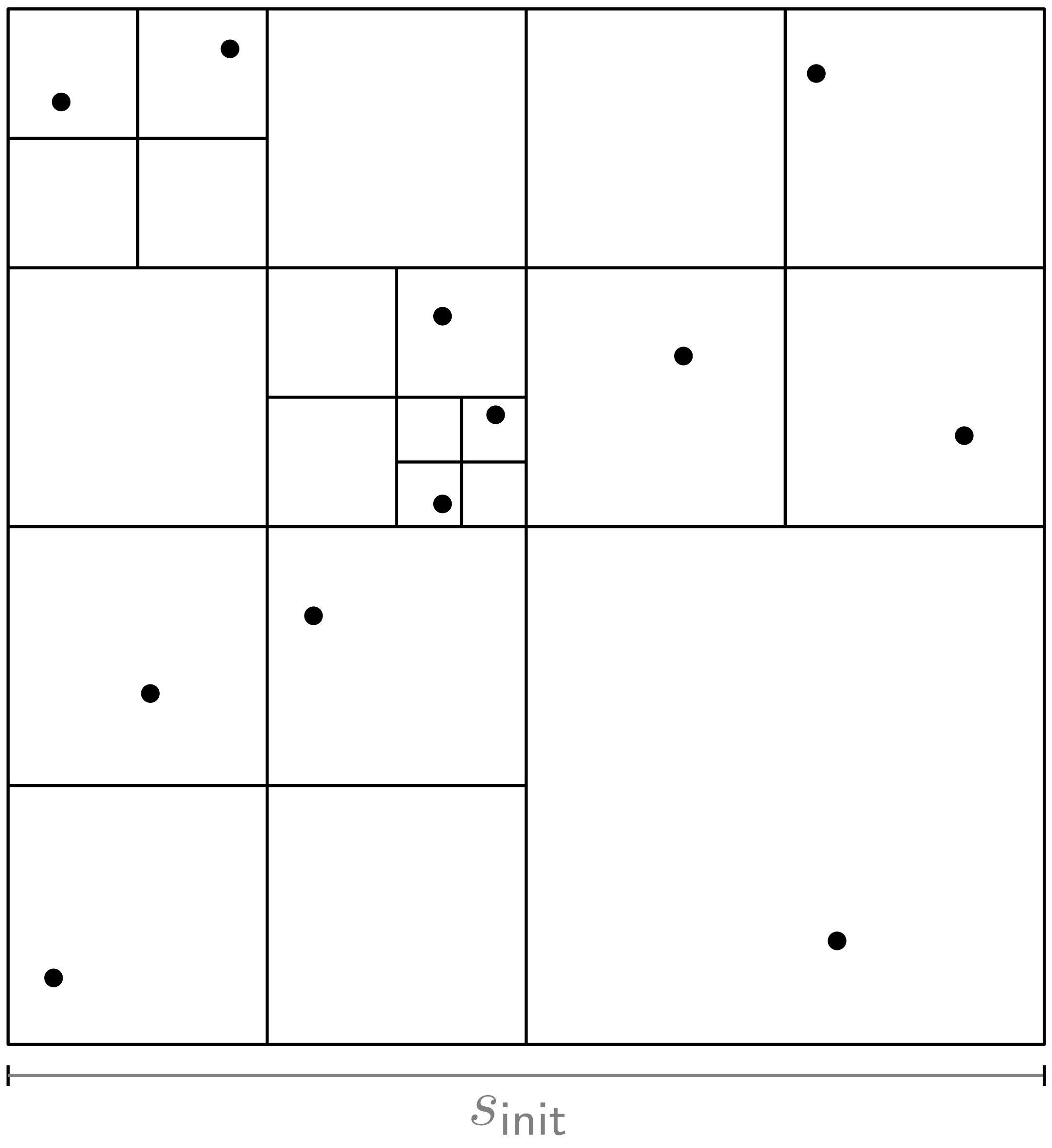
- divide plane into a grid
- consider repulsive forces only to vertices in neighboring cells
- and only if the distance is less than some threshold

