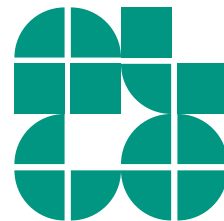


Algorithmen zur Visualisierung von Graphen

Kräftebasierte Verfahren

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

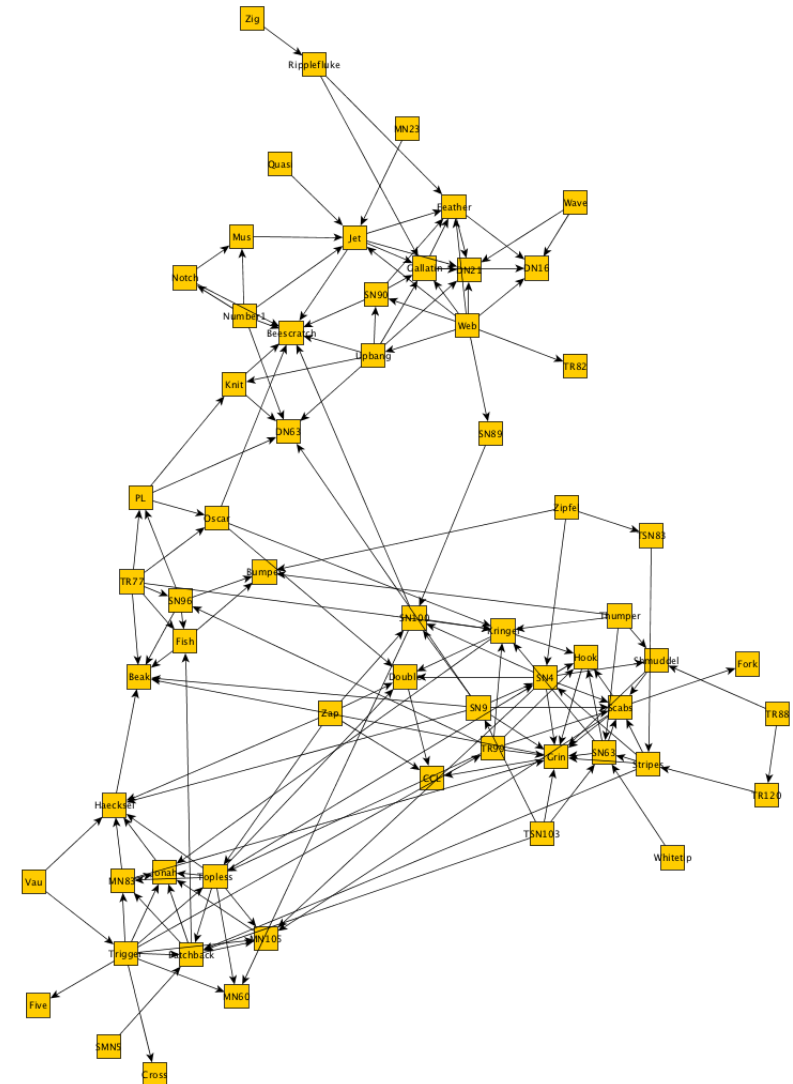
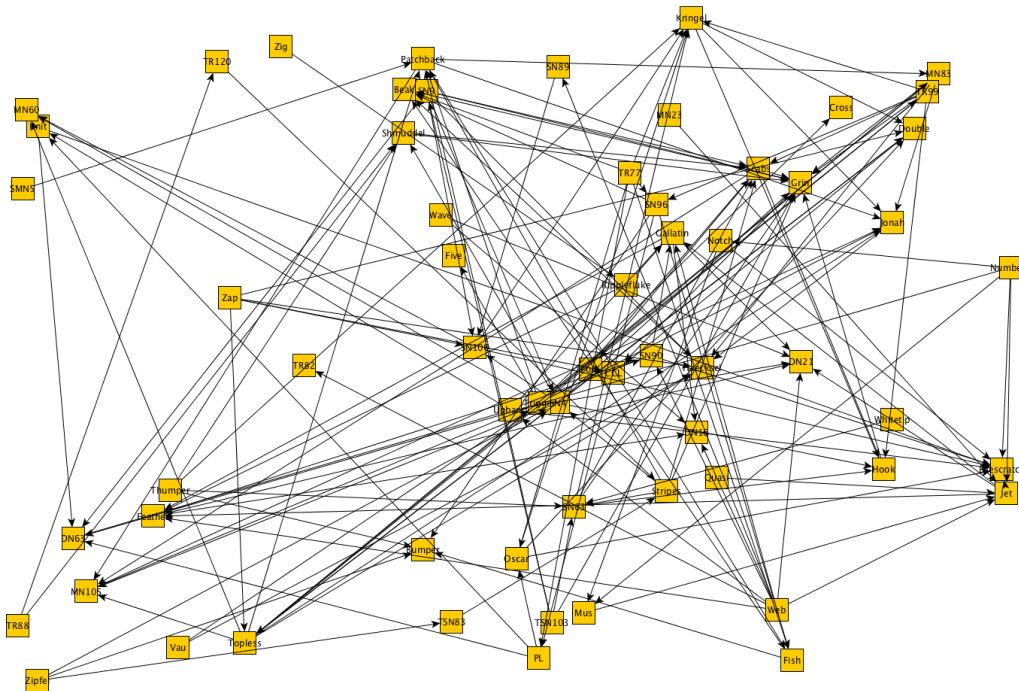
Tamara Mchedlidze · **Martin Nöllenburg**
19.01.2015



