Karlsruhe Institute of Technology

Algorithms for Graph Visualization

Force-Directed Algorithms

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze

20.11.2018

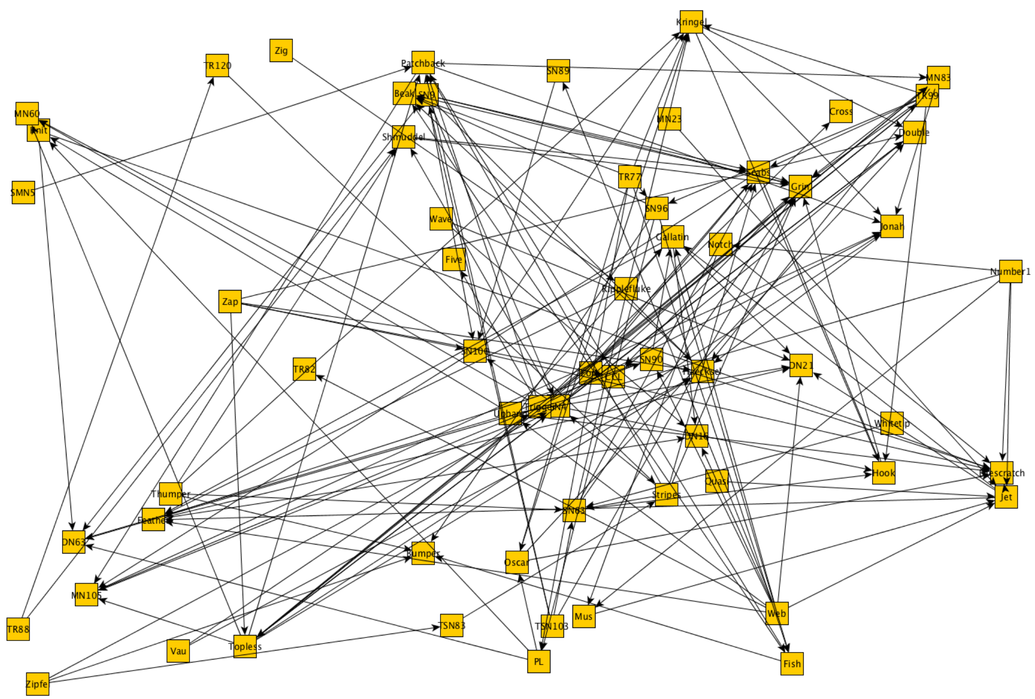
Introduction

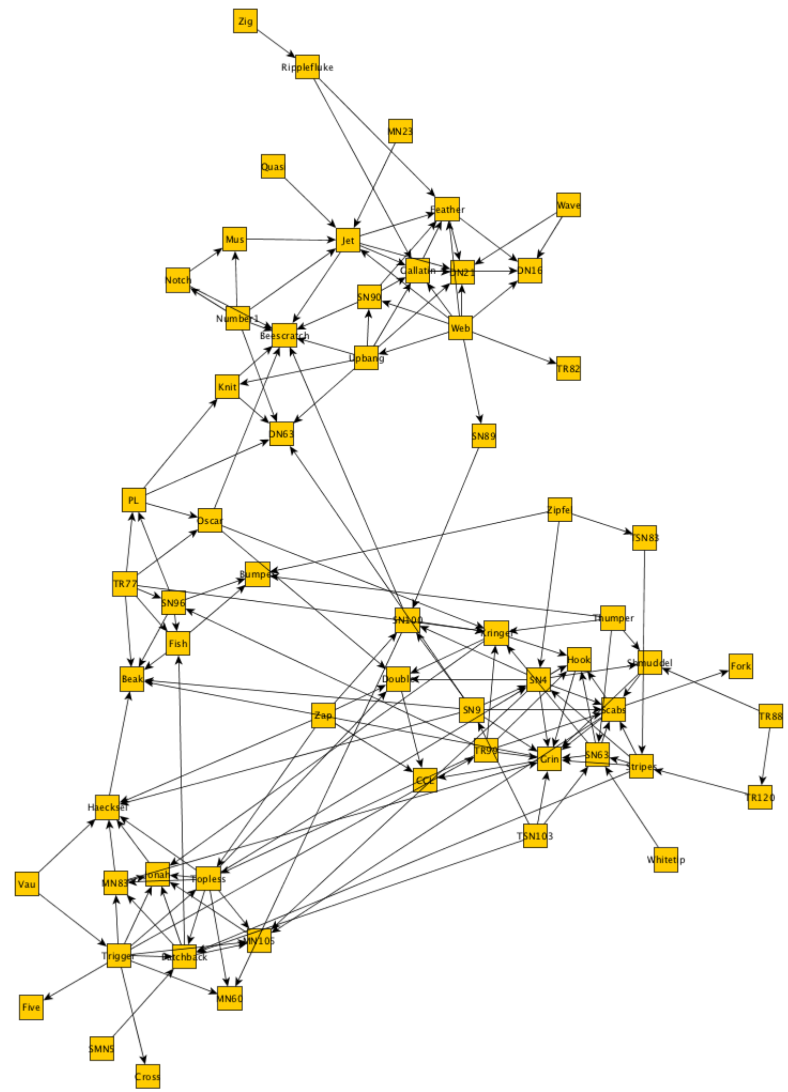
- Before: always based on some properties: tree, series-parallel graph, planar graph
- and on some additional information: ordering of the vertices, decompositions into SP-components
- Today: more direct and intuitive method based on physical analogies
- The methods are very popular: intuitiveness, easy to program, generality, fairly satisfactory results,...

General Layout Problem

Given: Graph $G = (V, E)$

Find: Clear and readable drawing of G





Which aesthetic criteria would you optimize?

General Layout Problem

Given: Graph $G = (V, E)$

Find: Clear and readable drawing of G

Criteria:

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

General Layout Problem

Given: Graph $G = (V, E)$

Find: Clear and readable drawing of G

Criteria:

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

⚠ Optimization criteria partially contradict each other

Example: Fixed edge-length

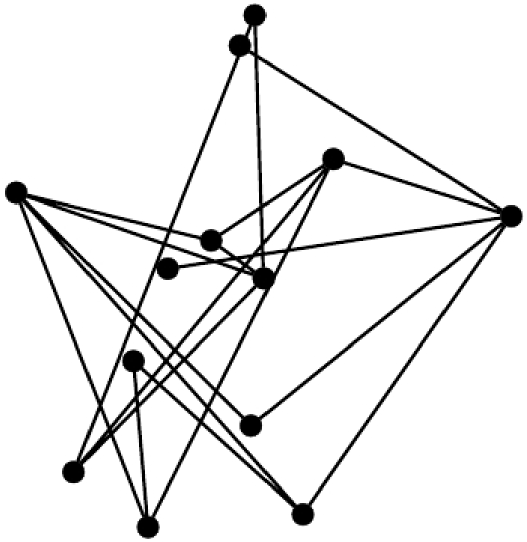
Given: Graph $G = (V, E)$, required edge length $\ell(e)$, $\forall e \in E$

Find: Drawing of G which realizes all the edge lengths

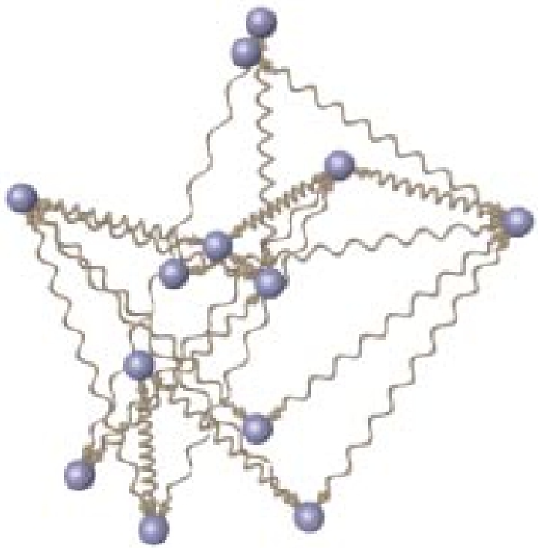
NP-hard for

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

Physical Model

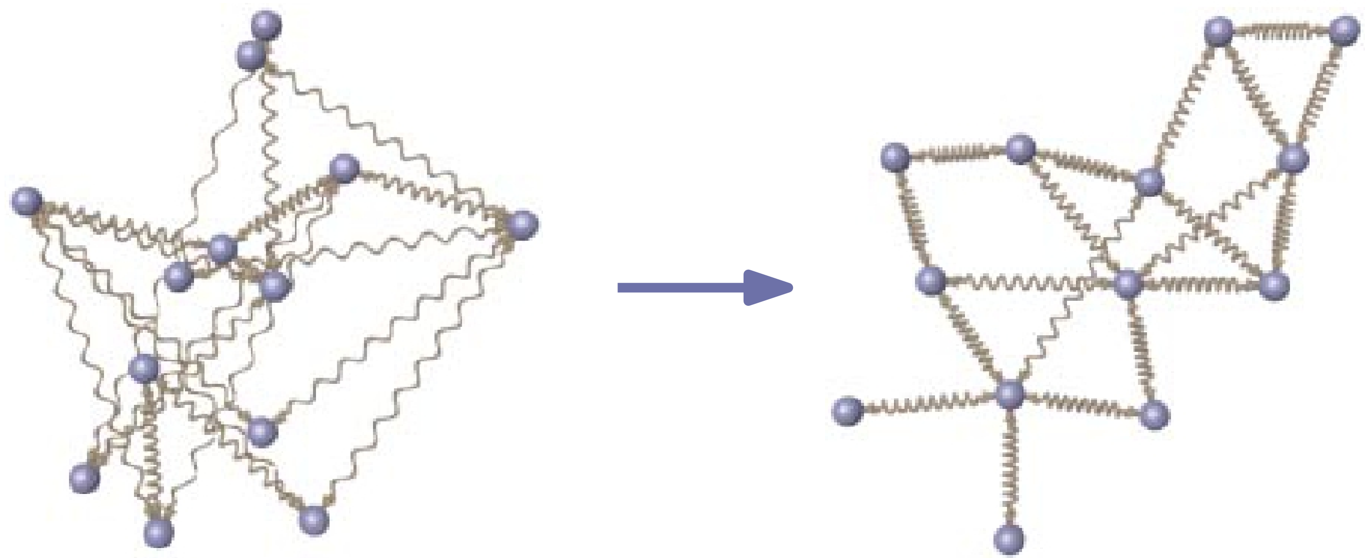


Physical Model



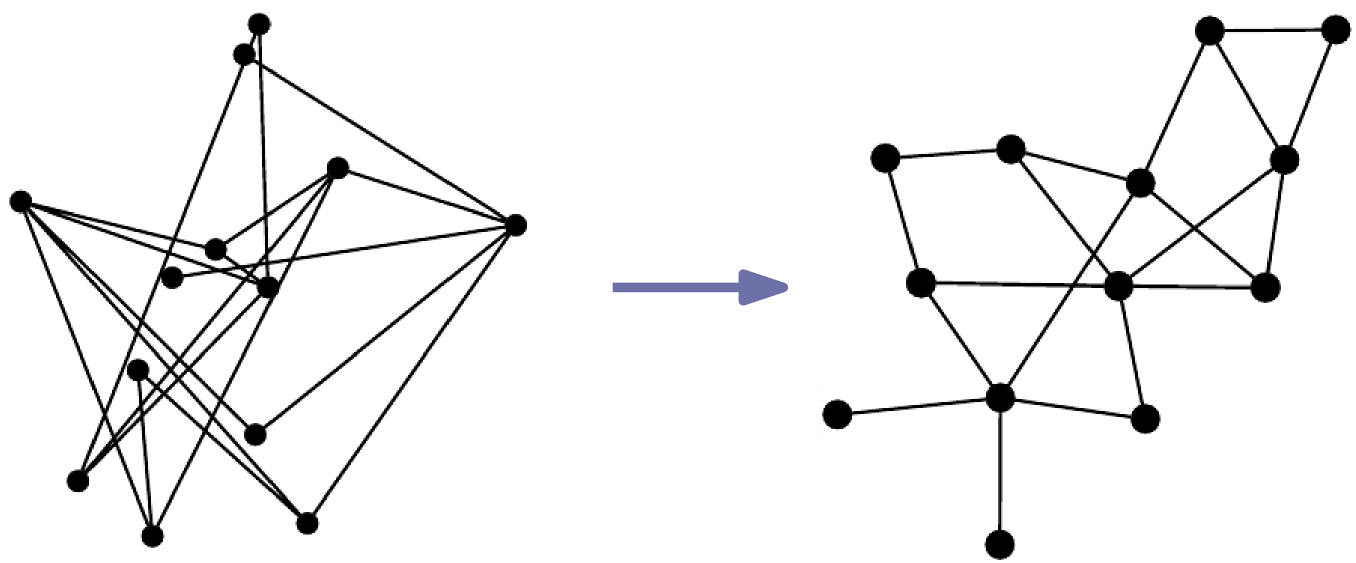
“To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system . . .

Physical Model



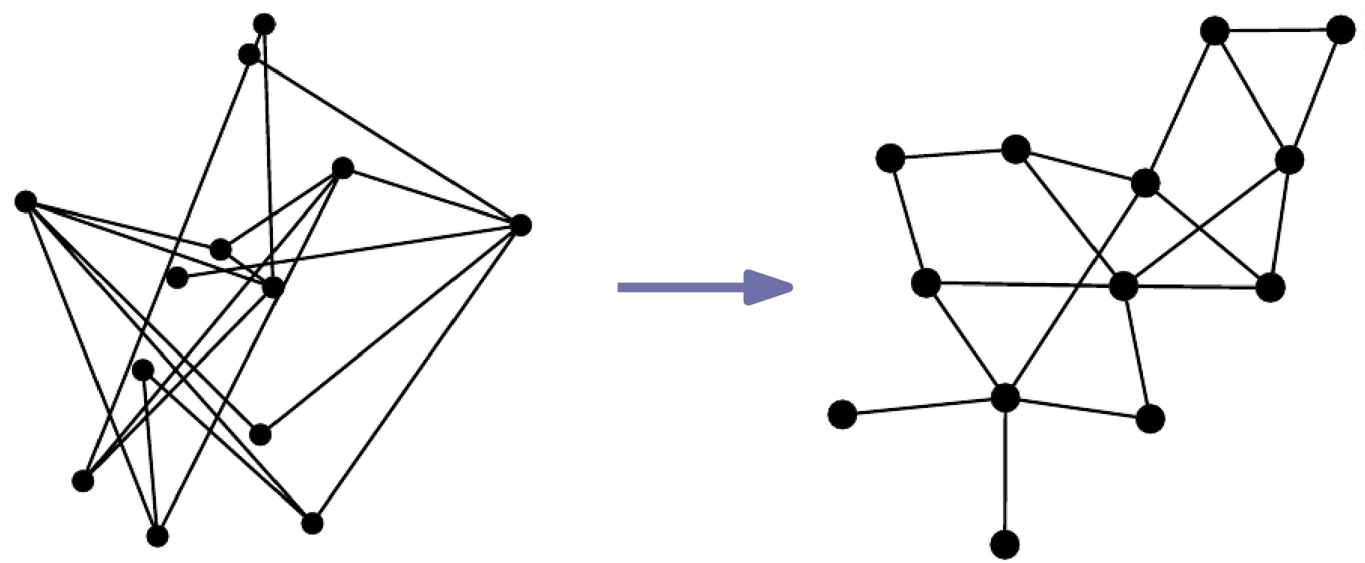
"To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system . . . The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state." [Eades, '84]

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." [Eades, '84]

Physical Model



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

“To … eac… are pla… e rings move the system to a minimal energy state.” [Eades, ’84]

Notation

Karlsruhe Institute of Technology

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

Spring-Embedder (Eades, 1984)

Model:

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

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

Spring-Embedder (Eades, 1984)

Model:

- repulsive force between two 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

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

Spring-Embedder (Eades, 1984)

Karlsruhe Institute of Technology

Model:

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

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

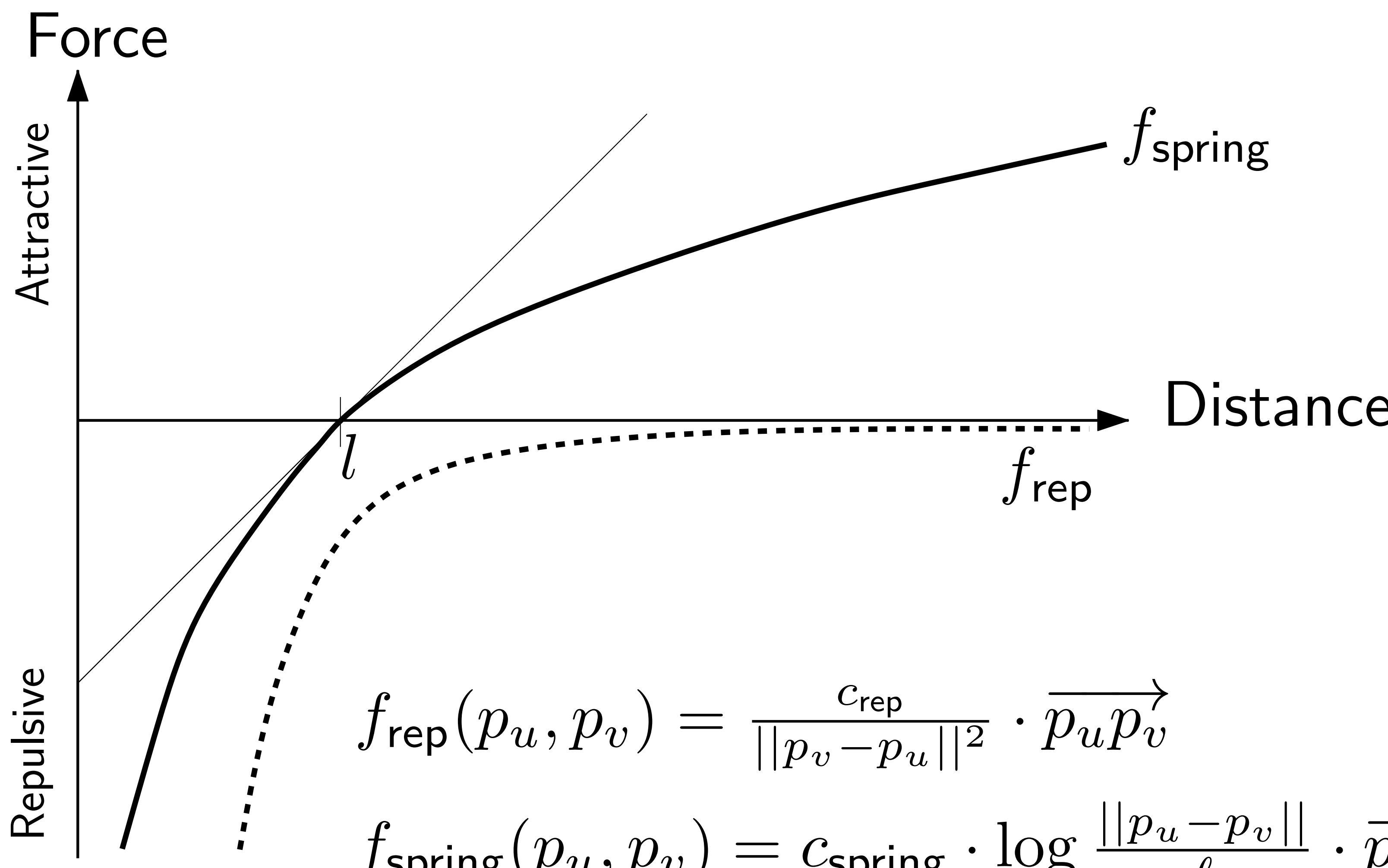
- attractive force between adjacent vertices u and v

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

- 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 Spring-Embedder Forces (Eades, 1984)



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

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

Algorithm Spring-Embedder (Eades, 1984)

Input: $G = (V, E)$ connected undirected graph with initial placement $p = (p_v)_{v \in V}$, number of interations $K \in \mathbb{N}$, threshold $\varepsilon > 0$, constant $\delta > 0$