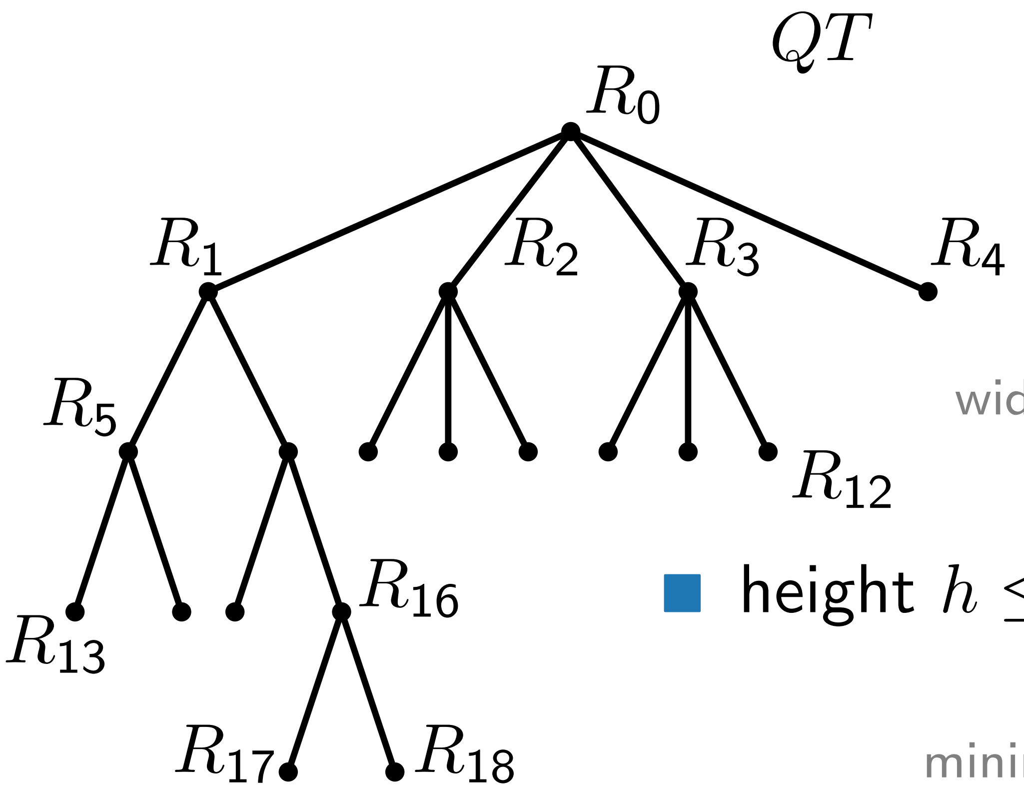
Discussion.

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

Speeding up Repulsive-Force Computation with Quad Trees

[Barnes, Hut '86]