Generisches Layoutproblem

Geg.: Graph $G = (V, E)$

Ges.: übersichtliche lesbare Zeichnung von G



Generisches Layoutproblem

Geg.: Graph $G = (V, E)$

Ges.: übersichtliche lesbare Zeichnung von G

Kriterien:

- adjazente Knoten nahe
- nicht-adjazente Knoten weiter entfernt
- Kanten kurz, geradlinig, ähnlich lang
- möglichst wenige Kantenkreuzungen
- Knoten gleichmäßig verteilt
- dichte Teilgraphen (Cluster) in gemeinsamem Gebiet

Generisches Layoutproblem

Geg.: Graph $G = (V, E)$

Ges.: übersichtliche lesbare Zeichnung von G

Kriterien:

- adjazente Knoten nahe
- nicht-adjazente Knoten weiter entfernt
- Kanten kurz, geradlinig, ähnlich lang
- möglichst wenige Kantenkreuzungen
- Knoten gleichmäßig verteilt
- dichte Teilgraphen (Cluster) in gemeinsamem Gebiet



Optimierungskriterien widersprechen sich zum Teil

Beispiel: feste Kantenlängen

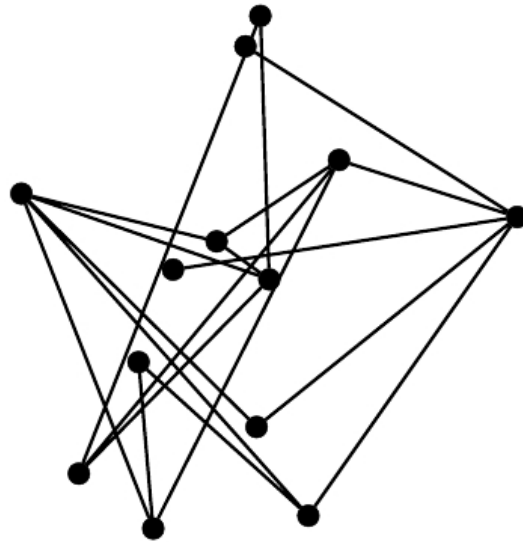
Geg.: Graph $G = (V, E)$, Soll-Längen $\ell(e)$ für alle $e \in E$

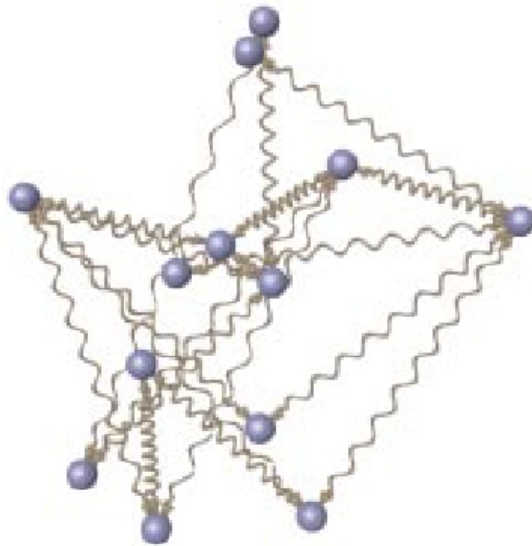
Ges.: Zeichnung von G , die die geg. Kantenlängen realisiert