Output: Layout p with "low internal stress"

$t \leftarrow 1$

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

  foreach $v \in V$ **do**

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

  foreach $v \in V$ **do**

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

  $t \leftarrow t + 1$

Algorithm Spring-Embedder (Eades, 1984)

Input: $G = (V, E)$ connected undirected graph with initial placement $p = (p_v)_{v \in V}$, number of interations $K \in \mathbb{N}$, threshold $\varepsilon > 0$, constant $\delta > 0$

Output: Layout p with "low internal stress"

$t \leftarrow 1$

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

  foreach $v \in V$ **do**

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

  foreach $v \in V$ **do**

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

  $t \leftarrow t + 1$

Algorithm Spring-Embedder (Eades, 1984)

Input: $G = (V, E)$ connected undirected graph with initial placement $p = (p_v)_{v \in V}$, number of interatios $K \in \mathbb{N}$, threshold $\varepsilon >$

Output: Layout p with "low

$t \leftarrow 1$

while $t < K$ **and** $\max_{v \in V} \| F$

  foreach $v \in V$ **do**

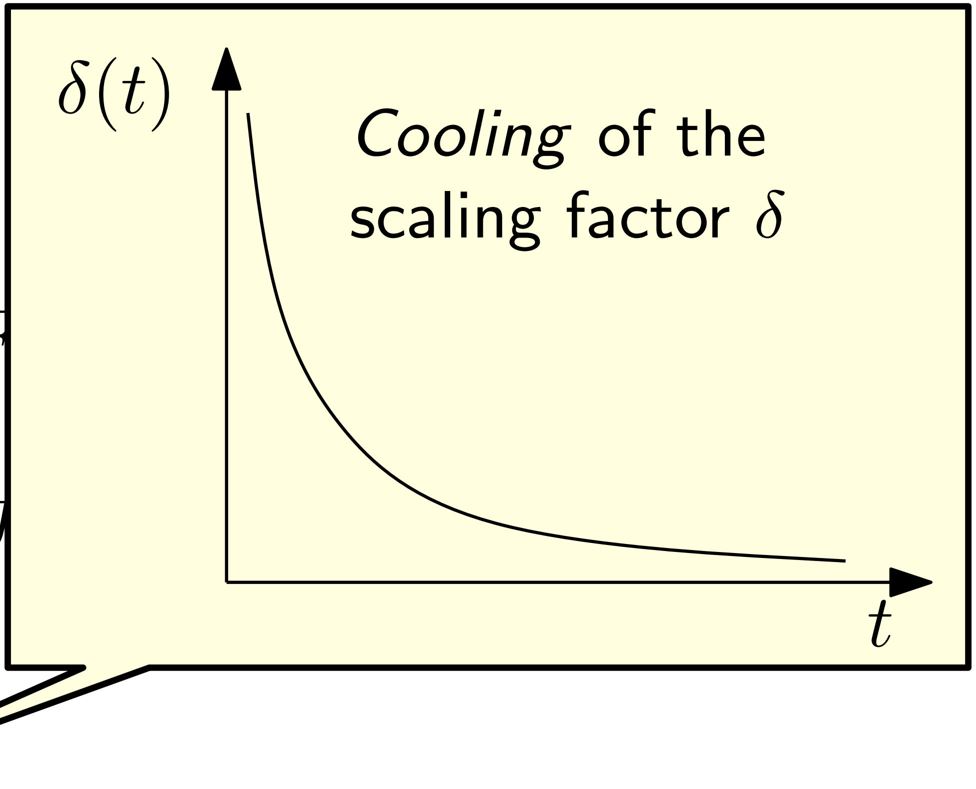
    $F_v(t) \leftarrow \sum_{u:\{u,v\} \notin E} f$

      $\sum_{u:\{u,v\} \in E}$

  foreach $v \in V$ **do**

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

  $t \leftarrow t + 1$



Discussion

Advantages

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

Discussion

Advantages

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

Disadvantages

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

Discussion

Advantages

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

Disadvantages

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

Influence

- Original paper by Peter Eades got 1800 citations (400 in the past four years)
- Basis for many further ideas

Variant: Fruchterman & Reingold (1991)

Model:

- repulsive force between **all** node pairs u and v

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

Variant: Fruchterman & Reingold (1991)

Model:

- repulsive force between **all** node pairs u and v

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

- attractive force between two adjacent nodes u and v

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

Variant: Fruchterman & Reingold (1991)

Model:

- repulsive force between **all** node pairs u and v

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

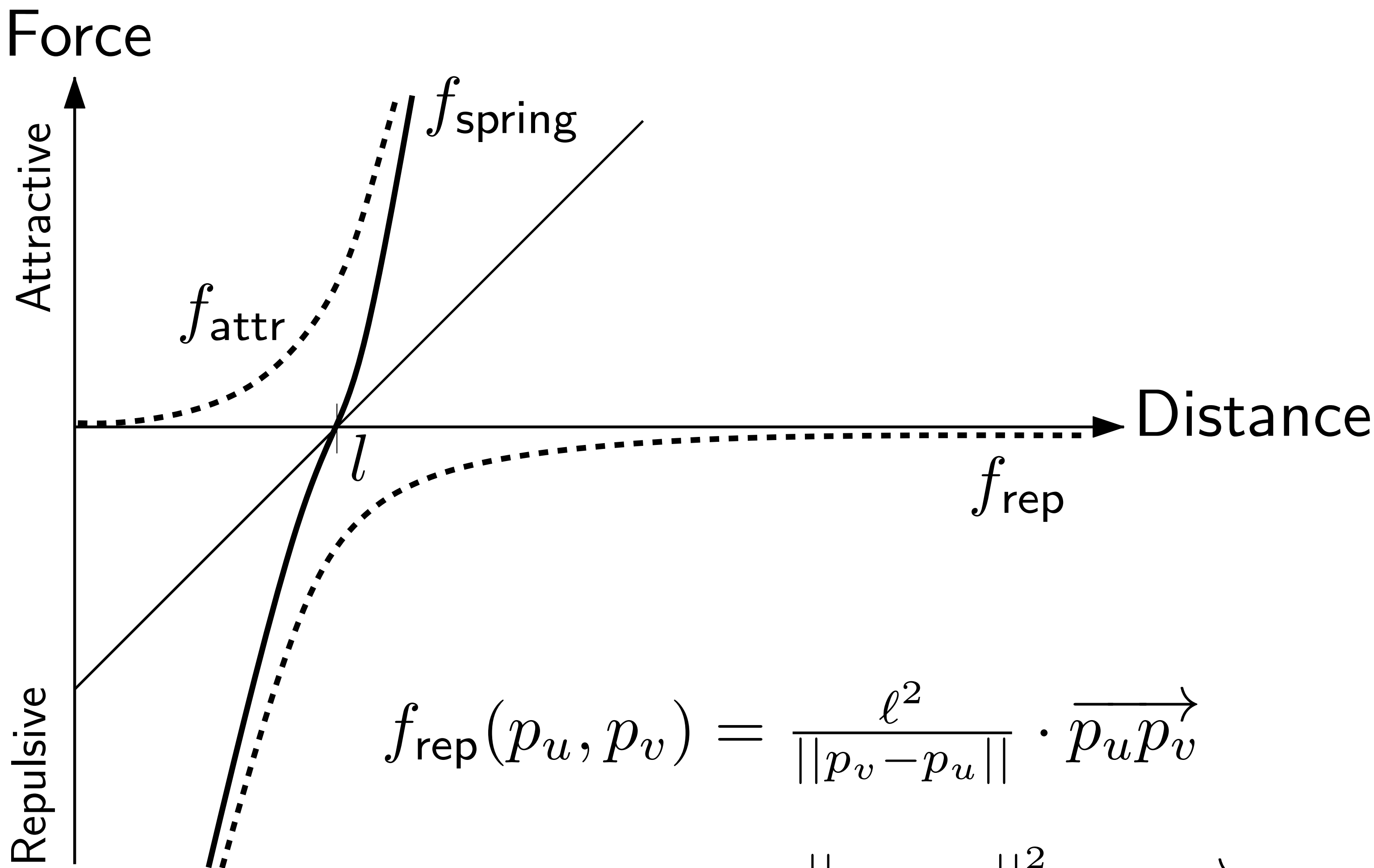
- attractive force between two adjacent nodes u and v

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

- resulting force between adjacent nodes u and v

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

Diagramm of Fruchtermann & Reingold Forces

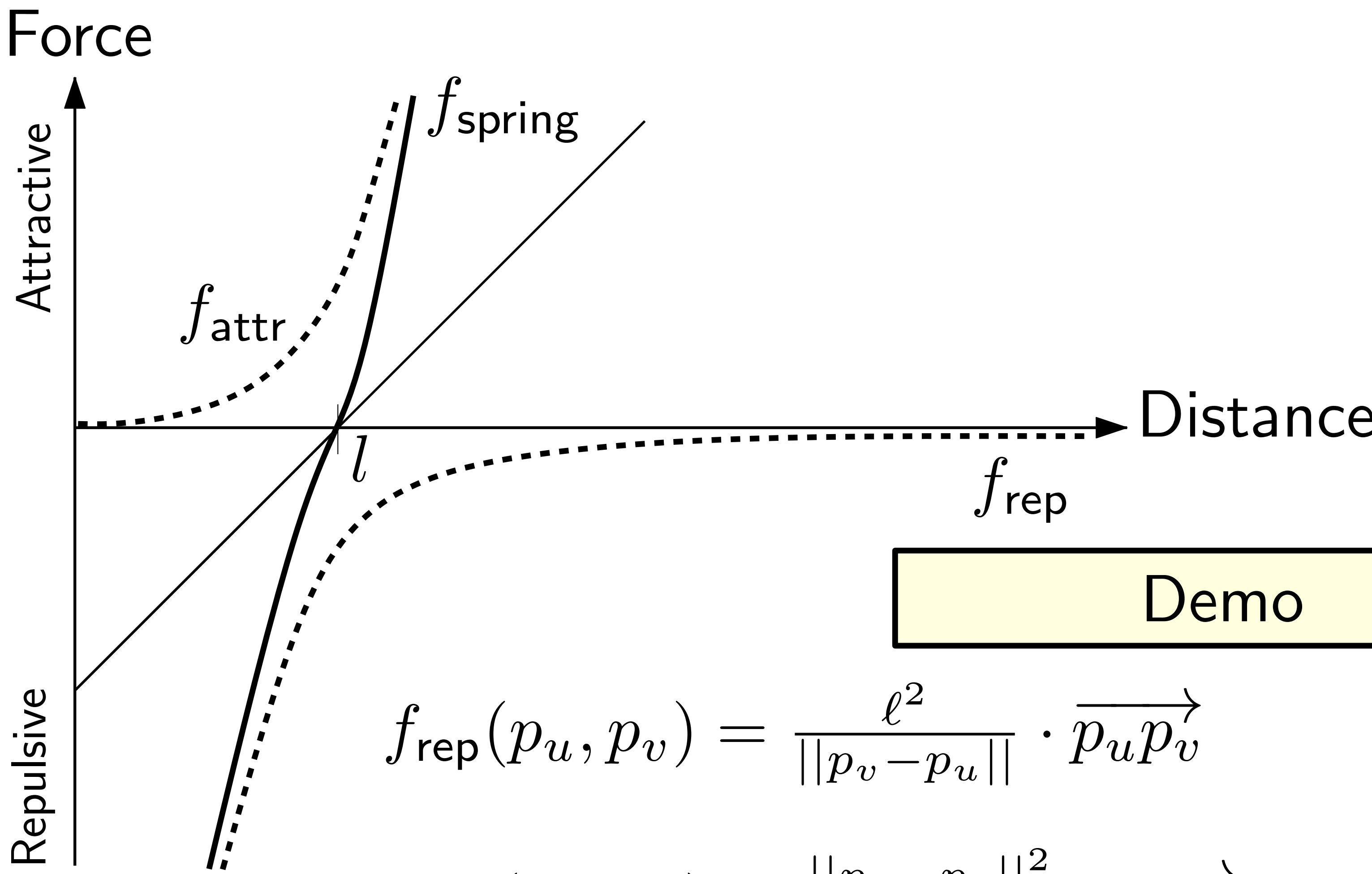


$$f_{\text{rep}}(p_u, p_v) = \frac{\ell^2}{||p_v - p_u||} \cdot \overrightarrow{p_u p_v}$$

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

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

Diagramm of Fruchtermann & Reingold Forces



Demo

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

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

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

Other Possible Modifications

- **Inertia**

- **Gravitation**

- **Magnetic forces**

Other Possible Modifications

- **Inertia**

  define node mass as $\Phi(v) = 1 + \deg(v)/2$

  set $f_{\mathsf{attr}}(p_u, p_v) \leftarrow f_{\mathsf{attr}}(p_u, p_v) \cdot 1/\Phi(v)$

- **Gravitation**

- **Magnetic forces**

Other Possible Modifications

- **Inertia**

  define node mass as $\Phi(v) = 1 + \deg(v)/2$

  set $f_{\text{attr}}(p_u, p_v) \leftarrow f_{\text{attr}}(p_u, p_v) \cdot 1/\Phi(v)$

- **Gravitation**

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

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

- **Magnetic forces**

Other Possible Modifications

- **Inertia**

  define node mass as $\Phi(v) = 1 + \deg(v)/2$

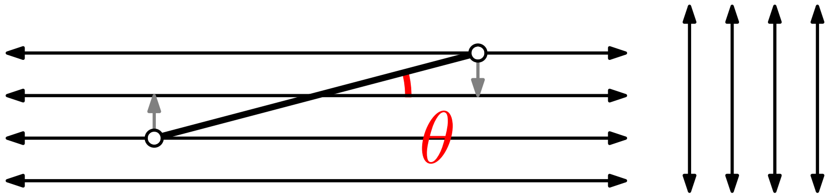
  set $f_{\text{attr}}(p_u, p_v) \leftarrow f_{\text{attr}}(p_u, p_v) \cdot 1/\Phi(v)$

- **Gravitation**

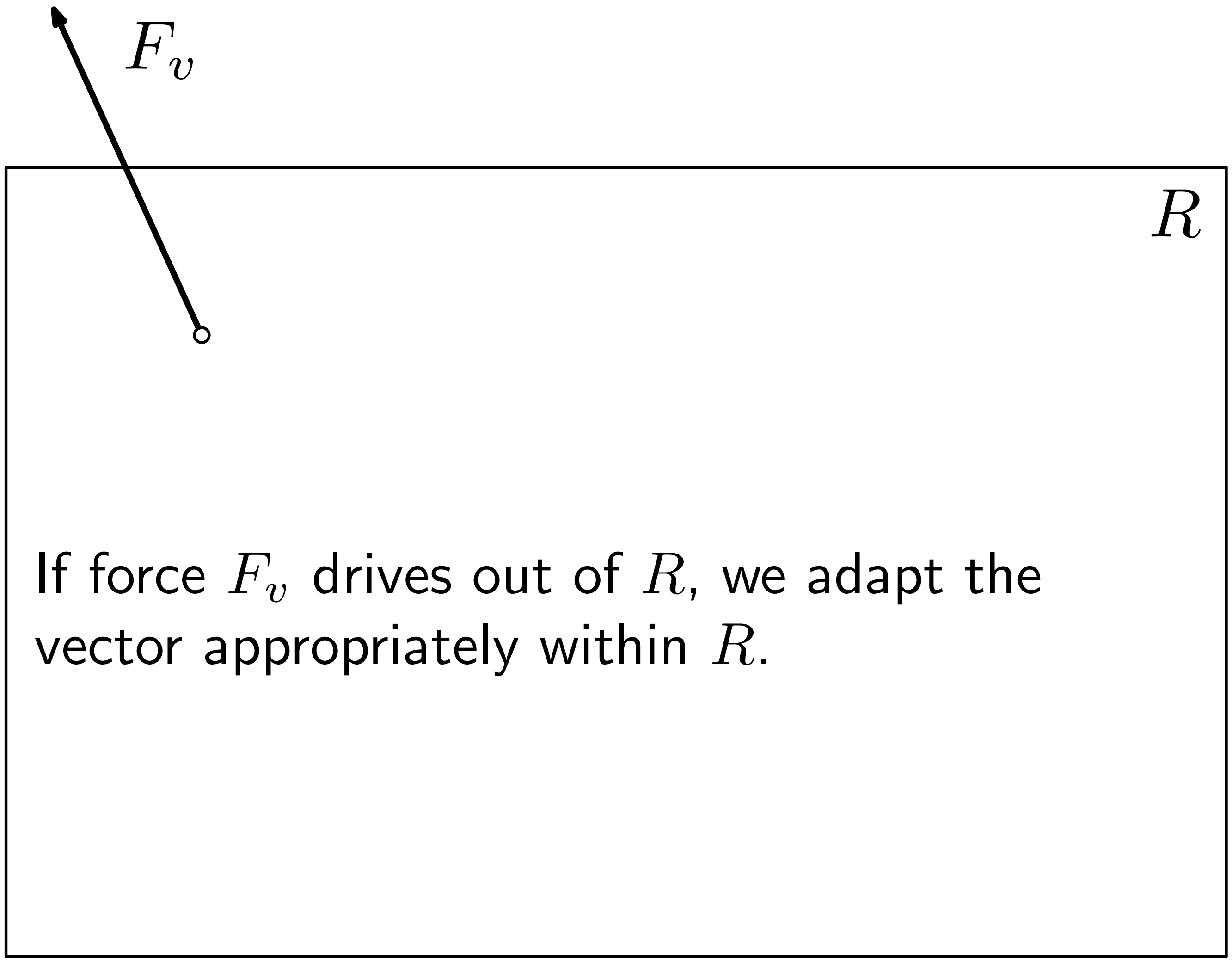
  define barycenter $p_{\text{bary}} = 1/|V| \cdot \sum_{v \in V} p_v$

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

- **Magnetic forces**
  - define magnetic fields (e.g. vertical, horizontal)
  - angle θ between edge and the direction of the field
  - define force that reduces this angle