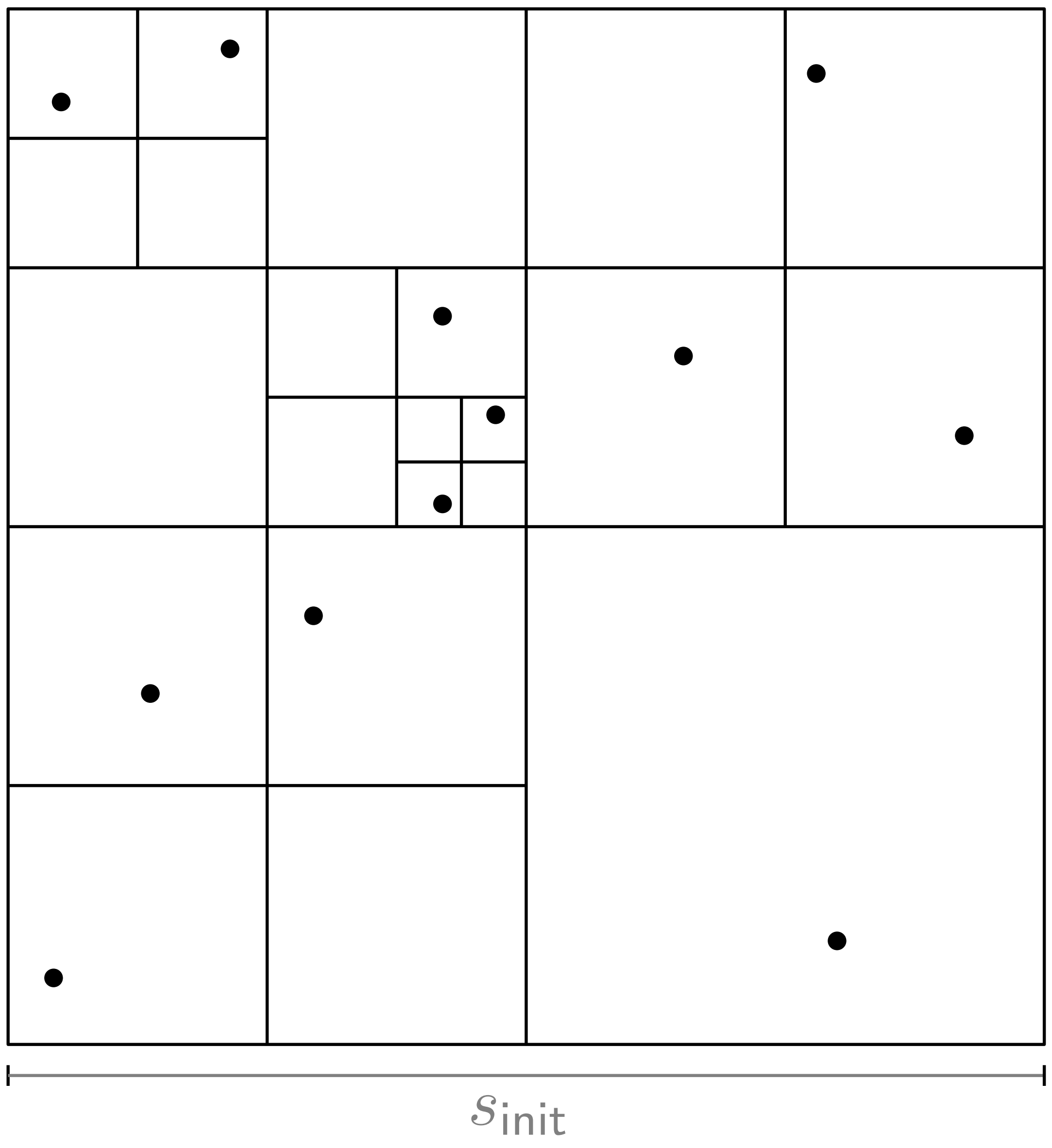
width/height of outer square

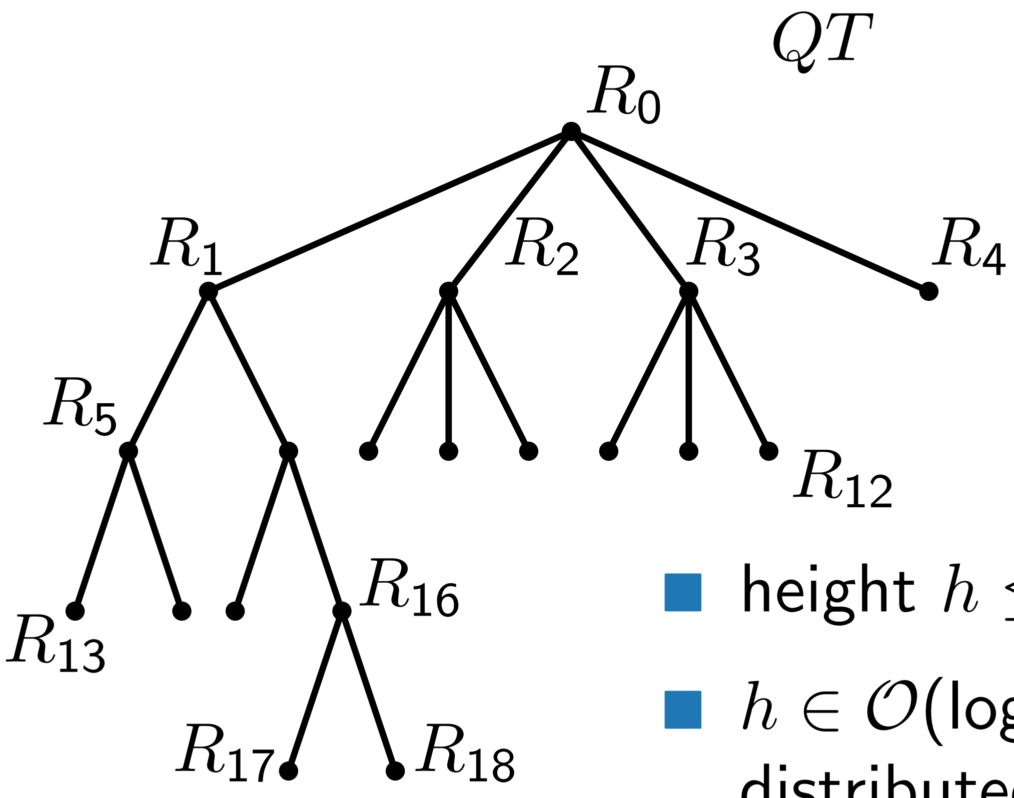
- height $h \leq \log_2\left(\frac{s_{\text{init}}}{d_{\min}}\right) + \frac{3}{2}$

minimum distance btw. two pts.

Speeding up Repulsive-Force Computation with Quad Trees

[Barnes, Hut ’86]