NP-schwer für

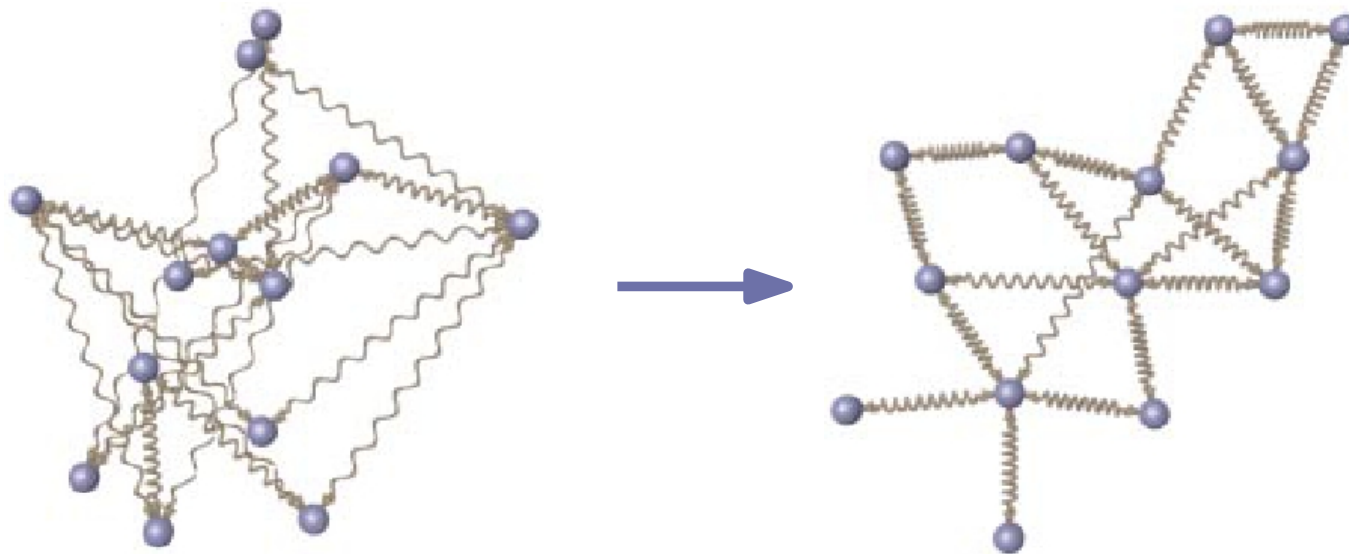
- Kantenlängen $\{1, 2\}$ [Saxe, '80]
- planare Zeichnung mit Einheitslängen [Eades, Wormald, '90]