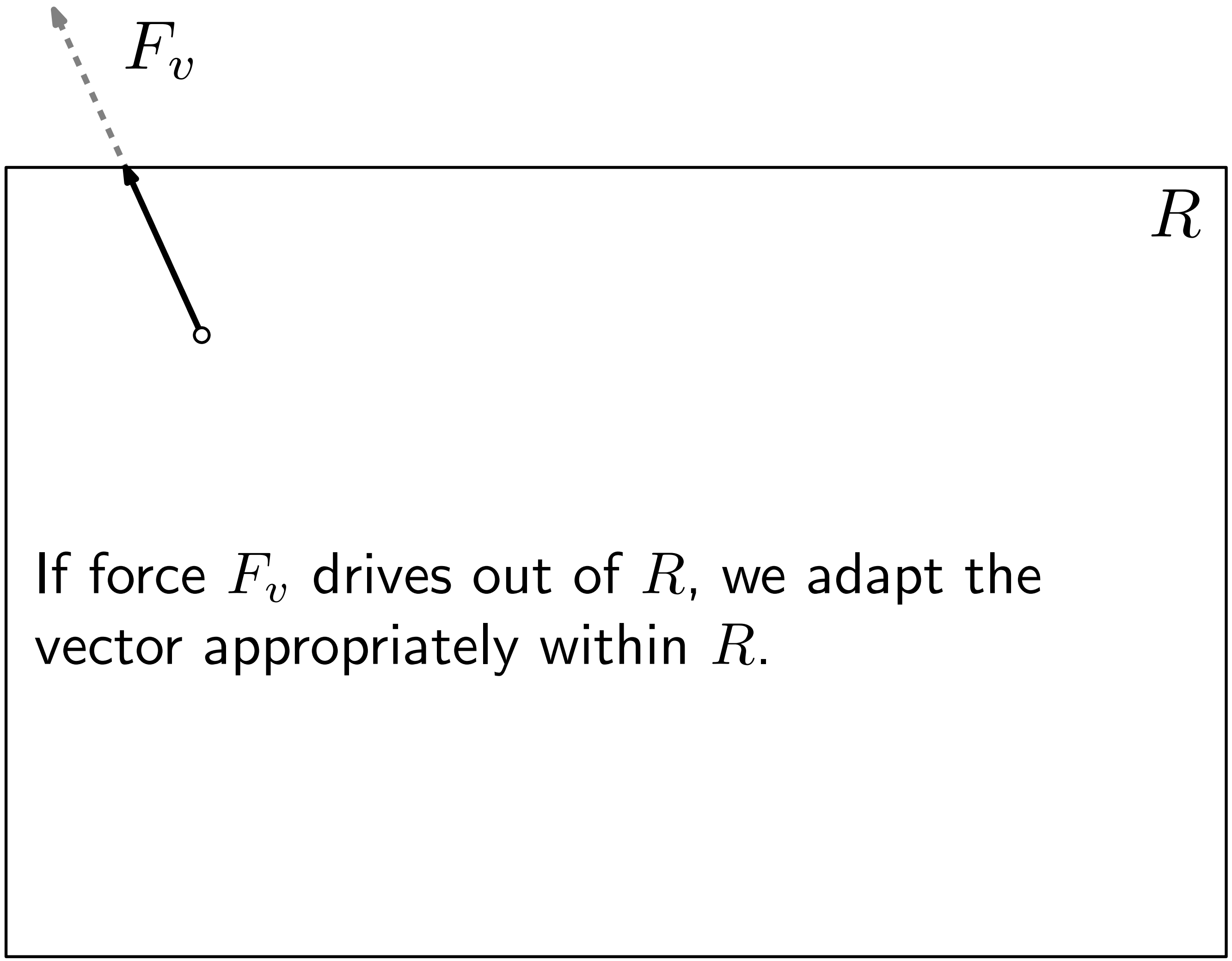


Bounded Drawing Area



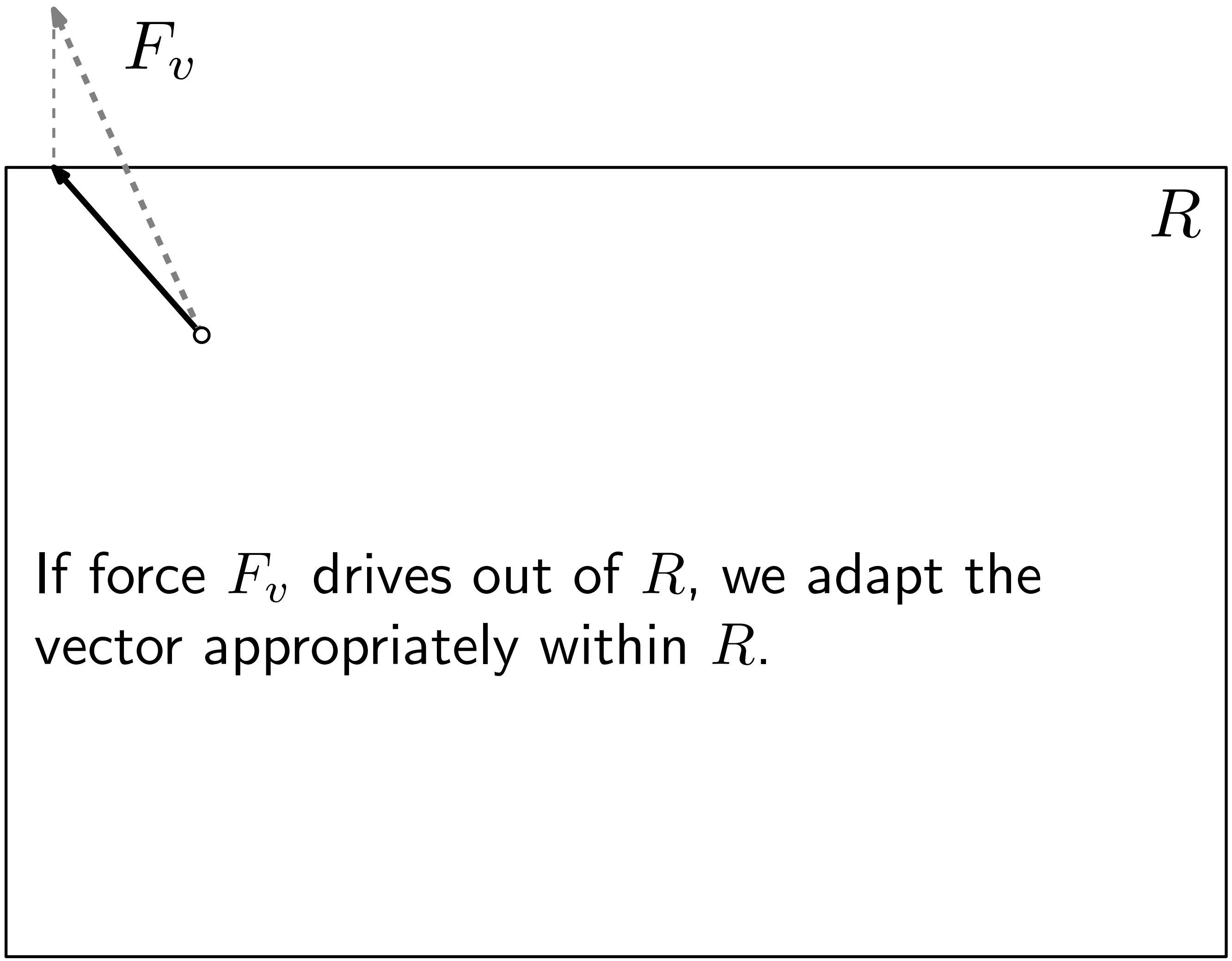
If force F_v drives out of R, we adapt the vector appropriately within R.

Bounded Drawing Area



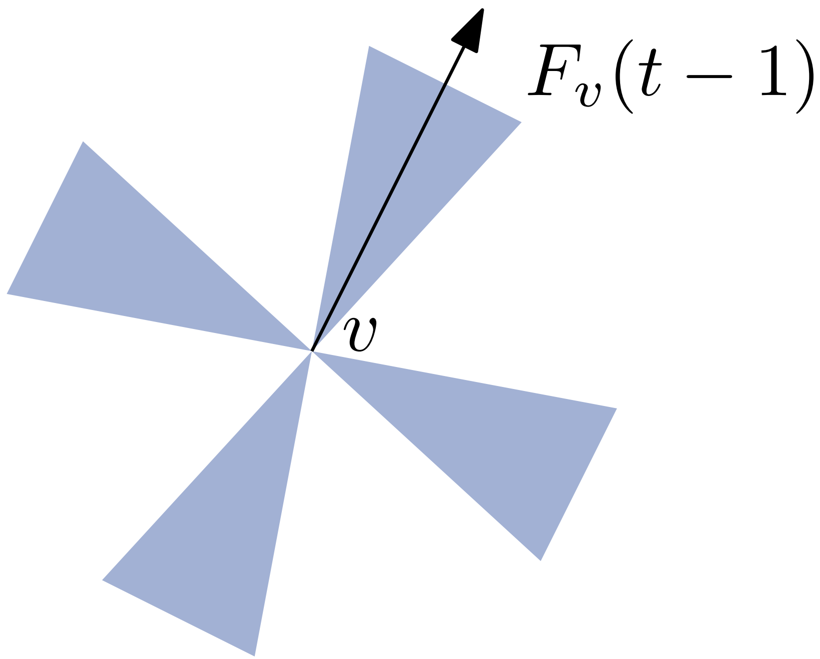
If force F_v drives out of R, we adapt the vector appropriately within R.

Bounded Drawing Area



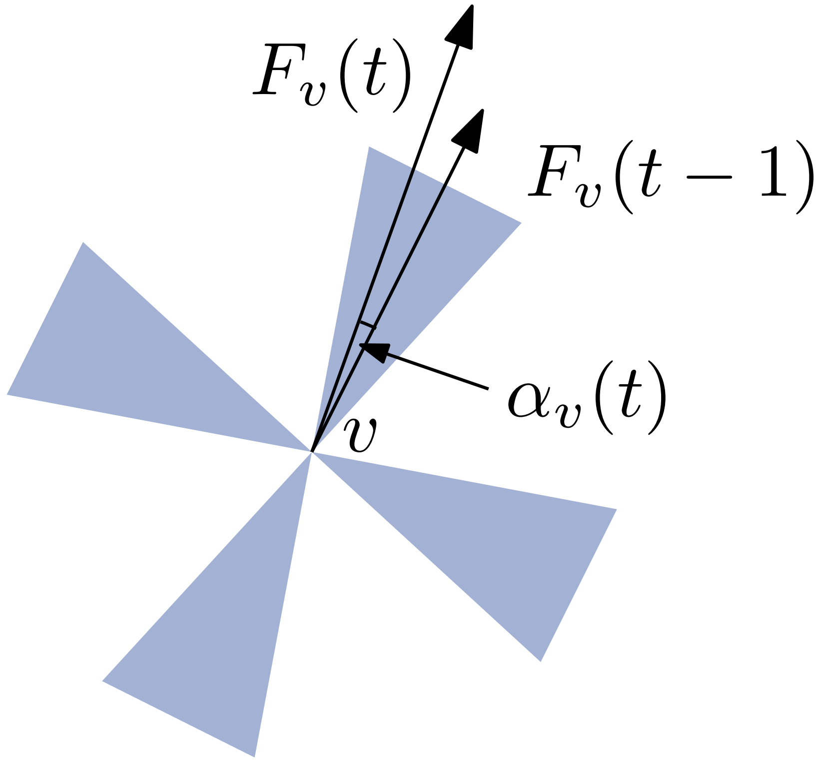
If force F_v drives out of R, we adapt the vector appropriately within R.

Adaptive Displacement (Frick, Ludwig, Mehldau 1995)



- store previous displacement vector $F_v(t-1)$

Adaptive Displacement (Frick, Ludwig, Mehldau 1995)

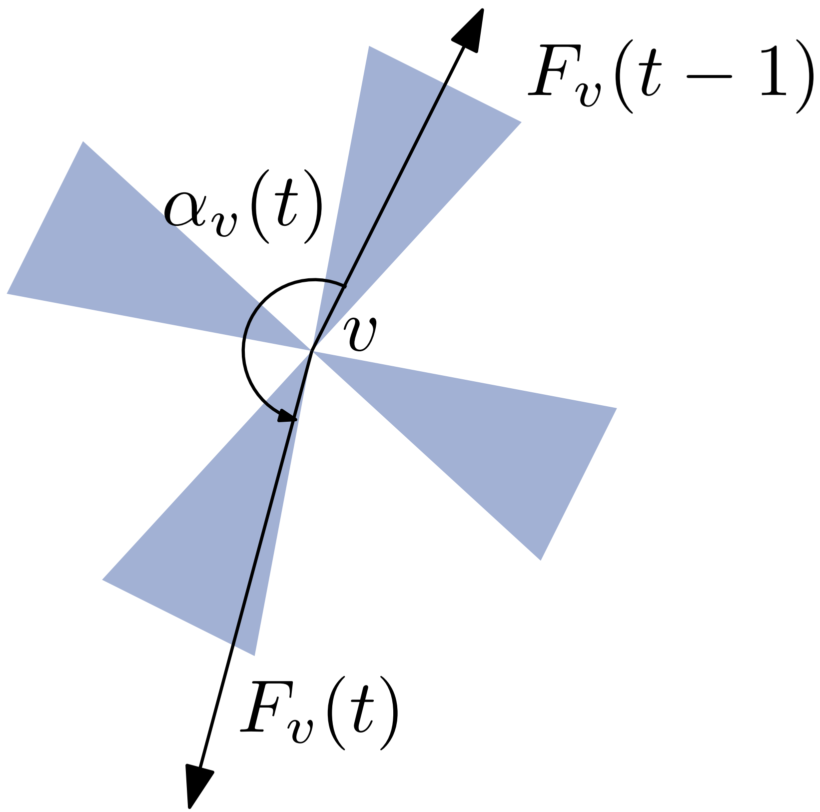


- store previous displacement vector $F_v(t-1)$

local temperature

- $\cos(\alpha_v(t)) \approx 1$:
  similar direction
  $\rightarrow$ increase the temperature

Adaptive Displacement (Frick, Ludwig, Mehldau 1995)

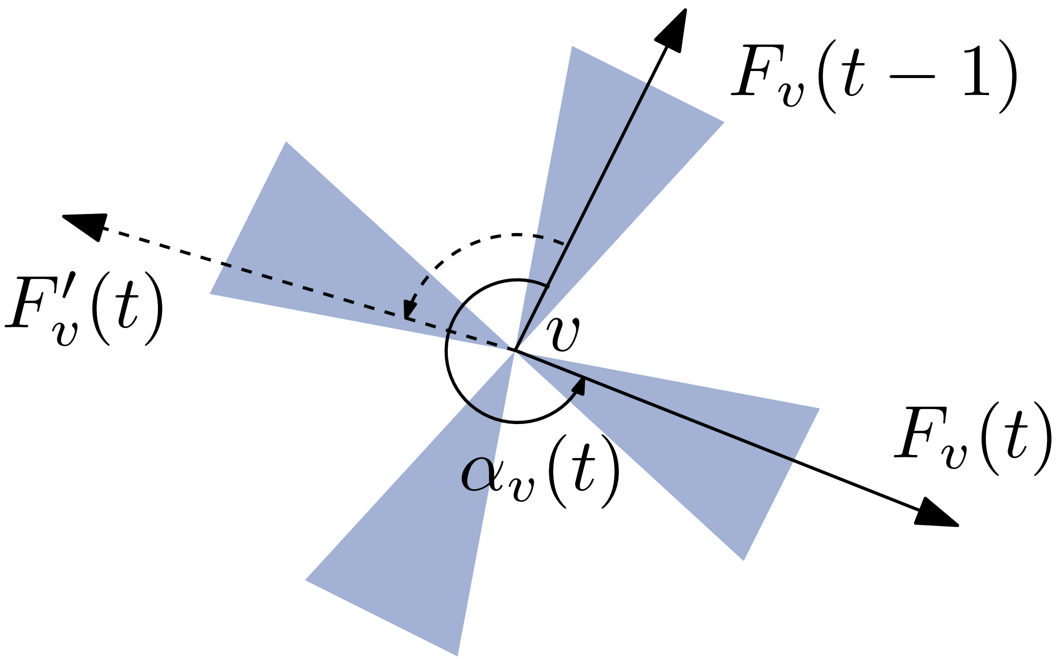


- store previous displacement vector $F_v(t-1)$

local temperature

- $\cos(\alpha_v(t)) \approx 1$:
  similar direction
  $\rightarrow$ increase the temperature
- $\cos(\alpha_v(t)) \approx -1$:
  oscillation
  $\rightarrow$ reduce the temperature

Adaptive Displacement (Frick, Ludwig, Mehldau 1995)



- store previous displacement vector $F_v(t-1)$

local temperature

- $\cos(\alpha_v(t)) \approx 1$:
  similar direction
  $\rightarrow$ increase the temperature
- $\cos(\alpha_v(t)) \approx -1$:
  oscillation
  $\rightarrow$ reduce the temperature
- $\cos(\alpha_v(t)) \approx 0$:
  Rotation
  $\rightarrow$ update rotation counter and decrease temperature if necessary

Discussion

Advantages

- still very simple algorithm
- further modifications improve layout quality and lead to faster convergence

Discussion

Advantages

- still very simple algorithm
- further modifications improve layout quality and lead to faster convergence

Disadvantages

- stability is not guaranteed
- local minima possible
- quadratic time for repulsive forces

Discussion

Advantages

- still very simple algorithm
- further modifications improve layout quality and lead to faster convergence

Disadvantages

- stability is not guaranteed
- local minima possible
- quadratic time for repulsive forces

Influence

- Variants of Fruchterman and Reingold algorithms are probably the most popular force-based methods (original paper cited 4500 times (2000 in the past two years))

Discussion

Advantages

- still very simple algorithm
- further modifications improve layout quality and lead to faster convergence

Disadvantages

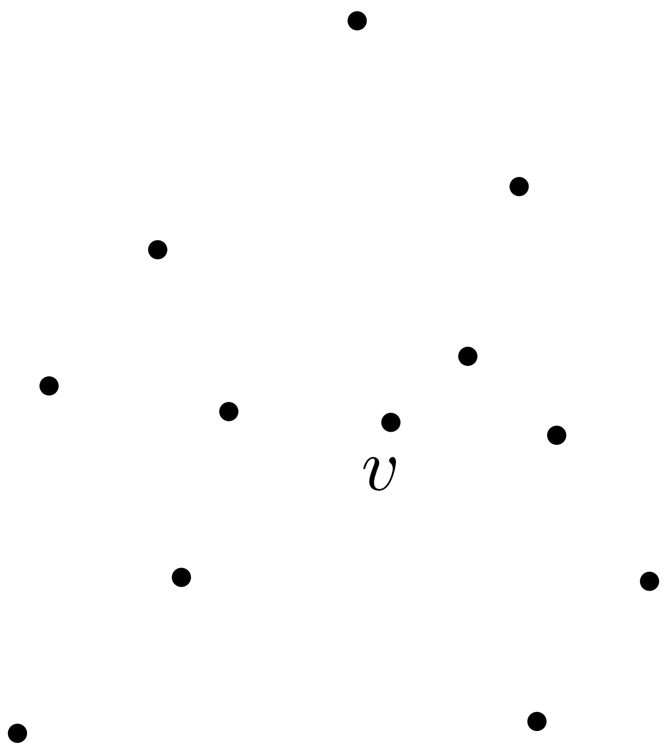
- stability is not guaranteed
- local minima possible
- quadratic time for repulsive forces

Could we reduce this?