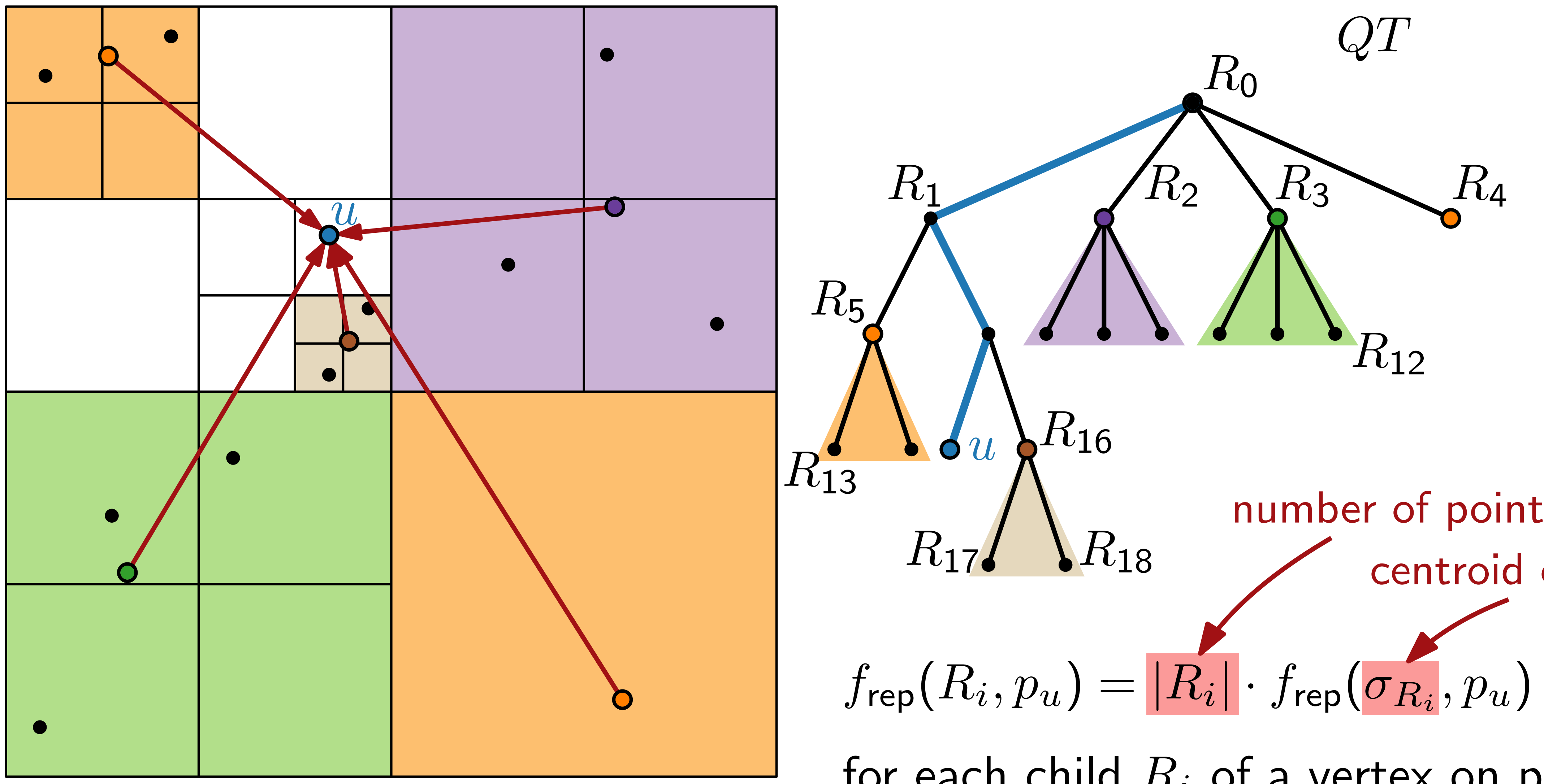




- height $h \leq \log_2\left(\frac{s_{\text{init}}}{d_{\min}}\right) + \frac{3}{2}$
- $h \in \mathcal{O}(\log n)$ if vertices evenly distributed in the initial box
- time/space in $\mathcal{O}(hn)$
- compressed quad tree can be computed in $\mathcal{O}(n \log n)$ time

Speeding up Repulsive-Force Computation with Quad Trees

[Barnes, Hut '86]



number of points in the subtree R_i

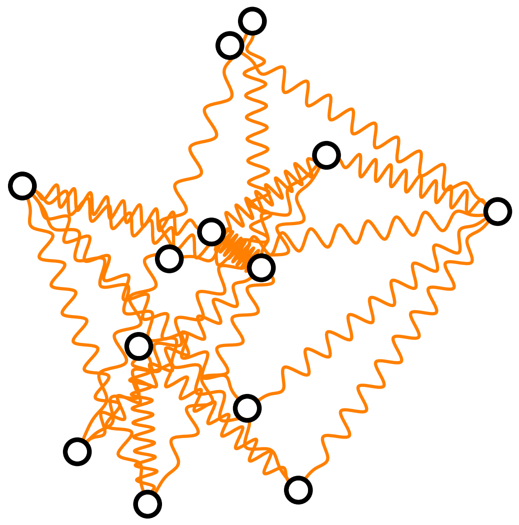
centroid of R_i (pre-computed)

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

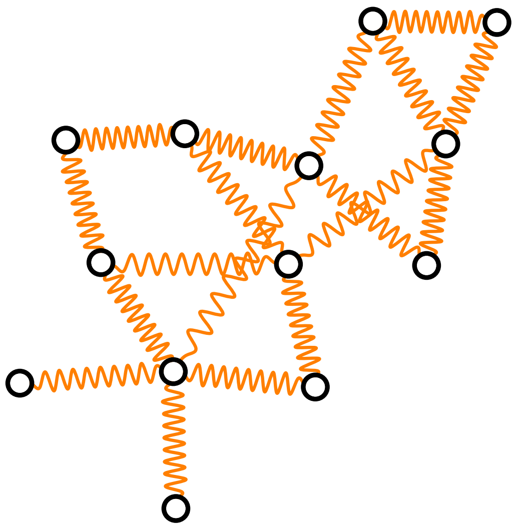
for each child R_i of a vertex on path from root to u.