Physikalisches Modell

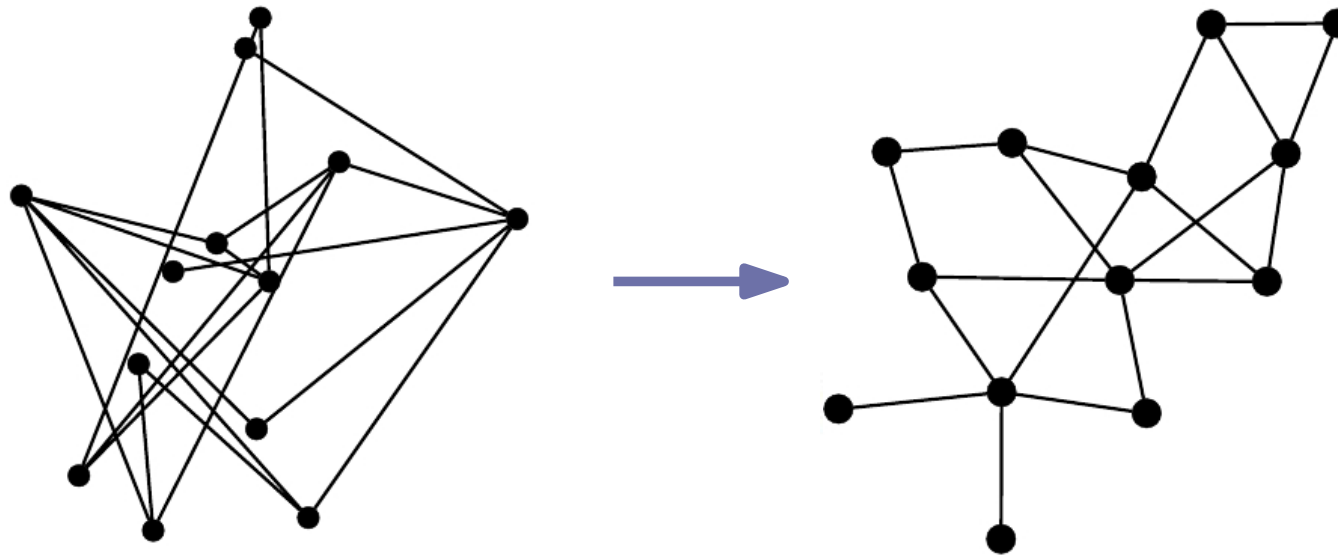




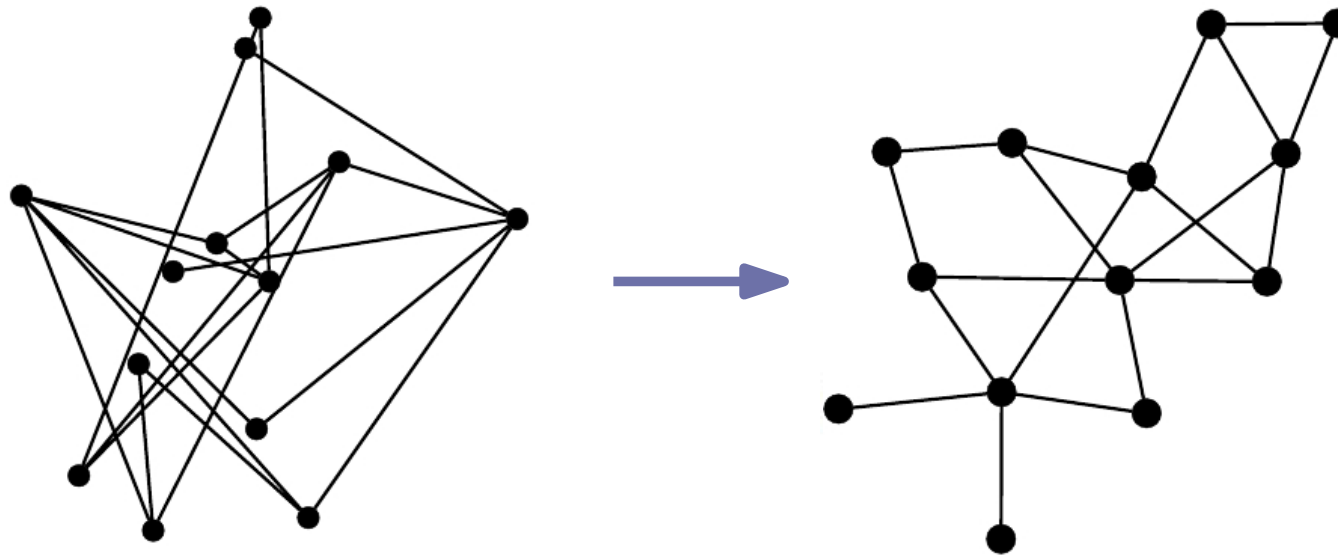
“To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system . . .



“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]



“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]



Sogenannte **spring-embedder** Algorithmen, die nach diesem oder ähnlichen Prinzipien arbeiten, gehören zu den häufigst verwendeten Graphenzeichenmethoden in der Praxis.