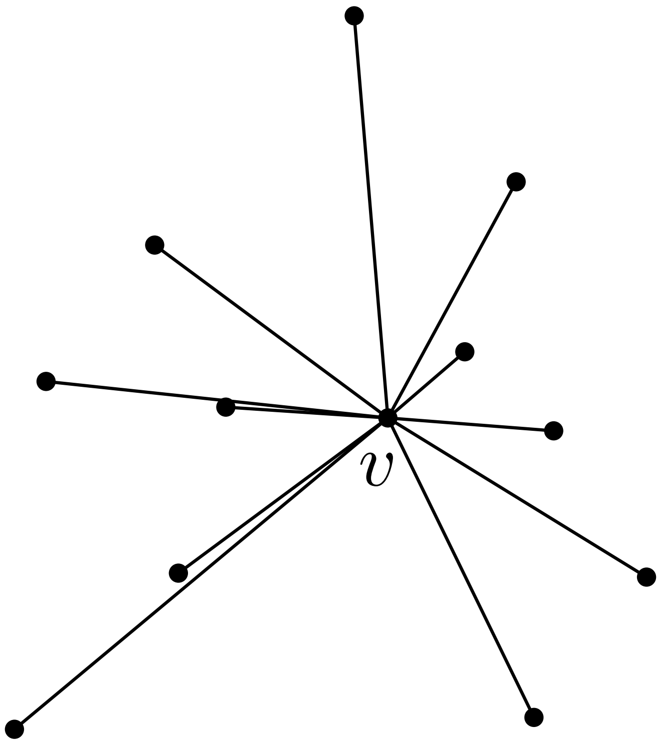
Influence

- Variants of Fruchterman and Reingold algorithms are probably the most popular force-based methods (original paper cited 4500 times (2000 in the past two years))

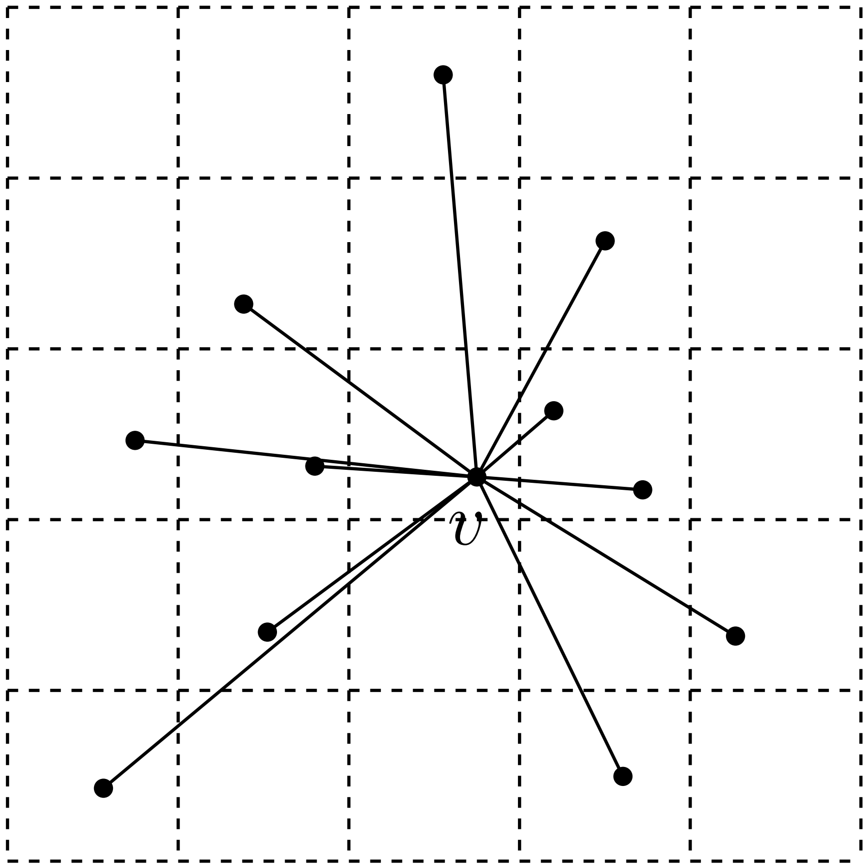
Grid Version (Fruchterman, Reingold, 1990)



Grid Version (Fruchterman, Reingold, 1990)

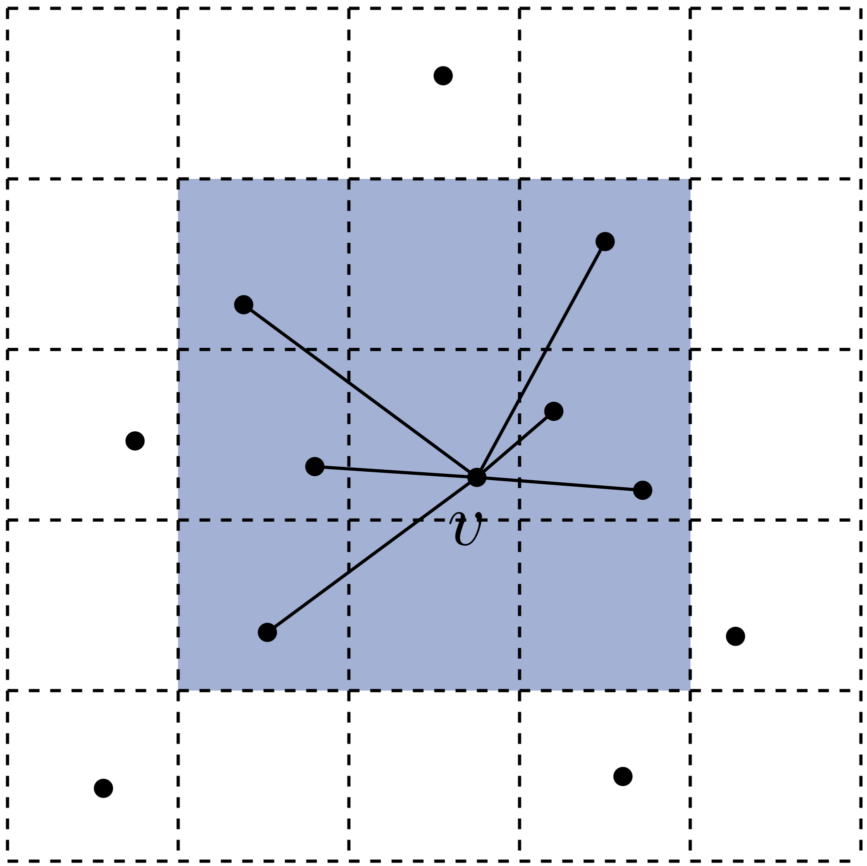


Grid Version (Fruchterman, Reingold, 1990)



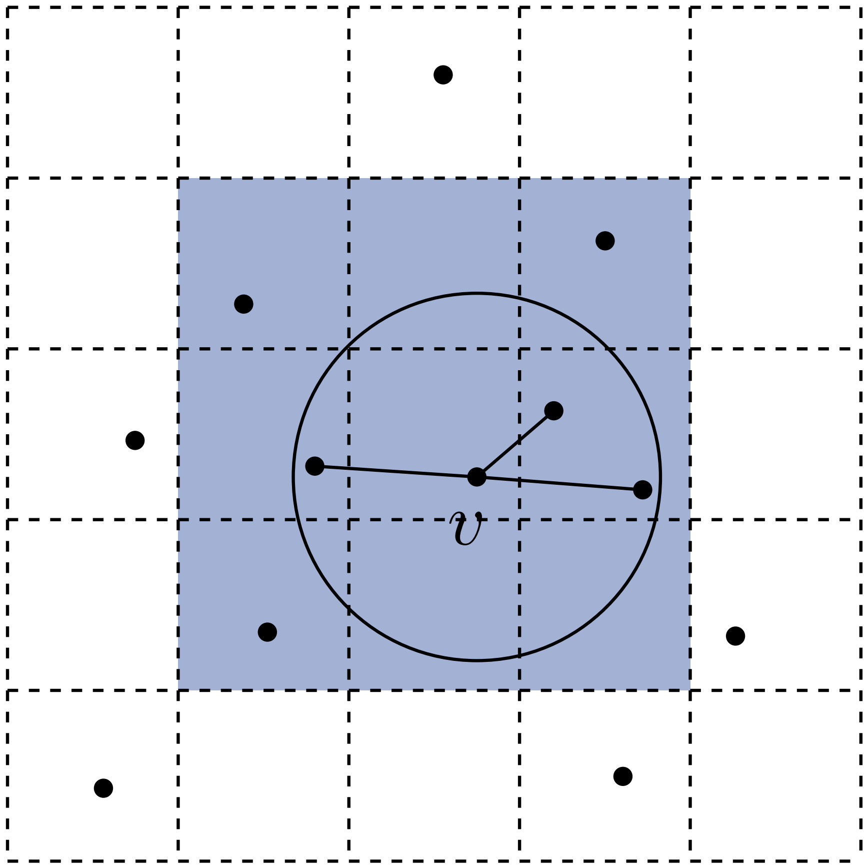
- subdivide plane by a gird

Grid Version (Fruchterman, Reingold, 1990)



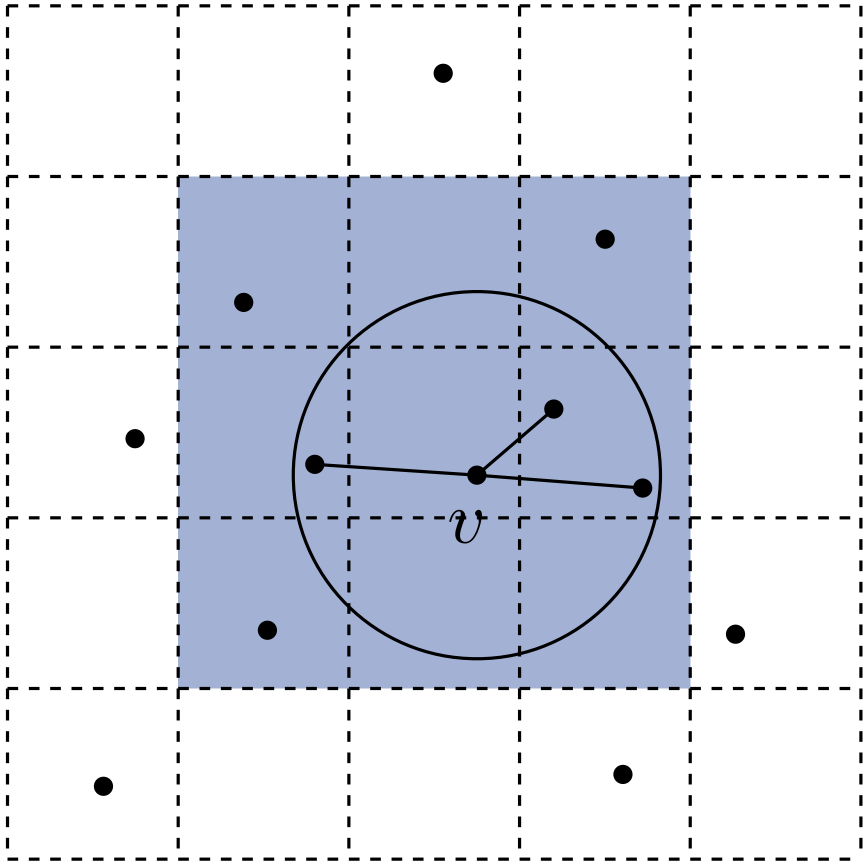
- subdivide plane by a gird
- compute repulsive forces only for the nodes in the neighbouring cells

Grid Version (Fruchterman, Reingold, 1990)



- subdivide plane by a gird
- compute repulsive forces only for the nodes in the neighbouring cells
- and only when the distance is at most $d_{\max}$

Grid Version (Fruchterman, Reingold, 1990)

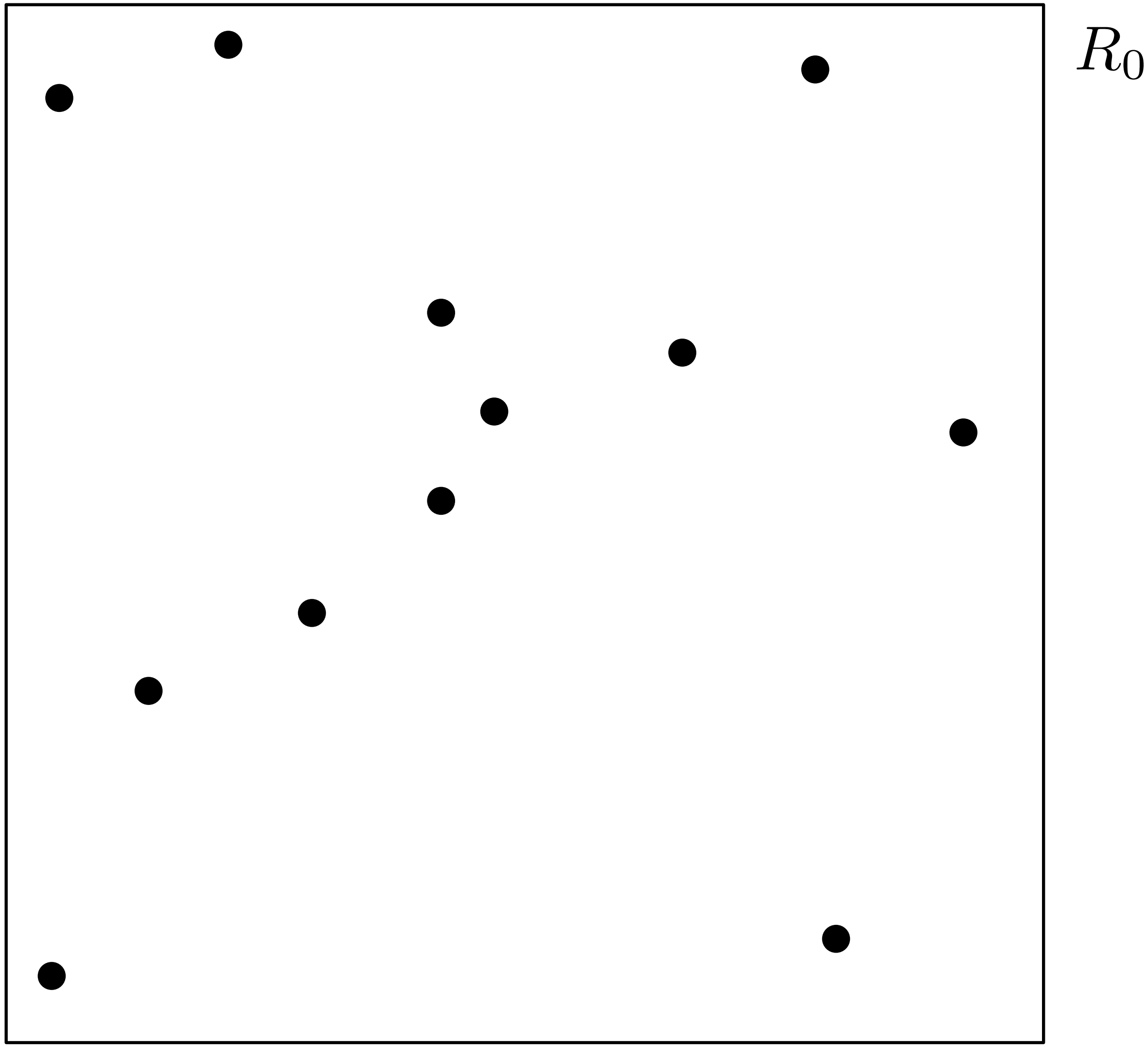


- subdivide plane by a gird
- compute repulsive forces only for the nodes in the neighbouring cells
- and only when the distance is at most $d_{\max}$

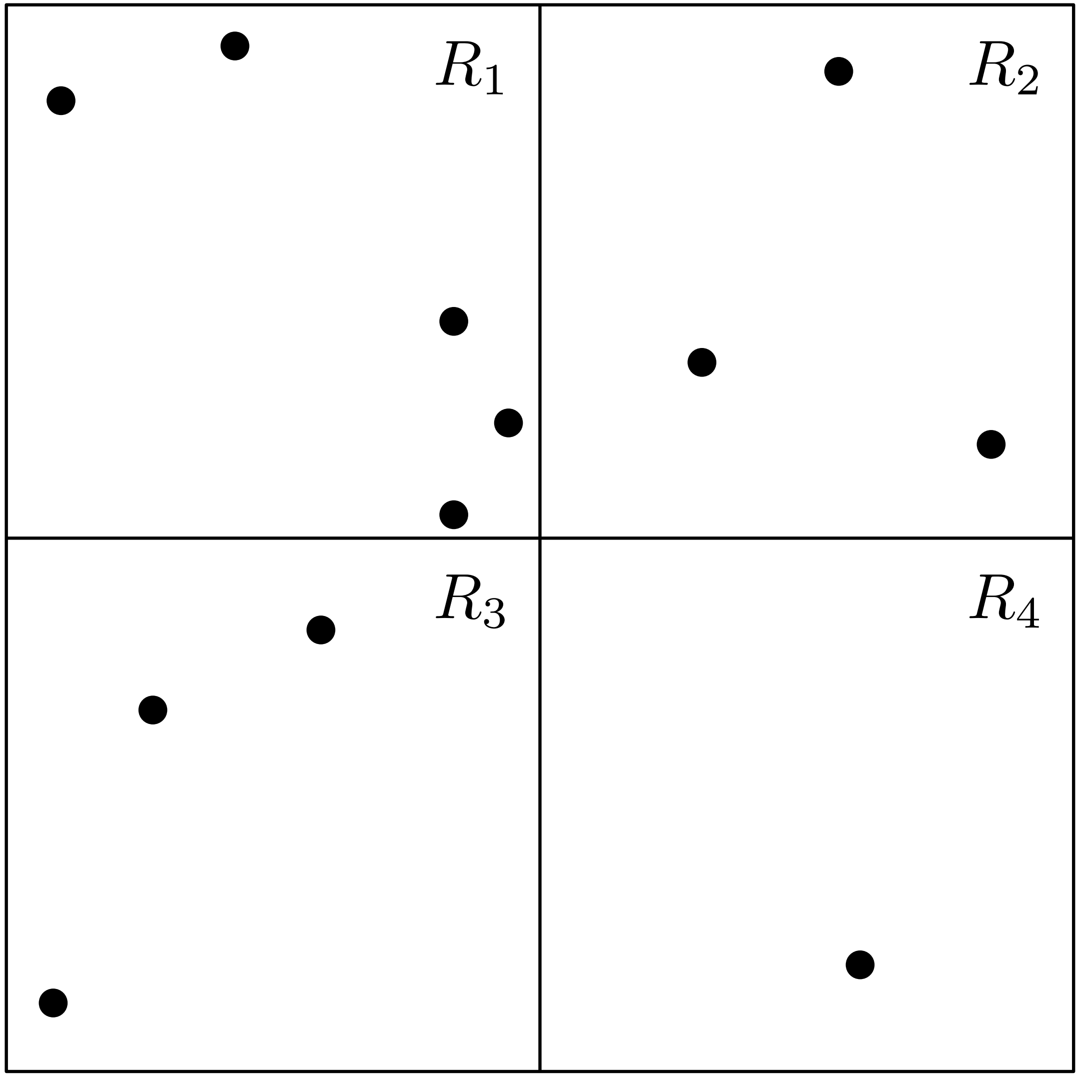
Discussion

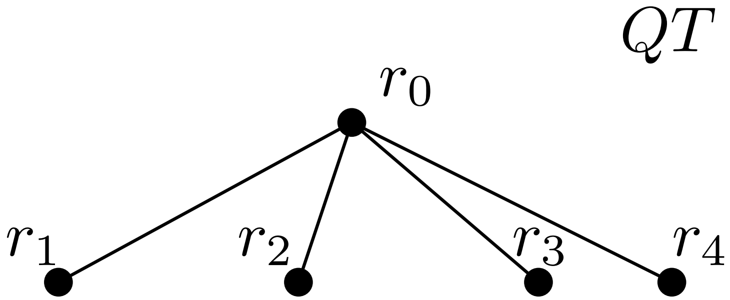
- meaningful idea to improve runtime
- worst-case no advantage
- Quality loss