Visualization of Graphs

Lecture 2:
Force-Directed Drawing Algorithms



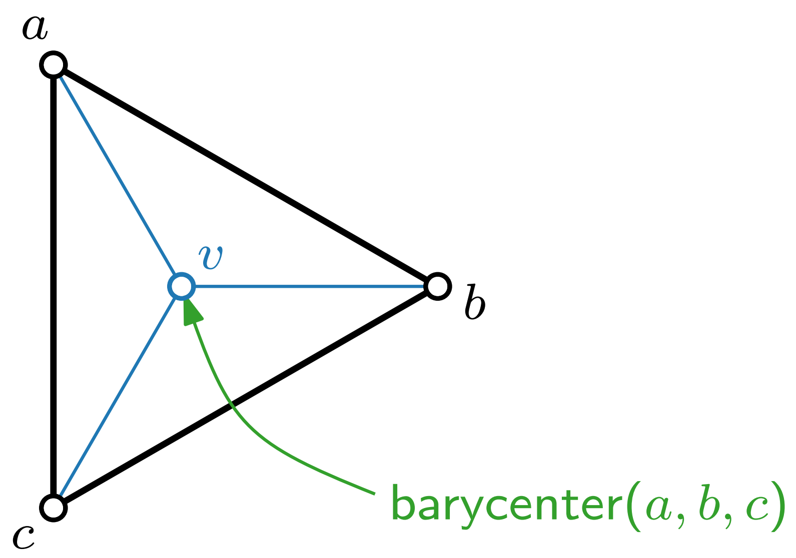
Part II:
Tutte Embeddings



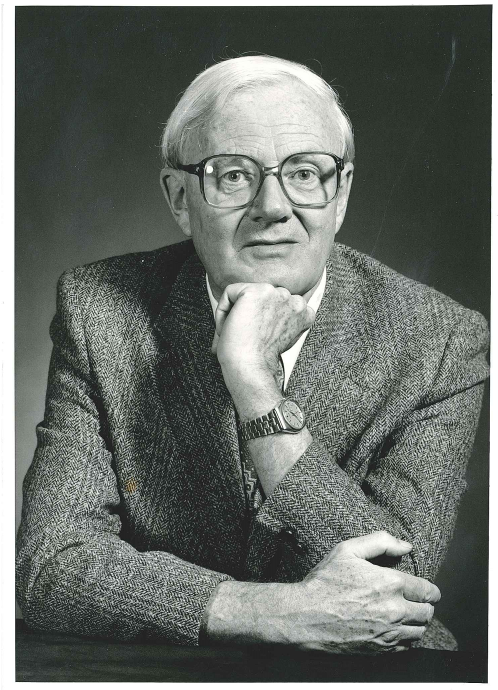
Idea

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.

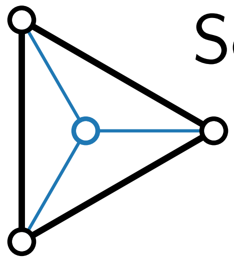
$p_u = \text{barycenter}(\text{Adj}[u])$
$\quad = \sum_{v \in \text{Adj}[u]} p_v / \deg(u)$

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

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

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

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



Solution: fix coordinates of outer face!

- **Repulsive forces** $\boxed{f_{\text{rep}}(p_u, p_v) = 0}$
- **Attractive forces**

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

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

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

System of Linear Equations

Goal. $p_u = (x_u, y_u)$

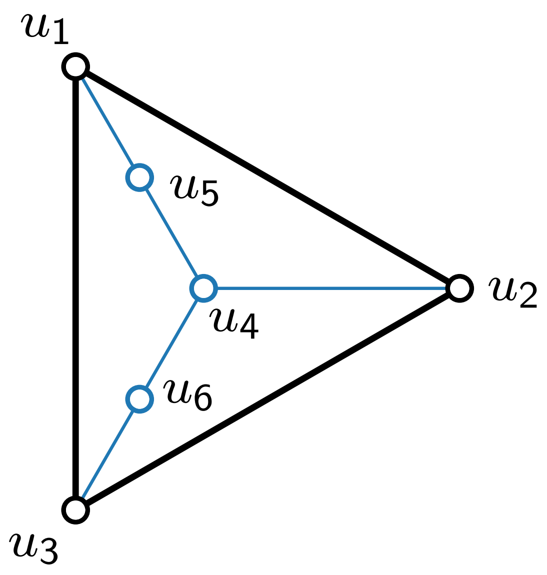
$p_u = \text{barycenter}(\text{Adj}[u]) = \sum_{v \in \text{Adj}[u]} p_v / \deg(u)$

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

Two systems of linear equations:

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

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



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

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

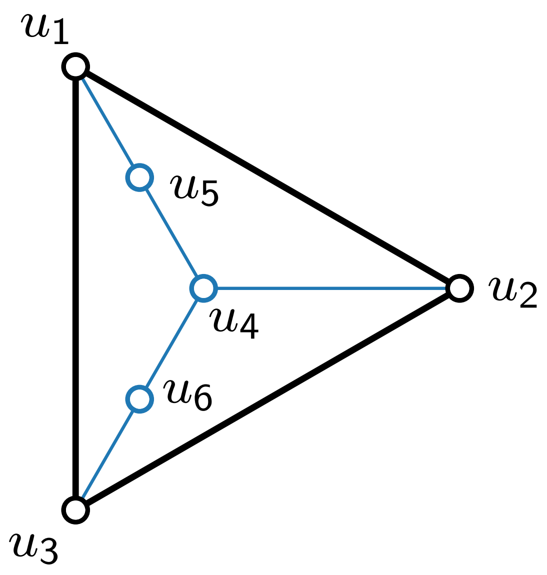
Theorem. **= Tutte drawing**
Tutte's barycentric algorithm admits a unique solution.
It can be computed in polynomial time.

Goal. $p_u = (x_u, y_u)$

$p_u =$ barycenter(Adj[u]) $=$

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

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



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

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:
1. No need to change fixed vertices.
2. Constraints that depend on fixed vertices are constant and can be moved into b.