“To each node in the graph, a force is applied, and the nodes move to a minimal energy state. [Lades, 04]”

$$\ell = \ell(e)$$

Ideallänge der Feder für Kante e

$$p_v = (x_v, y_v)$$

Position von Knoten v

$$\|p_u - p_v\|$$

Euklidischer Abstand zwischen u und v

$$\overrightarrow{p_u p_v}$$

Einheitsvektor von u nach v

Modell:

- abstoßende Kraft zw. nicht adjazenten Knoten u und v

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

Modell:

- abstoßende Kraft zw. nicht adjazenten Knoten u und v

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

- anziehende Kraft zwischen adjazenten Knoten u und v

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

Modell:

- abstoßende Kraft zw. nicht adjazenten Knoten u und v

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

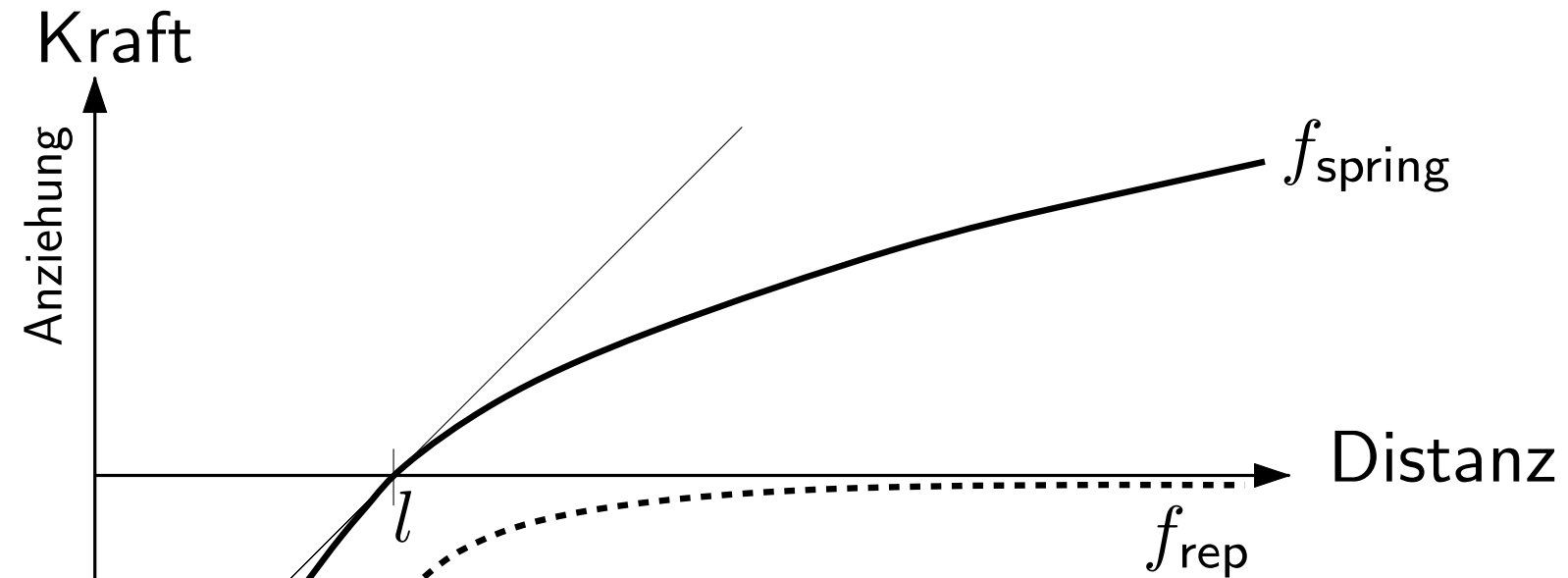
- anziehende Kraft zwischen adjazenten Knoten u und v

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

- resultierender Verschiebungsvektor für Knoten v

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

Kräfte diagramm Spring-Embedder (Eades, 1984)



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

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

Input: $G = (V, E)$ zusammenhängender ungerichteter Graph mit Anfangslayout $p = (p_v)_{v \in V}$,
Iterationszahl $K \in \mathbb{N}$, Schwellwert $\varepsilon > 0$

Output: Layout p mit „niedriger innerer Anspannung“

$t \leftarrow 1$

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

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

$$\left[\begin{array}{l} F_v(t) \leftarrow \sum_{u:\{u,v\} \notin E} f_{\text{rep}}(p_u, p_v) + \\ \sum_{u:\{u,v\} \in E} f_{\text{spring}}(p_u, p_v) \end{array} \right.$$

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

$$\left[p_v \leftarrow p_v + \delta \cdot F_v(t) \right.$$

$t \leftarrow t + 1$

Input: $G = (V, E)$ zusammenhängender ungerichteter Graph mit Anfangslayout $p = (p_v)_{v \in V}$,
Iterationszahl $K \in \mathbb{N}$, Schwellwert $\varepsilon > 0$