Quad-Tree

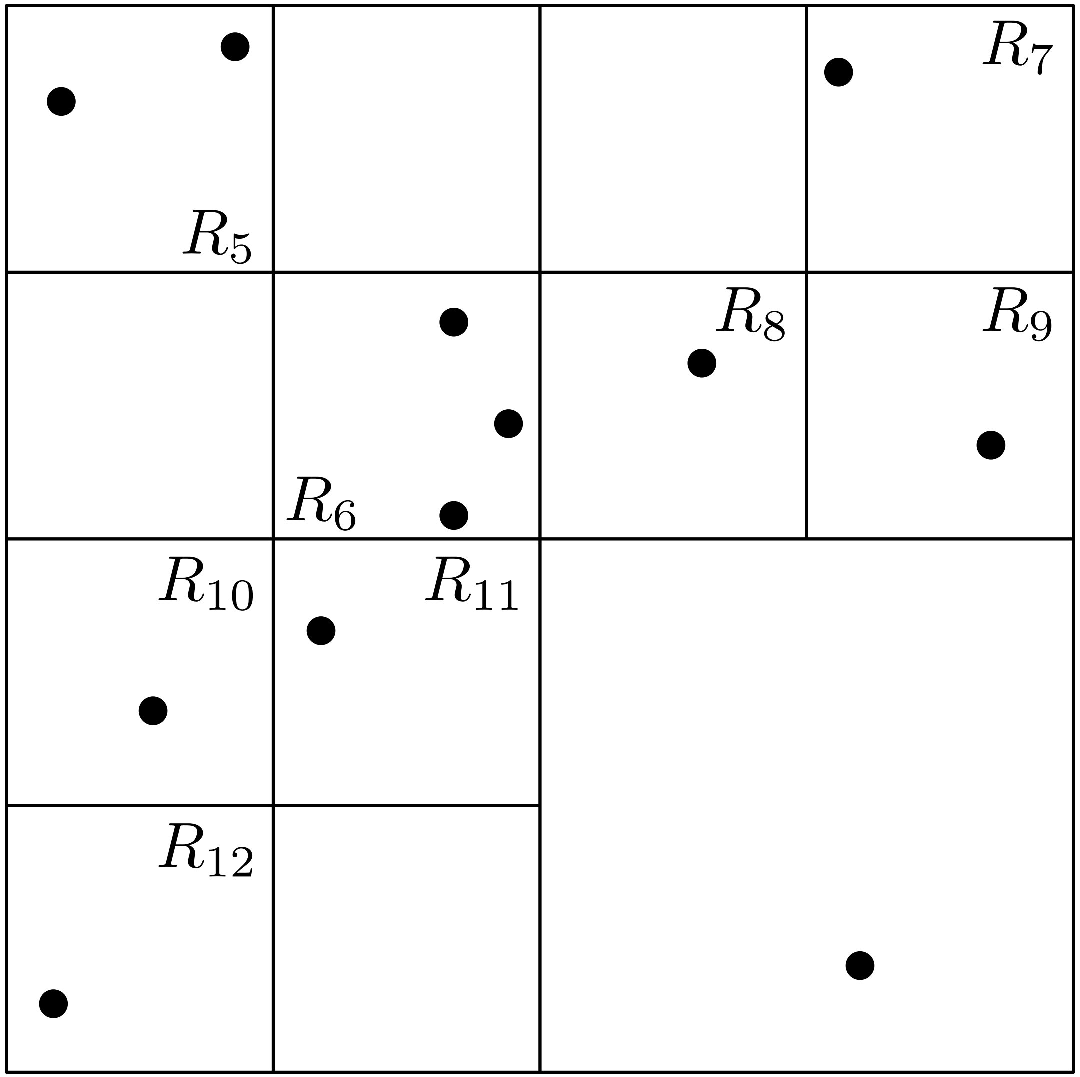


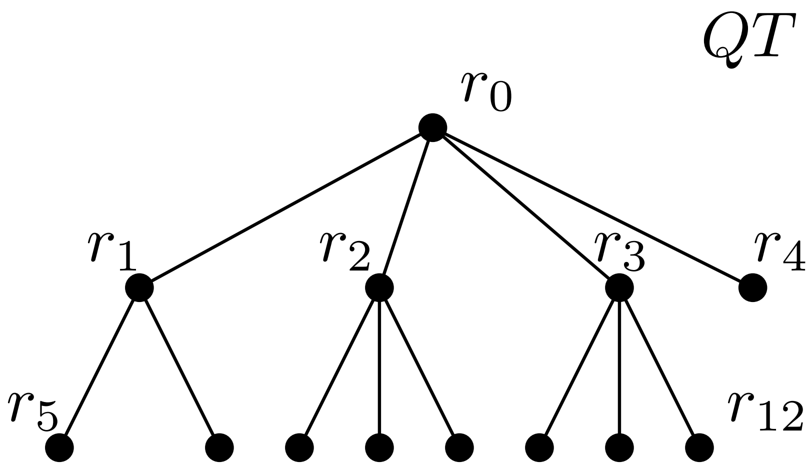
Quad-Tree



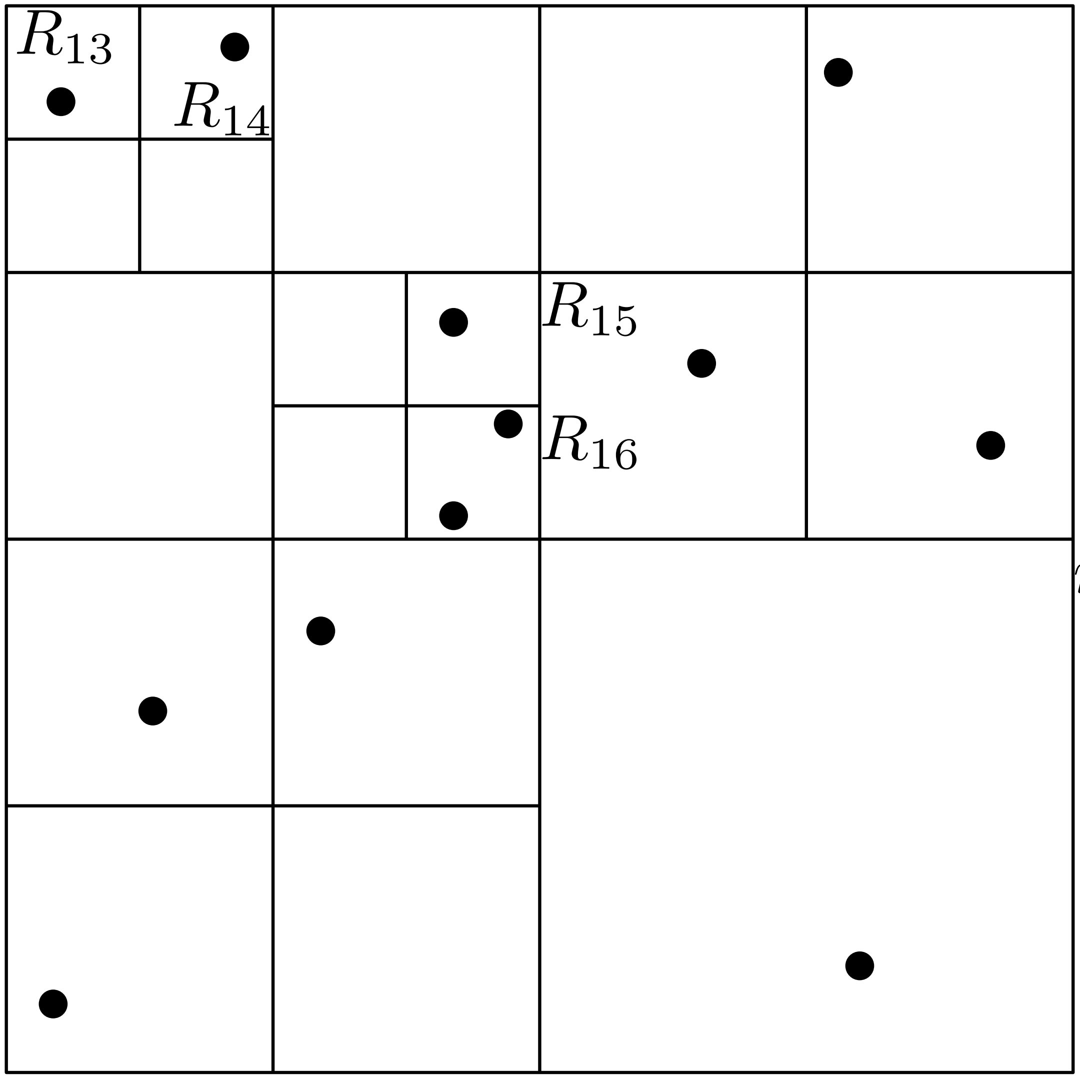


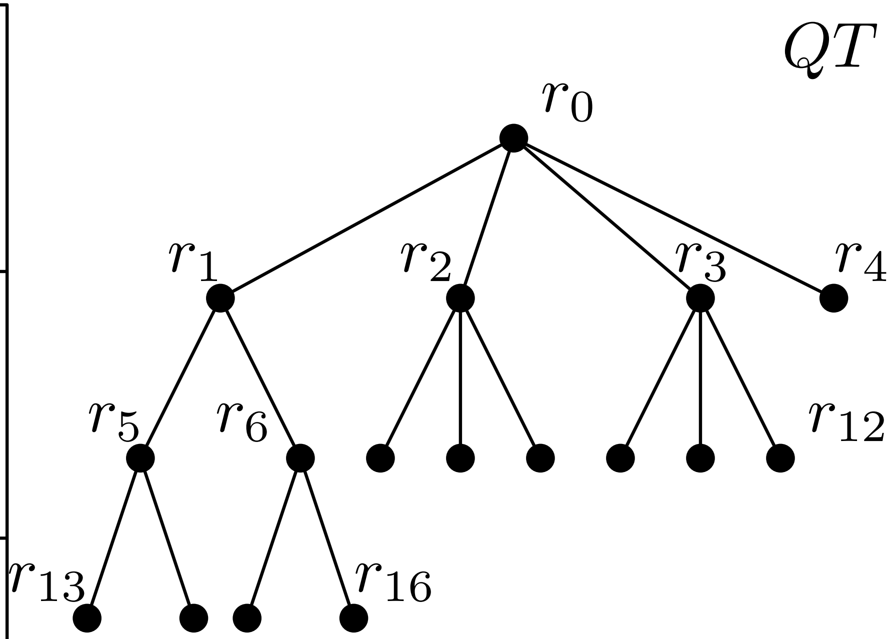
Quad-Tree



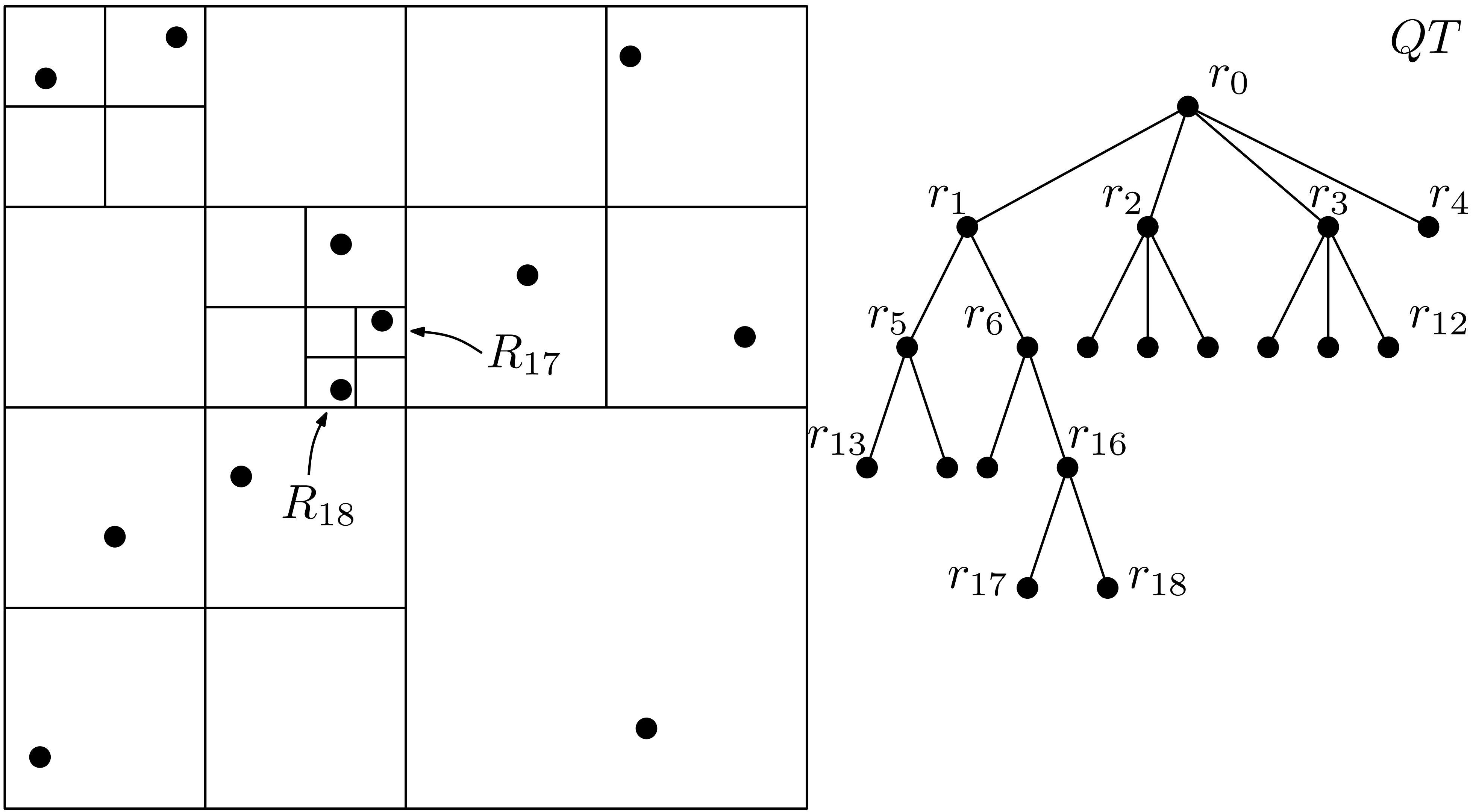


Quad-Tree



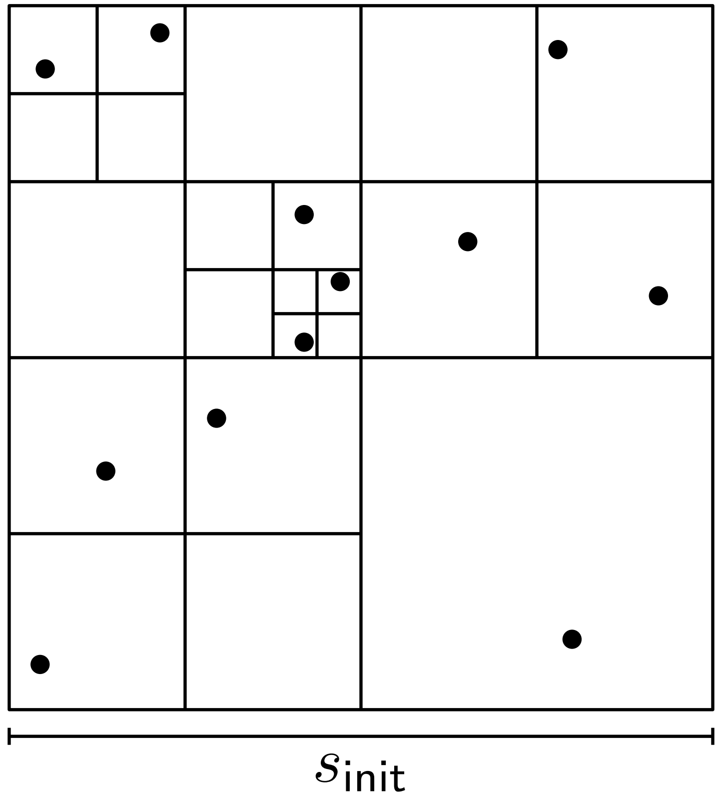


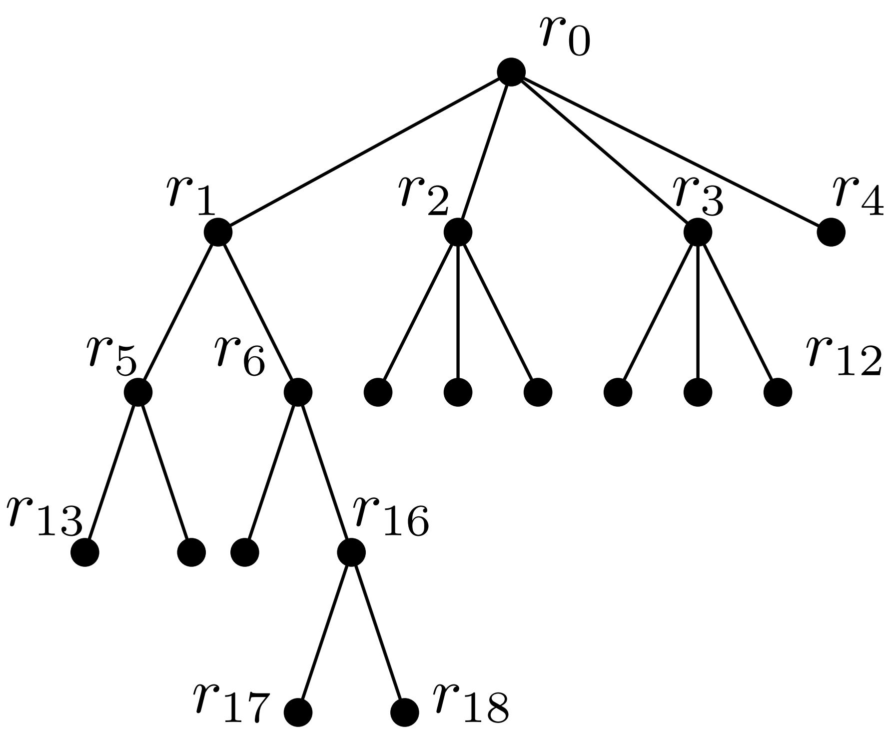
Quad-Tree



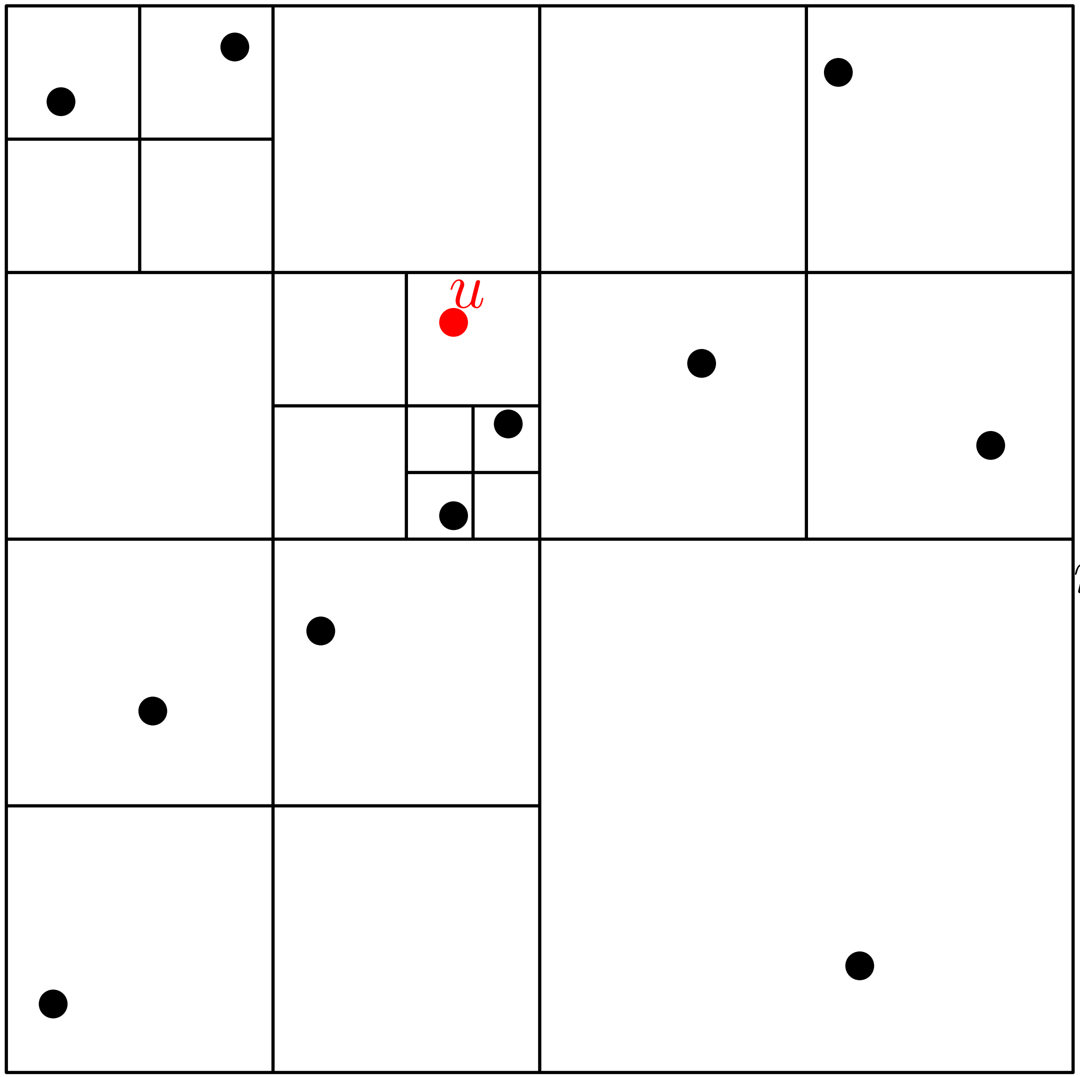
Properties of Quad-Tree

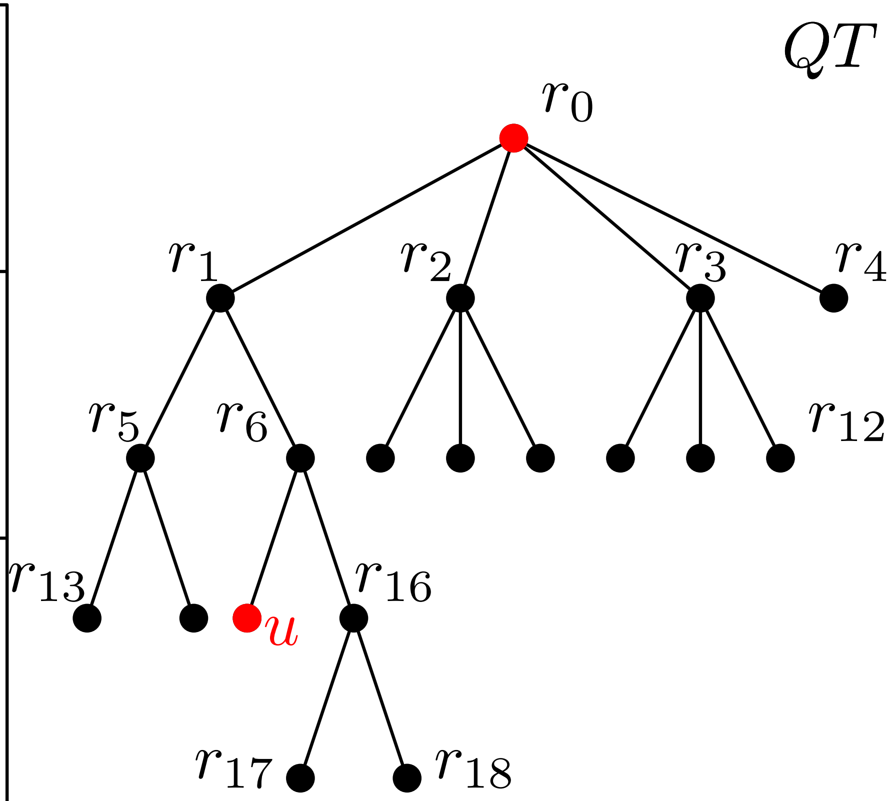
- height $h \leq \log \frac{s_{\text{init}}}{d_{\text{min}}} + \frac{3}{2}$, here d_{min}-smallest distance
- time and space $O(hn)$
- *compressed* quad-tree in $O(n \log n)$ time