3-Connected Planar Graphs

G **planar**: G can be drawn such that no two edges cross each other.

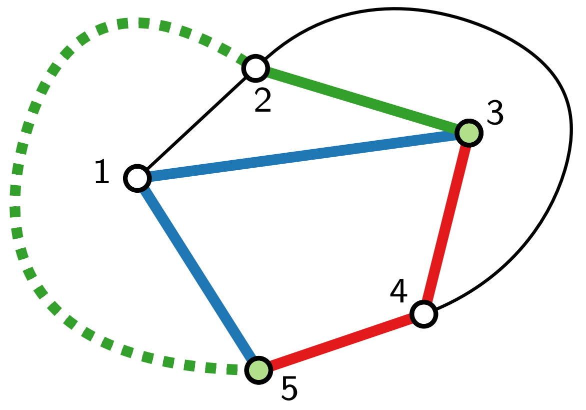
G **connected**: $\exists\ u$–v path for every vertex pair $\{u, v\}$.

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

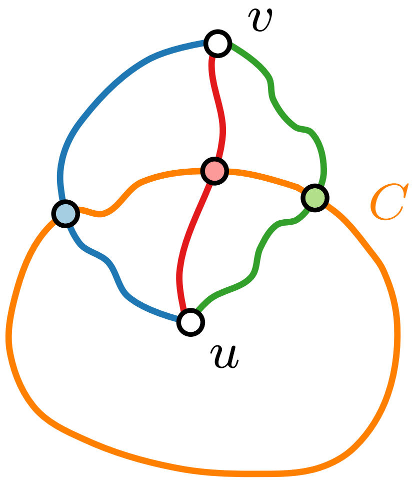
(up to the choice of the outer face and mirroring)

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

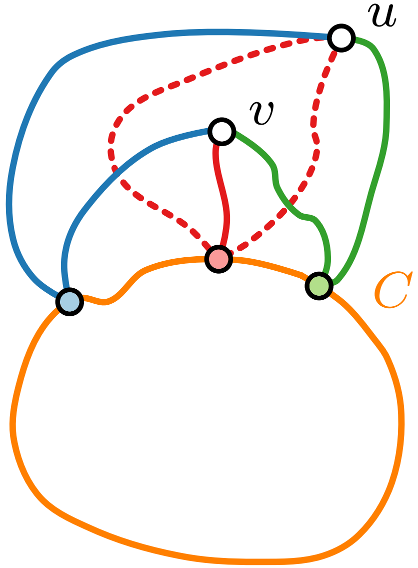
Proof sketch.
Γ_1, Γ_2 planar embeddings of G.
Let C be a face of Γ_2, but not of Γ_1.
u inside C in Γ_1, v outside C in Γ_1
both on same side in Γ_2



Γ_1

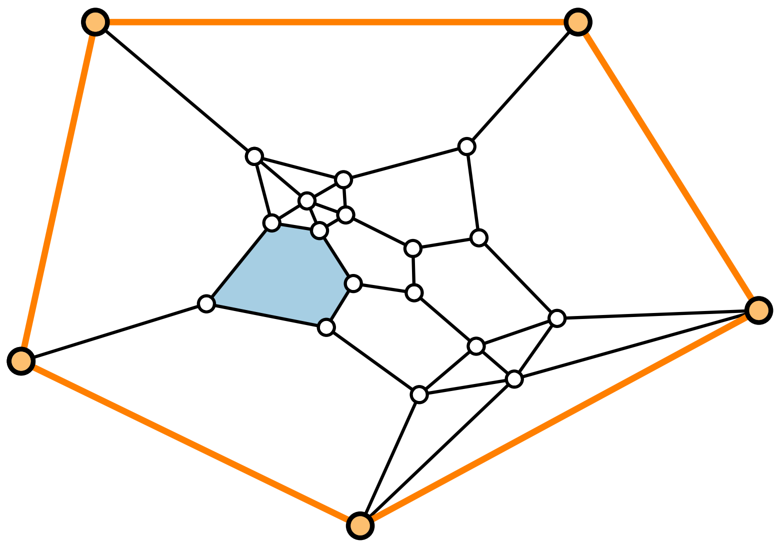


Γ_2



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.

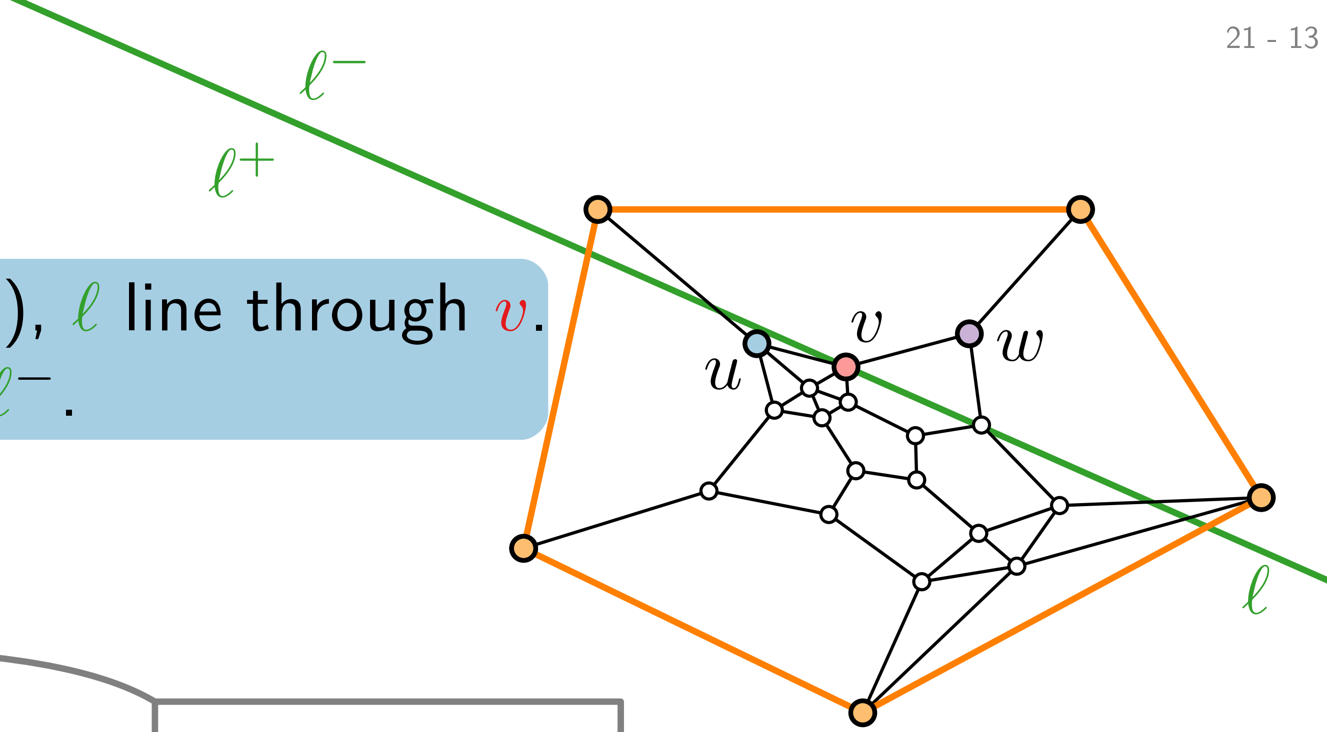


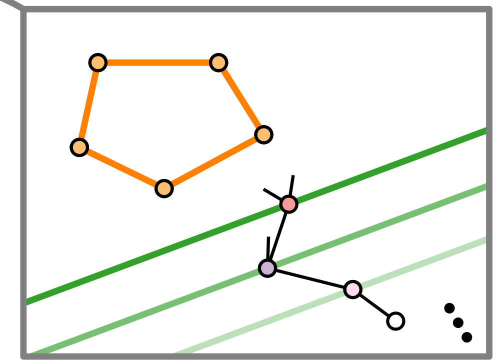
Properties of Tutte Drawings

Property 1. Let $v \in V(G)$ be free (i.e., not fixed), ℓ line through v.
$\exists u \in \mathrm{Adj}[v] \cap \ell^+ \Rightarrow \exists w \in \mathrm{Adj}[v] \cap \ell^-$.

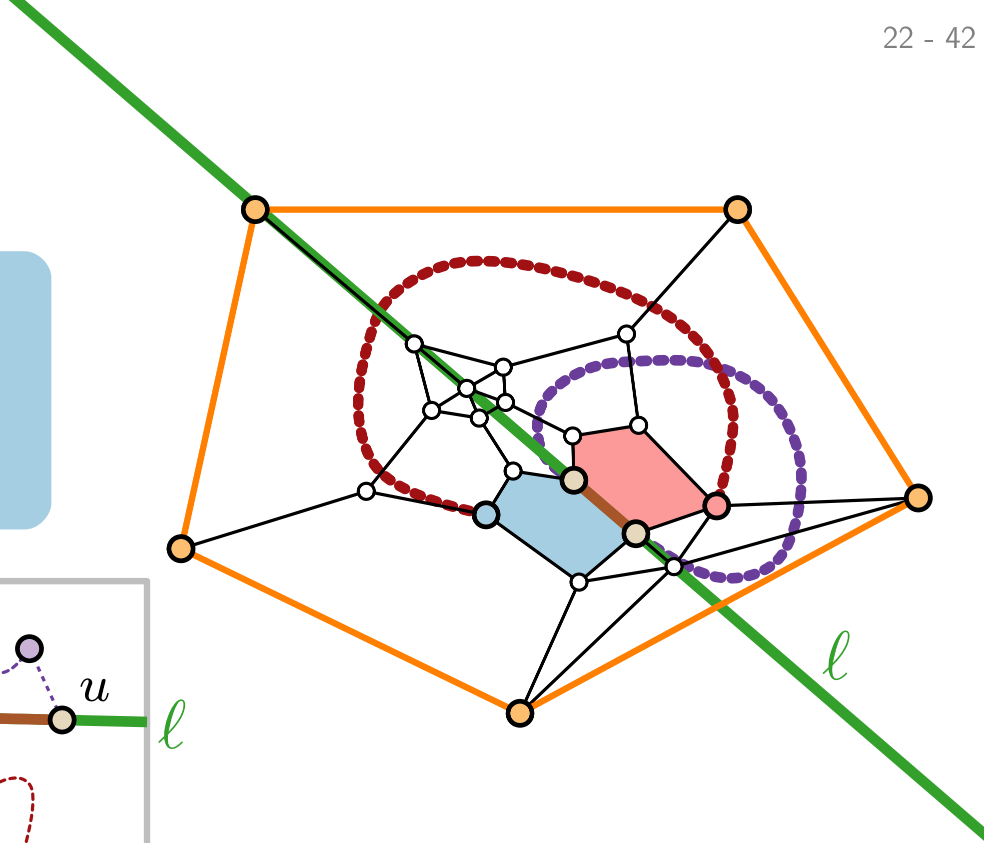
Otherwise, all forces pull v to the same side . . .