Forces with Quad-Trees (Barnes, Hut, 1986)



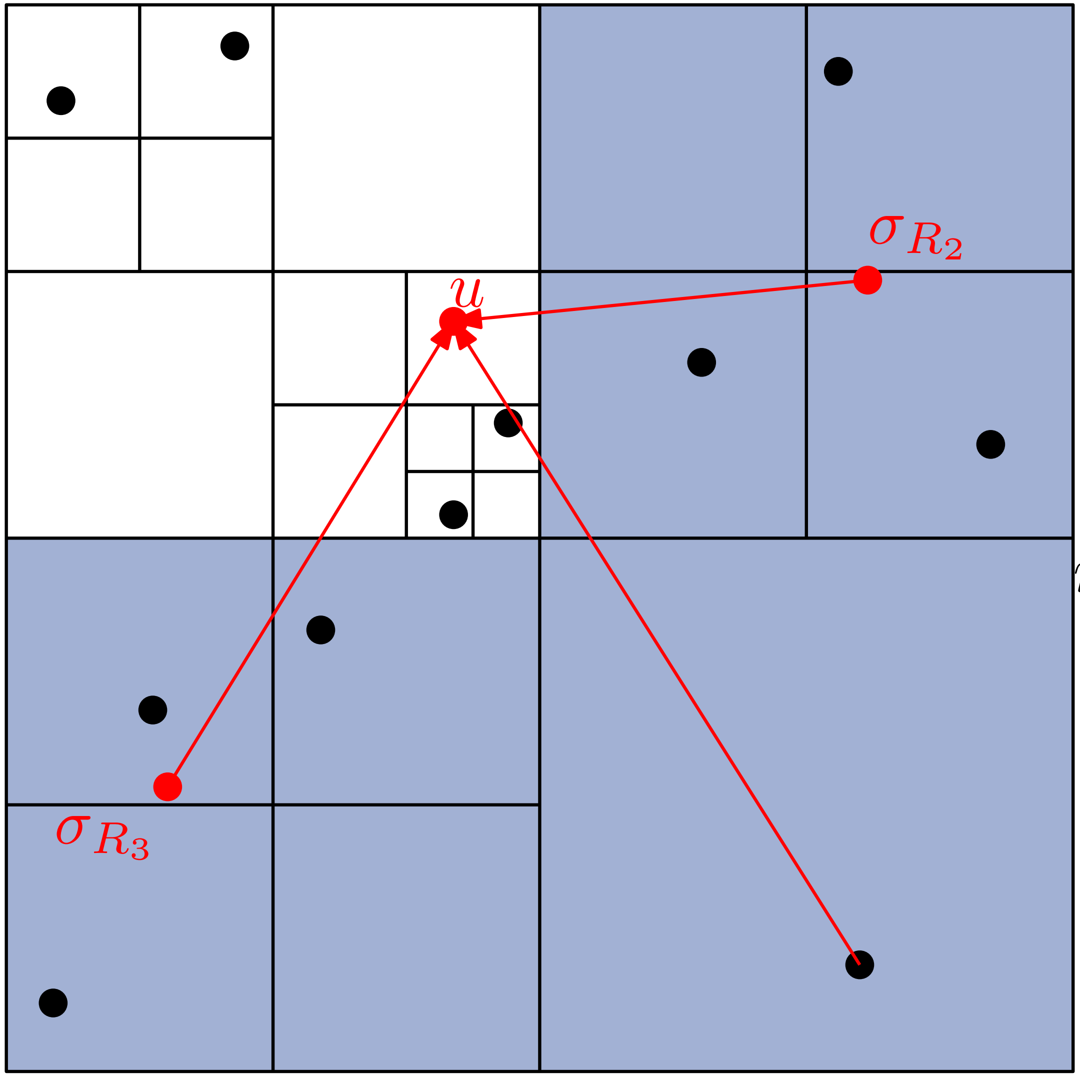


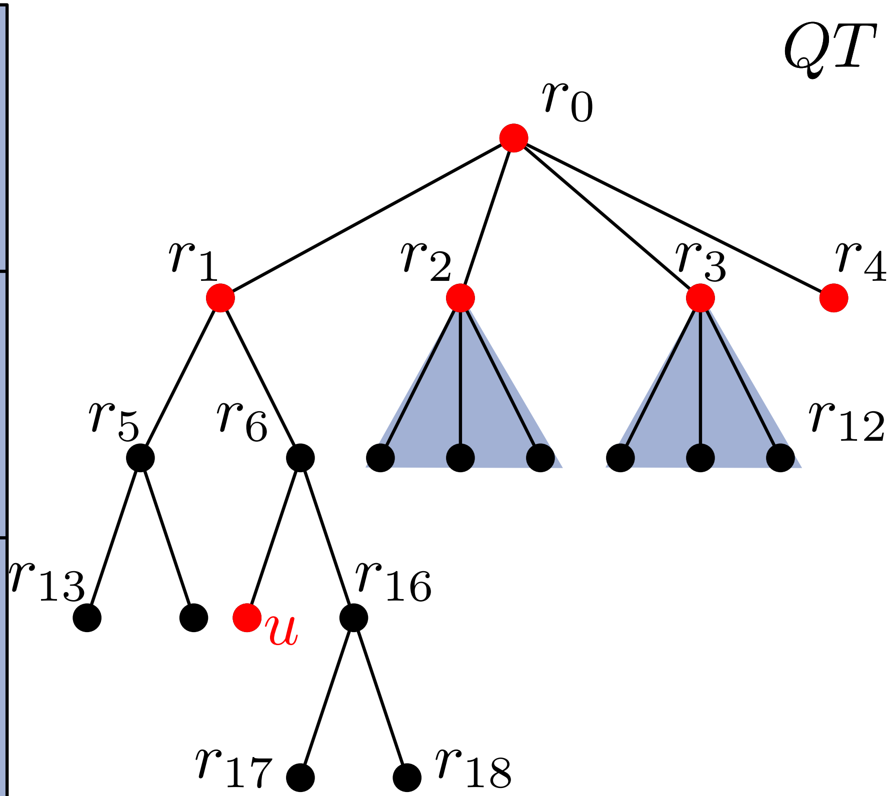
$$f_{rep}(R_i, u) := |R_i| \cdot f_{rep}(p_{R_i}, u)$$

, where p_{R_i} - the barycentre of the points in R_i

Compute repulsive force between u and the barycentre of the vertices in the corresponding square

Forces with Quad-Trees (Barnes, Hut, 1986)

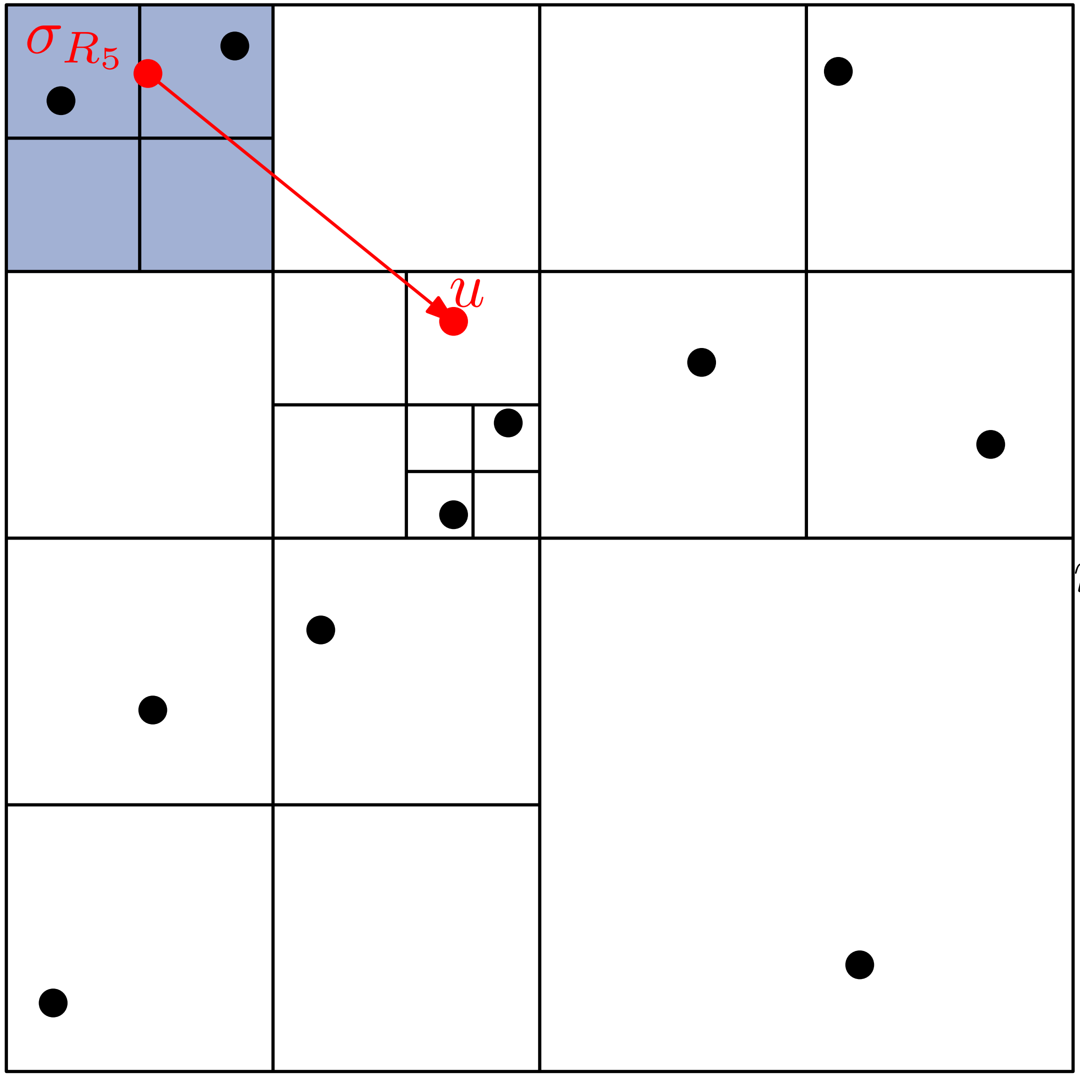


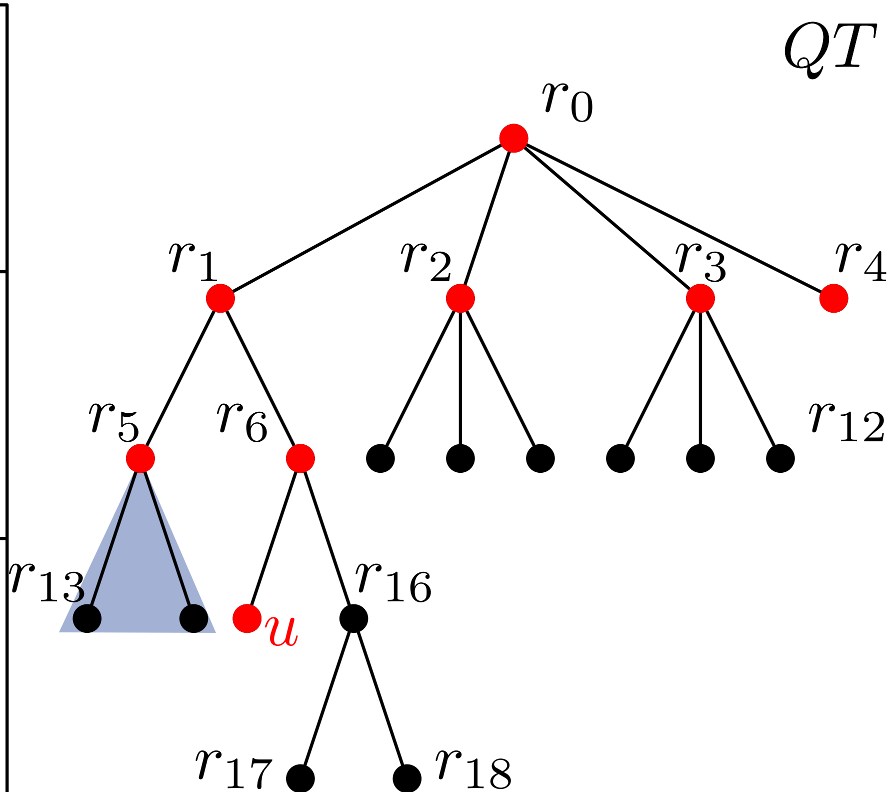


$f_{rep}(R_i, u) := |R_i| \cdot f_{rep}(p_{R_i}, u)$, where p_{R_i} - the barycentre of the points in R_i

Compute repulsive force between u and the barycentre of the vertices in the corresponding square

Forces with Quad-Trees (Barnes, Hut, 1986)

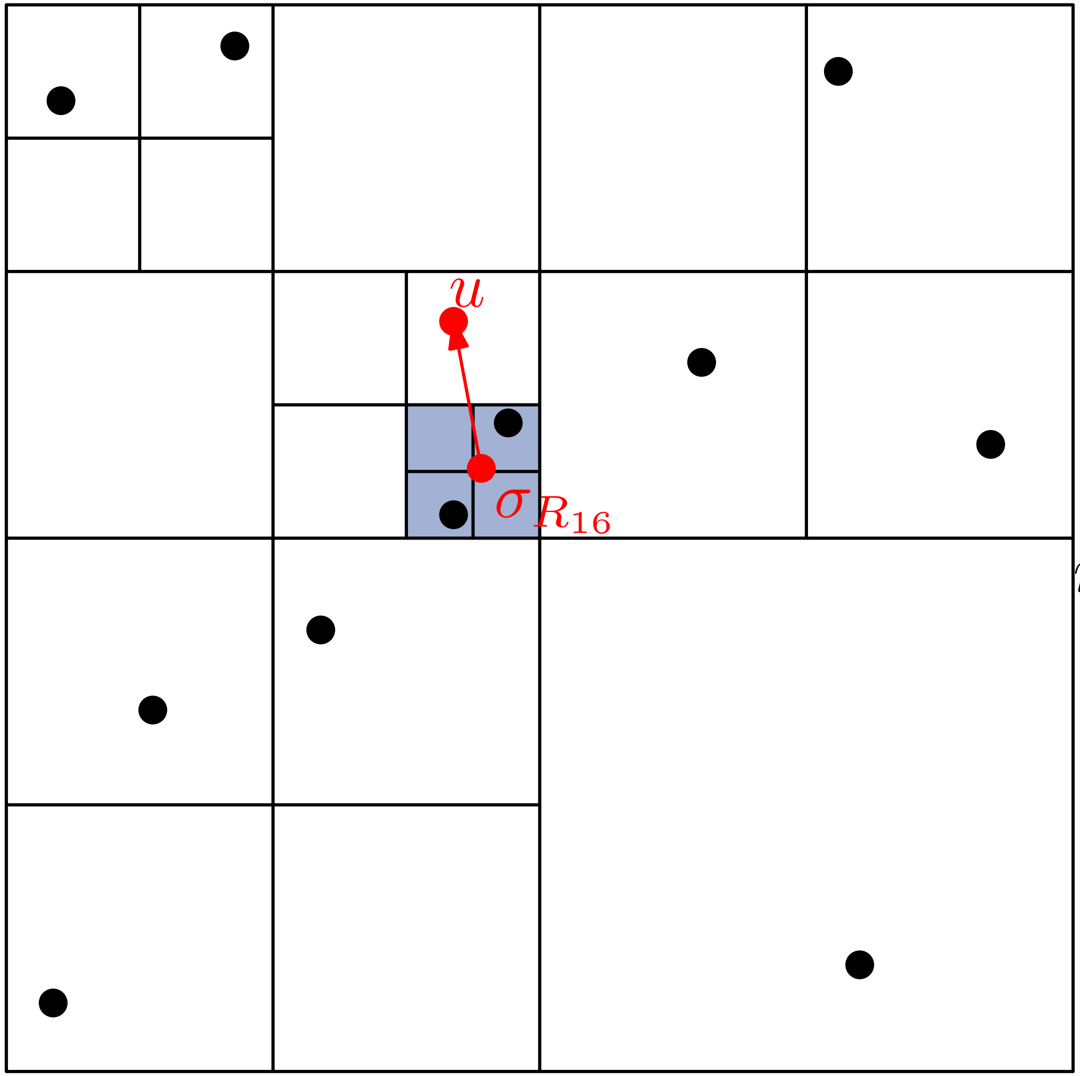


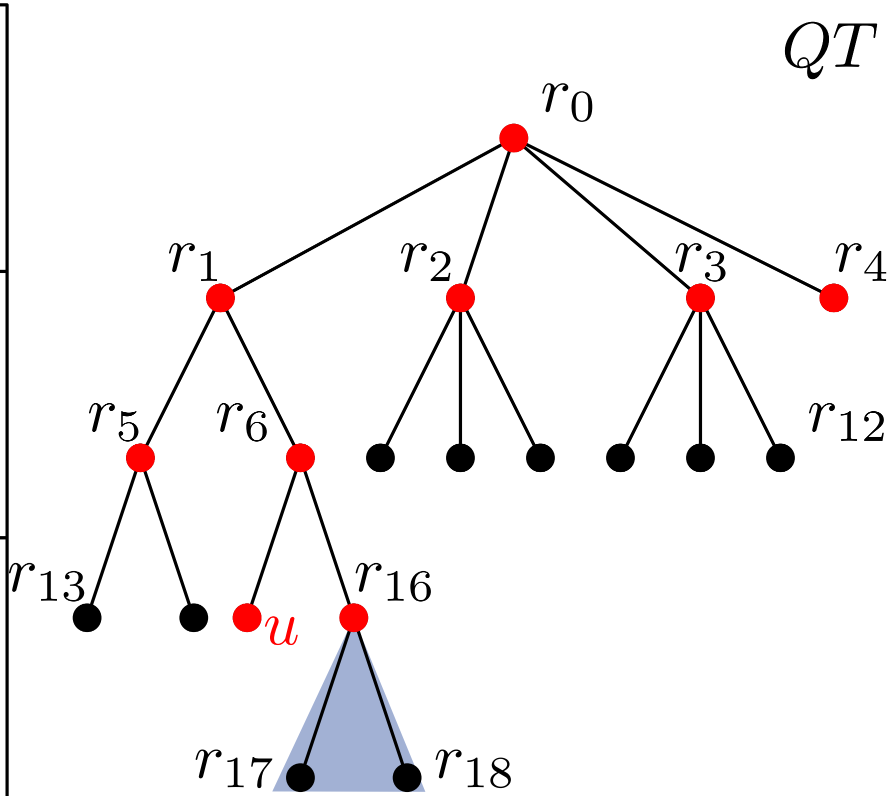


$f_{rep}(R_i, u) := |R_i| \cdot f_{rep}(p_{R_i}, u)$
, where p_{R_i} - the barycentre of the points in R_i