Property 2. All free vertices lie inside C.





Proof of Tutte's Theorem

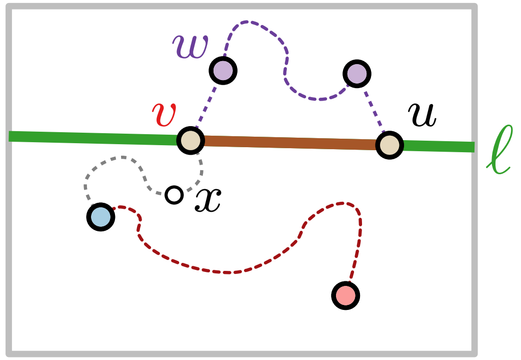


Lemma. Let uv be a non-boundary edge, ℓ line through uv. Then the two faces f_1, f_2 incident to uv lie completely on opposite sides of ℓ.

Property 1. Let v be a free vertex, ℓ line through v.
$\exists x \in \mathrm{Adj}[v] \cap \ell^+ \Rightarrow \exists w \in \mathrm{Adj}[v] \cap \ell^-$.

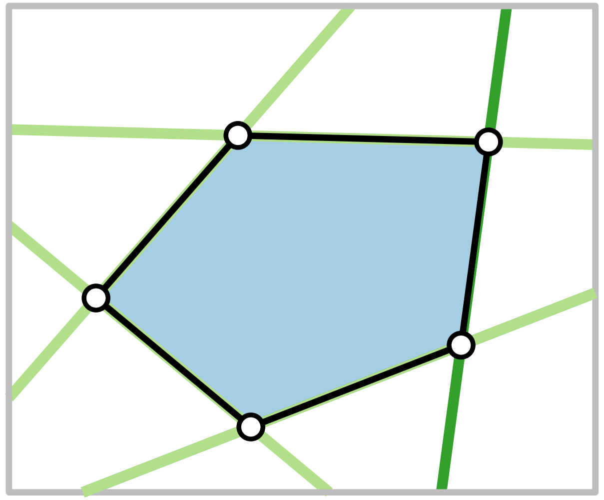
Property 3. Let ℓ be any line.
Let V_ℓ be the set of vertices on one side of ℓ.
Then $G[V_\ell]$ is connected.

Property 4. No vertex is collinear with all its neighbors.



Lemma. All internal faces are strictly convex.

Lemma. The drawing is planar.



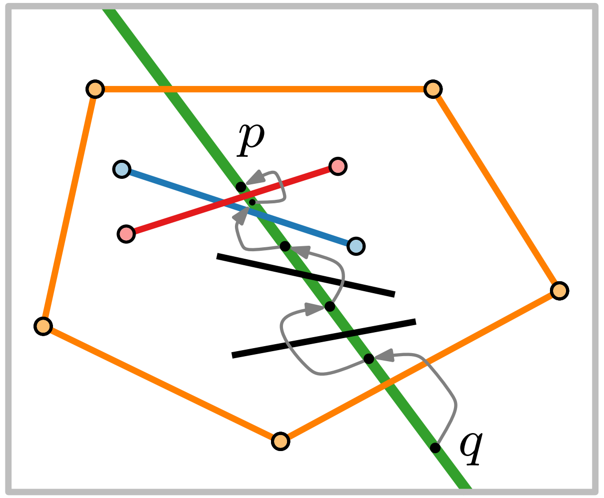
Assume that point p lies in two faces.

Property 2. All free vertices lie inside C.

$\Rightarrow$ q lies in one (i.e., the outer) face.

When jumping an edge, #faces doesn't change.

$\Rightarrow$ p lies in one face. ↯



Discussion

- In practice, force-directed graph drawing methods are very often used.
- Numerous variants, adaptations, and extensions exist.
- They are well-suited for small and medium-size graphs (up to ≈ 100 vertices).
- A way to deal with larger graphs, is to coarsen the graph by merging vertices and first to draw the coarsened graph and then to unpack and draw the vertices to the original graph.
- In practice, the related technique of *multidimensional scaling* (MDS) is often used, too. There, for every pair of vertices, an optimal distance (the distance in the graph) is determined and a drawing with these optimal distances is computed in high-dimensional space. Afterwards this drawing is projected into the plane.
- From a theoretical perspective, Tutte drawings posses many powerful properties.
- If a graph is not 3-connected, we can (temporarily) add sufficiently many edges.
- In practice, Tutte drawings are hardly used because the inner parts often become tiny.

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