Compute repulsive force between u and the barycentre of the vertices in the corresponding square

Forces with Quad-Trees (Barnes, Hut, 1986)

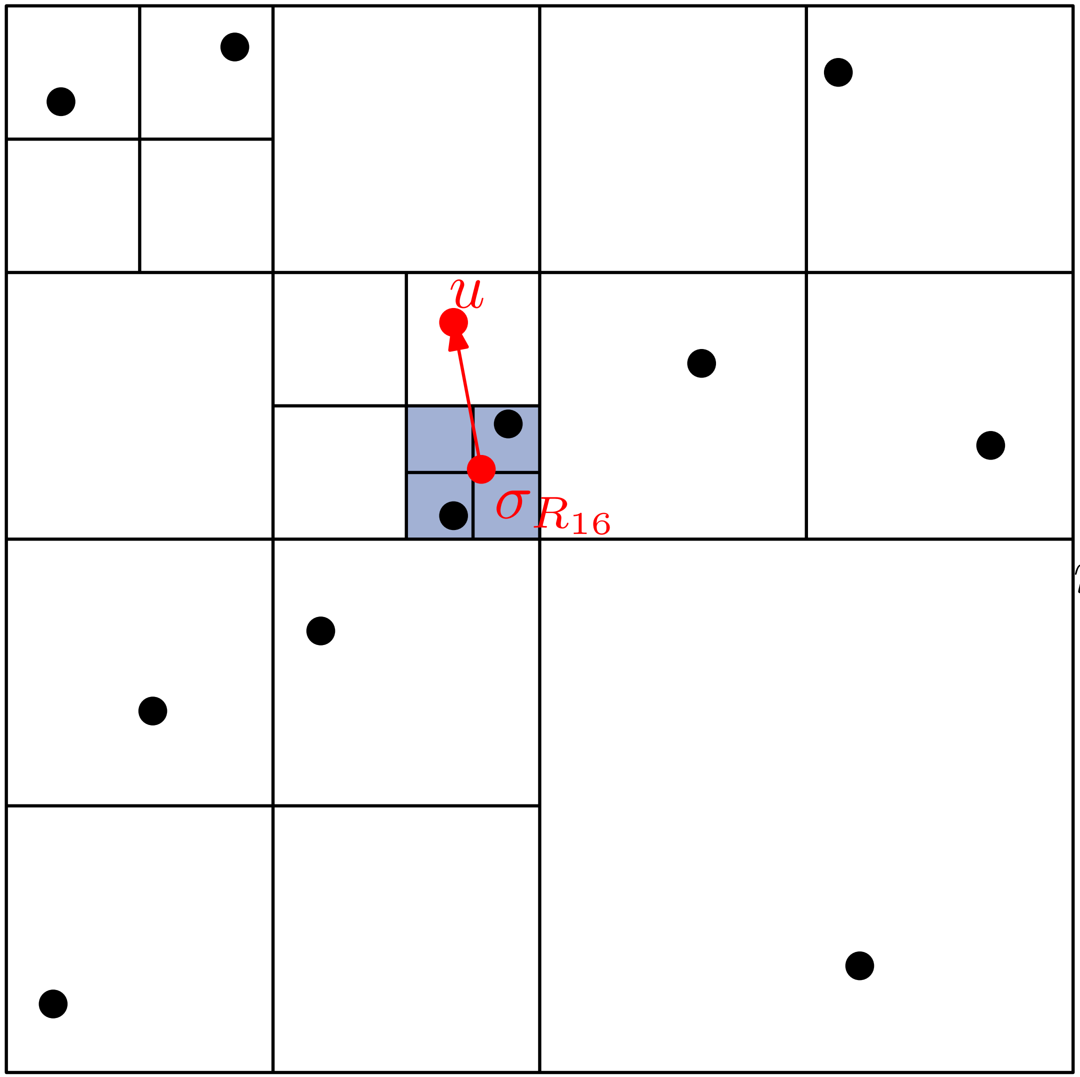


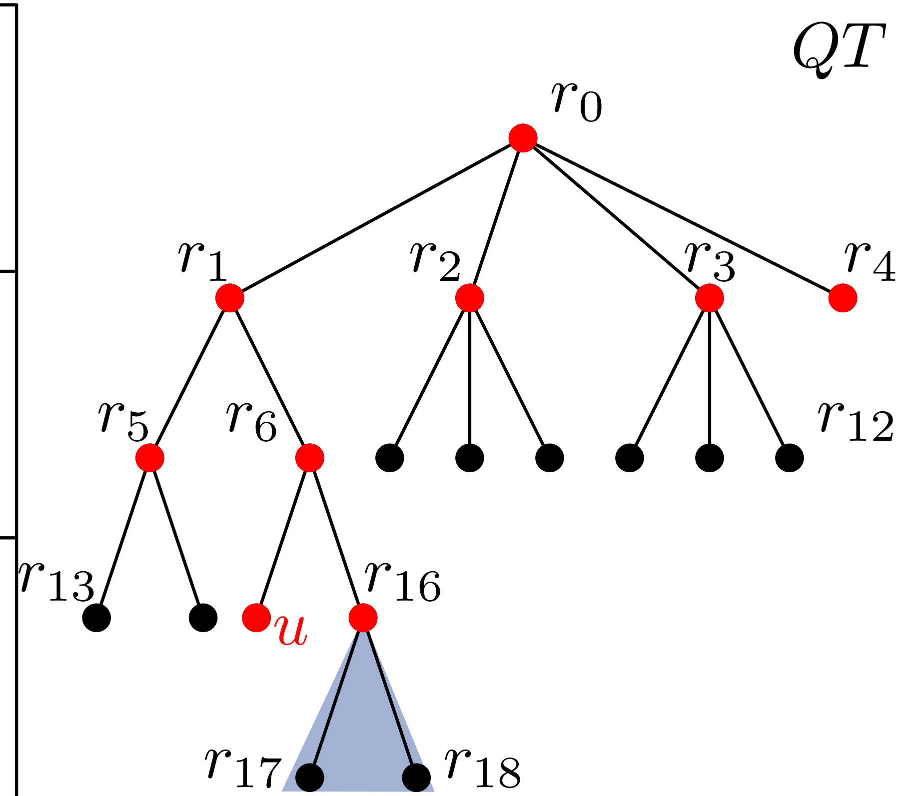


$f_{rep}(R_i, u) := |R_i| \cdot f_{rep}(p_{R_i}, u)$, where p_{R_i} - the barycentre of the points in R_i

Compute repulsive force between u and the barycentre of the vertices in the corresponding square

Forces with Quad-Trees (Barnes, Hut, 1986)



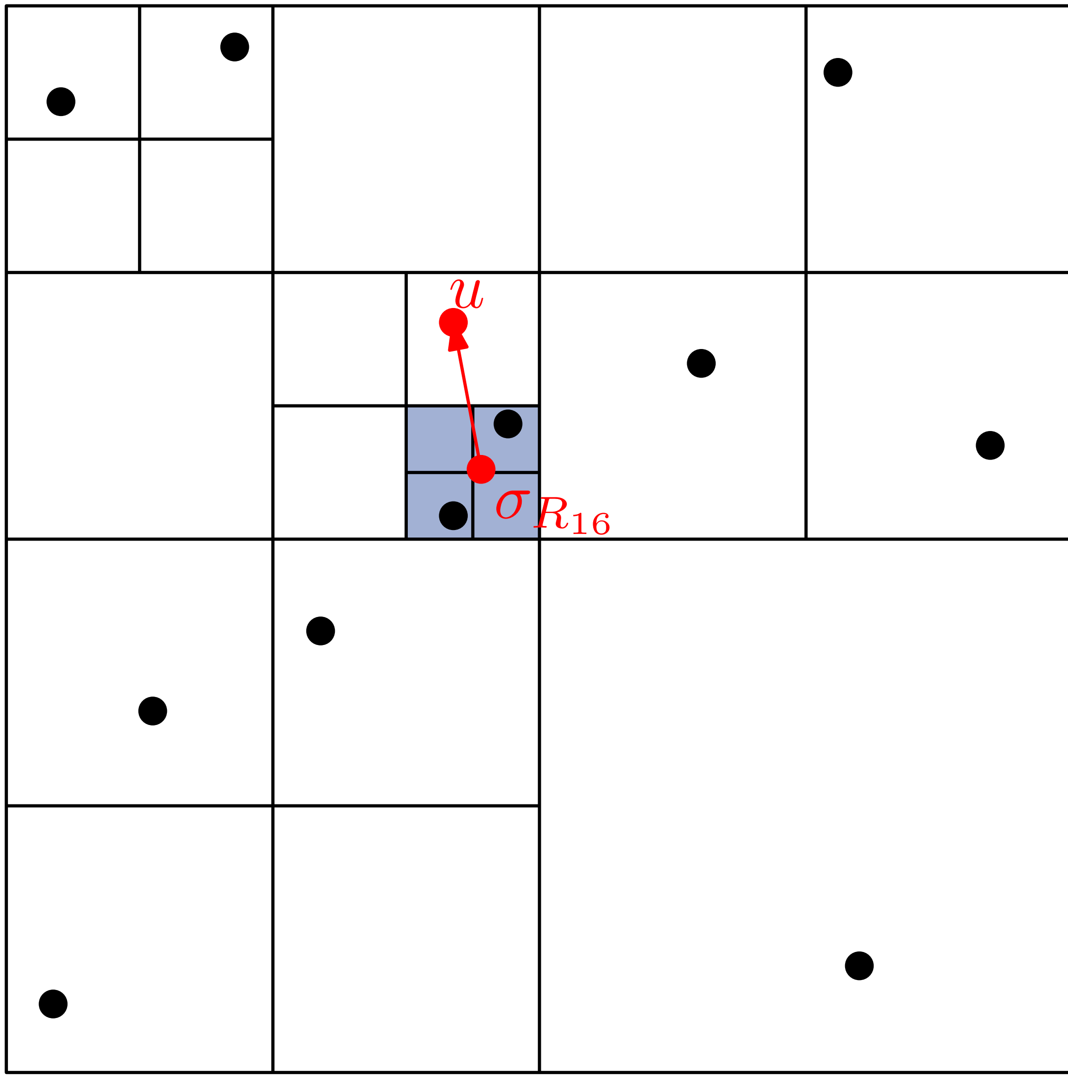


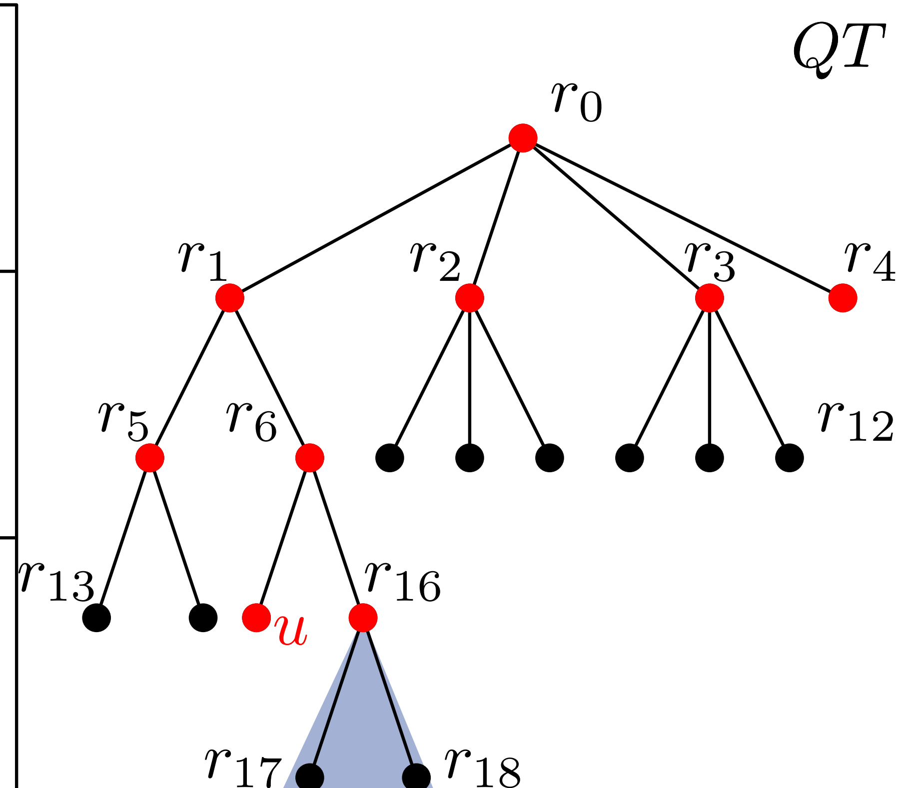
$f_{rep}(R_i, u) := |R_i| \cdot f_{rep}(p_{R_i}, u)$
, where p_{R_i} - the barycentre of the points in R_i

Compute repulsive force between u and the barycentre of the vertices in the corresponding square

Approximation control: if $\frac{\ell(R_i)}{dist(R_i,u)} > \gamma \rightarrow$ recurse in R_i

Forces with Quad-Trees (Barnes, Hut, 1986)





$f_{rep}(R_i, u) := |R_i| \cdot f_{rep}(p_{R_i}, u)$, where p_{R_i} - the barycentre of the points in R_i

- If $\frac{\ell(R_i)}{dist(R_i,u)} < \gamma$ always hold then we do $O(h)$ computations for u
- Computation time per iteration is $O(nh)$

Multilevel-Algorithms for Large Graphs

Motivation

- classical Spring-Embedder for large graphs are too slow
- scensitivity to random initialization of node positions

Multilevel-Algorithms for Large Graphs

Motivation

- classical Spring-Embedder for large graphs are too slow
- scensitivity to random initialization of node positions

GRIP – Graph dRawing with Intelligent Placement

(Gajer, Kobourov, 2004)

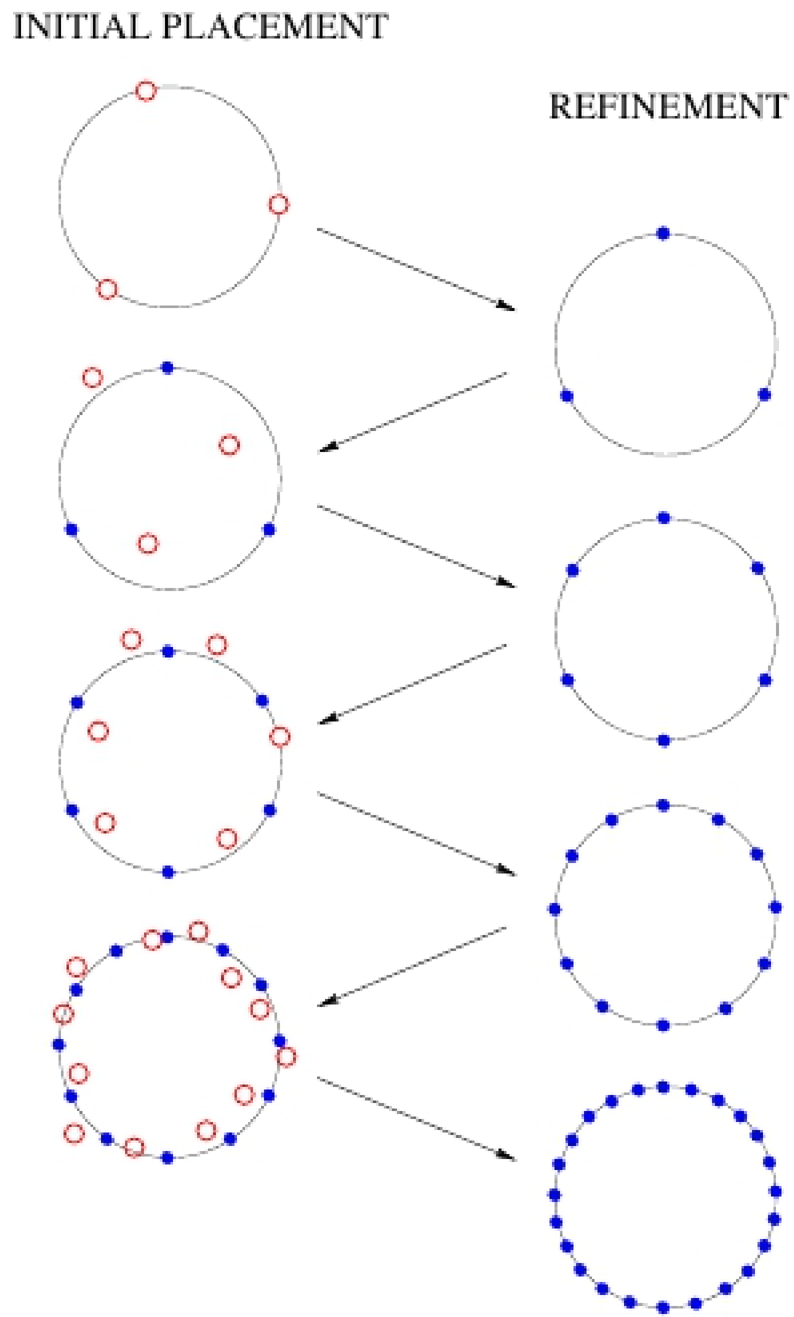
Approach

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

GRIP Algorithm

Input: Graph $G = (V, E)$

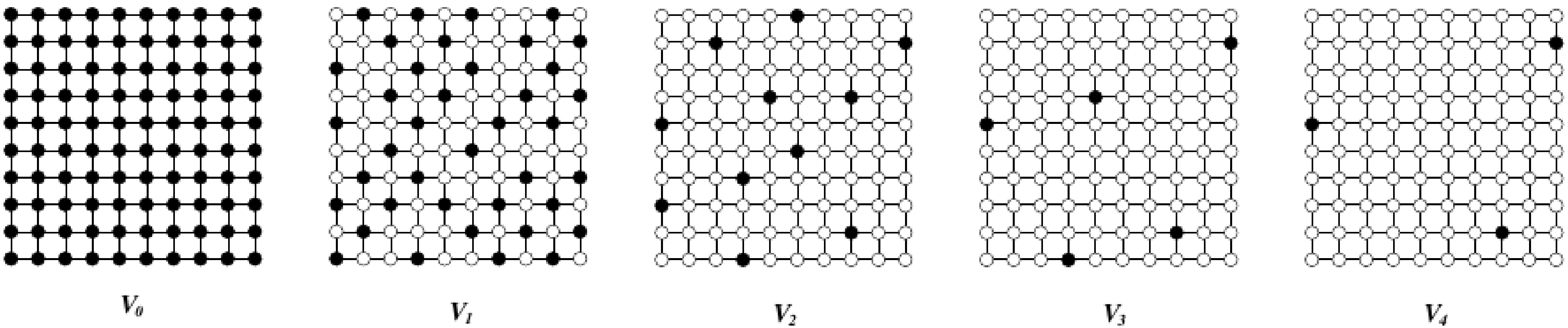
$\mathcal{V} \leftarrow$ Filtering $V = V_0 \supset V_1 \supset \cdots \supset V_k$
for $i = k$ **to** 0 **do**
  foreach $v \in V_i \setminus V_{i+1}$ **do**
    compute neighbours of v
    compute initial position of v
  for $j = 1$ **to** `rounds` **do**
    foreach $v \in V_i$ **do**
      force-based relaxation



MIS-filtering

Maximal Independent Set (MIS) filtering

- sequence of node sets $V = V_0 \supset V_1 \supset \cdots \supset V_k \supset \emptyset$
- V_i is (inclusion-) maximal set of nodes, so that
- distance in G between nodes in V_i is $\geq 2^{i-1} + 1$
- good balance betwen size of a level and depth of decompositon



Computing MIS-Filtering

Algorithm

- incremental procedure: given V_i compute V_{i+1}
- select random element v in V_i
- remove all elements from V_i with distance $\leq 2^i$ to v
- use BFS from v with search depth 2^i
- until no further node remains in V_i

Computing MIS-Filtering

Algorithm

- incremental procedure: given V_i compute V_{i+1}
- select random element v in V_i
- remove all elements from V_i with distance $\leq 2^i$ to v
- use BFS from v with search depth 2^i
- until no further node remains in V_i

Depth of Filtration

- for the last level k it holds that $2^k > \text{diam}(G)$
- therefore the depth is $O(\log \text{diam}(G))$

Computing MIS-Filtering

Algorithm

- incremental procedure: given V_i compute V_{i+1}
- select random element v in V_i
- remove all elements from V_i with distance $\leq 2^i$ to v
- use BFS from v with search depth 2^i
- until no further node remains in V_i

Depth of Filtration

- for the last level k it holds that $2^k > \text{diam}(G)$
- therefore the depth is $O(\log \text{diam}(G))$

Alternative

- coarsening with contracting matchings: end vertices of each matching edge are contracted (Walshaw, JGAA 2003)

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 of V_i, where forces are computed locally only to a constant number of nearest neighbors in V_i

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}

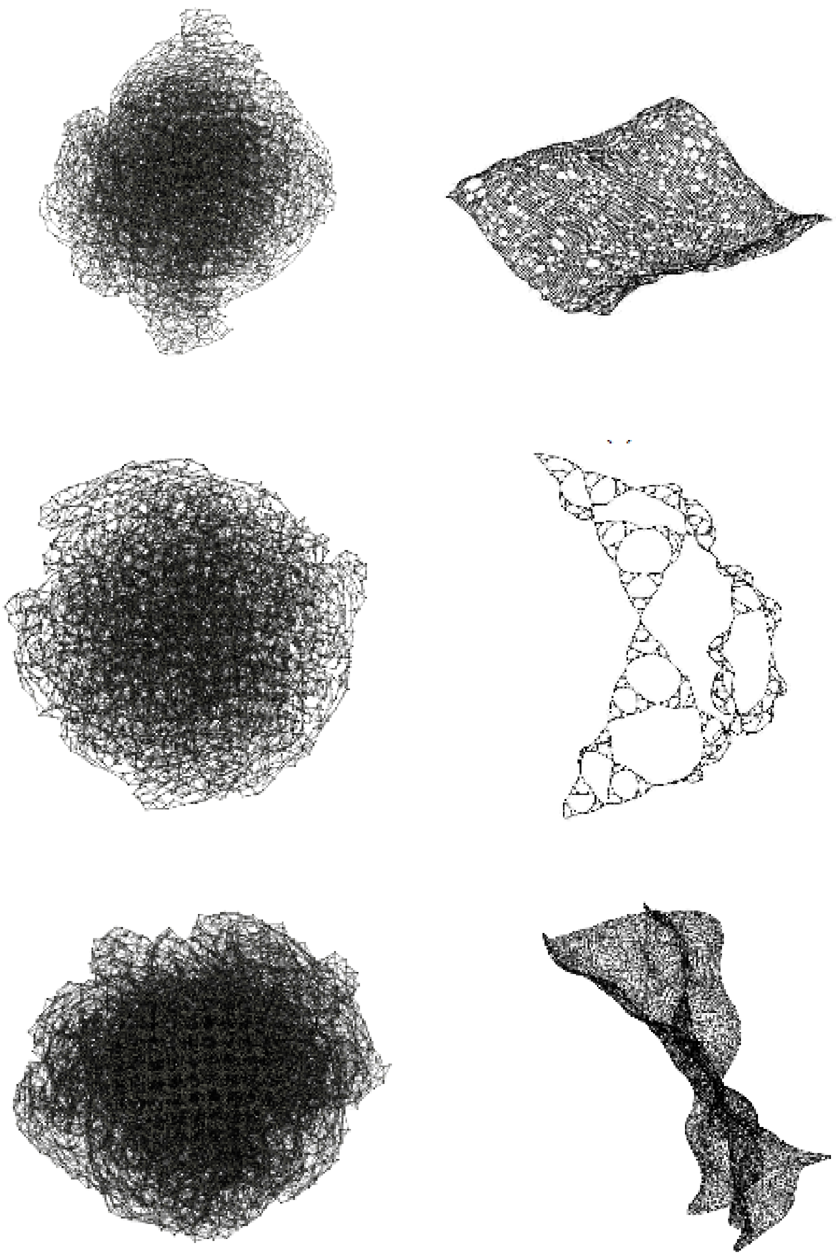
Step 2

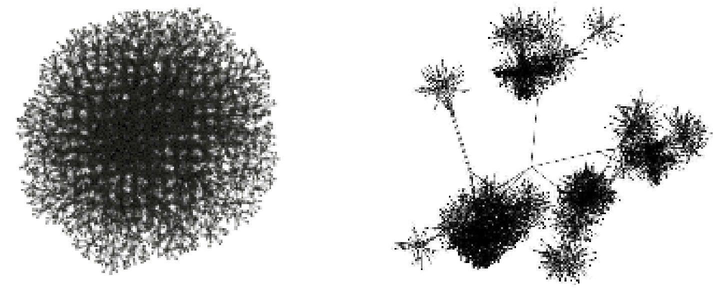
- perform force-based refinement of V_i, where forces are computed locally only to a constant number of nearest neighbors in V_i

Properties of GRIP

- intelligent step-by-step calculation starting from a good layout and using MIS-filtering
- significantly faster convergence
- graphs with $> 10{,}000$ nodes in seconds (2004)

Experimens (Hachul, Jünger 2007)



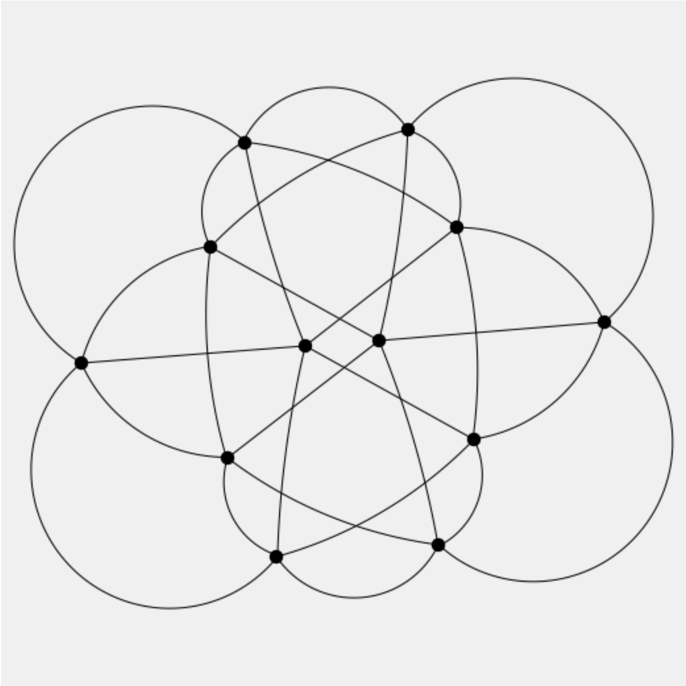


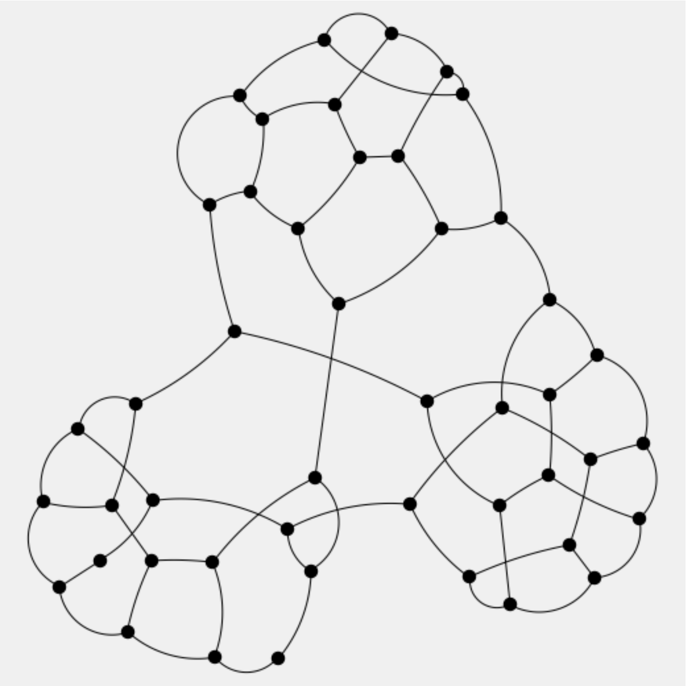
Left: Grid version of Fruchterman Reingold.
Right: GRIP

Other Examples

Lombardi-Spring-Embedder (Chernobelskiy et al. 2012)

- edges are circular arcs
- goal: optimal angular resolution $2\pi/\deg(v)$ at each node v
- additional rotational forces





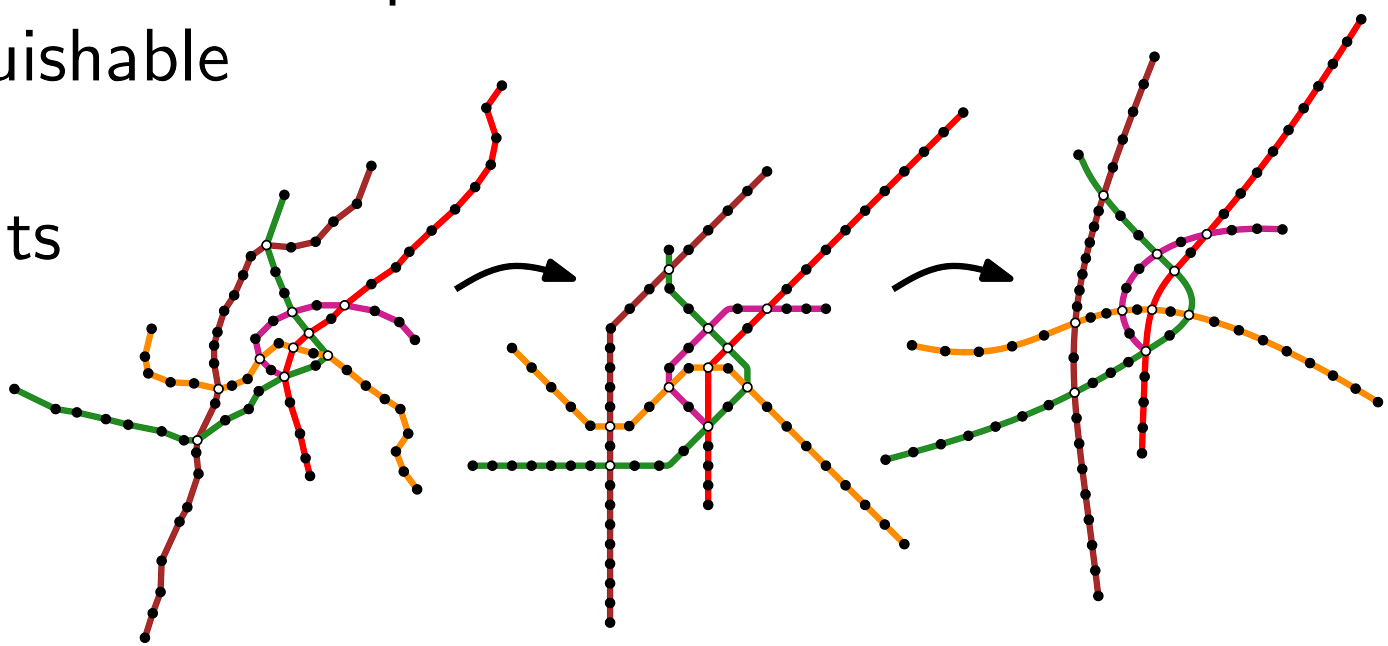
Other Examples

Lombardi-Spring-Embedder (Chernobelskiy et al. 2012)

- edges are circular arcs
- goal: optimal angular resolution $2\pi/\deg(v)$ at each node v
- additional rotational forces

Metro Maps with Bézier curves (Fink et al. 2013)

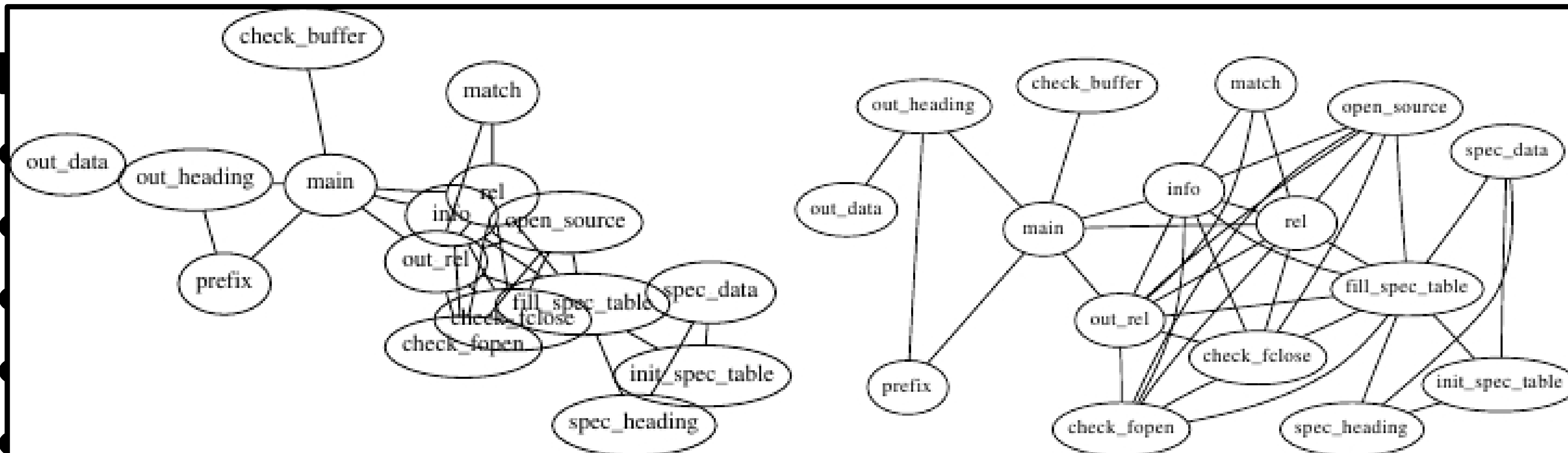
- model paths as Bézier curves
- forces on nodes and control points:
- lines are distinguishable
- few bend points
- few control points



Other Examples

Lombardi-Spring-Embedder (Chernobelskiy et al. 2012)

- edges are circular arcs
- goal: optimal angular resolution $2\pi/\deg(v)$ at each node v
- additional rotational forces



Realistic Node Sizes (Gansner, North 1998)

- node positions are adjusted to avoid overlaps

Summary

Force-based Approaches are

- easily understandable and implementable
- no special requirements on the input graph
- depending on the graphs (small and sparce) amazingly good layouts (Symmetries, Clustering, ...)
- easily adaptable and configurable
- robust
- scalable

Summary

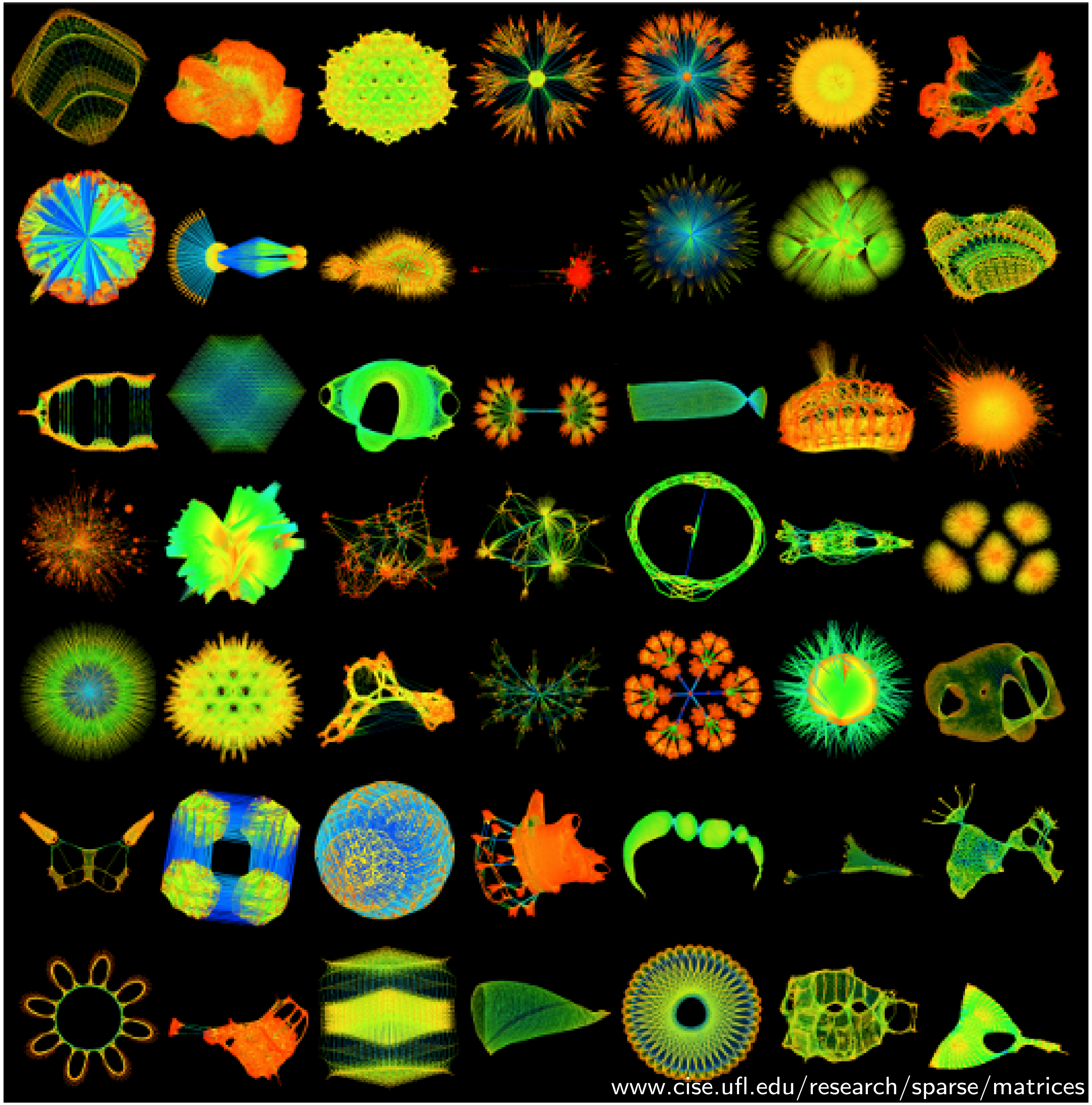
Force-based Approaches are

- easily understandable and implementable
- no special requirements on the input graph
- depending on the graphs (small and sparce) amazingly good layouts (Symmetries, Clustering, ...)
- easily adaptable and configurable
- robust
- scalable

But...

- usually no quality and running time guaranees
- bad choice of starting layout $\rightarrow$ slow convergence
- possibly slow for large graphs
- fine-turning can be done by experts
- no explicit optimization of aesthetic functions

Gallerie



©Davis & Hu