Output: Layout p mit „niedriger innerer Anspannung“

$t \leftarrow 1$

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**

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

$t \leftarrow t + 1$

Demo

Algorithmus Spring-Embedder (Eades, 1984)

Input: $G = (V, E)$ zusammenhängender ungerichteter Graph mit Anfangslayout $p = (p_v)_{v \in V}$,
Iterationszahl $K \in \mathbb{N}$,

Output: Layout p mit „niedriger Energie“

$t \leftarrow 1$

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

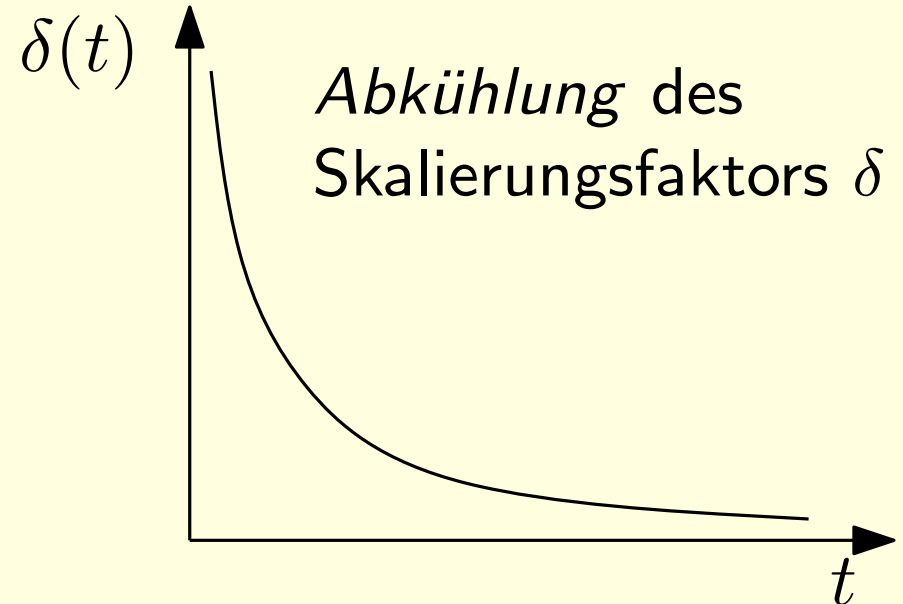
foreach $v \in V$ **do**

$$F_v(t) \leftarrow \sum_{u: \{u,v\} \notin E} \frac{1}{\|p_u - p_v\|} + \sum_{u: \{u,v\} \in E} \frac{1}{\|p_u - p_v\|^2}$$

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

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

$t \leftarrow t + 1$



Vorteile

- sehr einfacher Algorithmus
- gute Ergebnisse für kleine und mittel-große Graphen
- empirisch gute Wiedergabe von Symmetrien und Struktur

Vorteile

- sehr einfacher Algorithmus
- gute Ergebnisse für kleine und mittel-große Graphen
- empirisch gute Wiedergabe von Symmetrien und Struktur

Nachteile

- System am Ende möglicherweise nicht stabil
- lokale Minima
- Zeitaufwand für f_{spring} in $\mathcal{O}(|E|)$ und für f_{rep} in $\mathcal{O}(|V|^2)$

Vorteile

- sehr einfacher Algorithmus
- gute Ergebnisse für kleine und mittel-große Graphen
- empirisch gute Wiedergabe von Symmetrien und Struktur

Nachteile

- System am Ende möglicherweise nicht stabil
- lokale Minima
- Zeitaufwand für f_{spring} in $\mathcal{O}(|E|)$ und für f_{rep} in $\mathcal{O}(|V|^2)$

Einfluss

- Original-Paper von Peter Eades 1270-mal zitiert
- Basis für viele spätere Varianten

Modell:

- abstoßende Kraft zwischen **allen** Knotenpaaren u und v

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

Modell:

- abstoßende Kraft zwischen **allen** Knotenpaaren u und v

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

- anziehende Kraft zwischen adjazenten Knoten u und v

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

Modell:

- abstoßende Kraft zwischen **allen** Knotenpaaren u und v

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

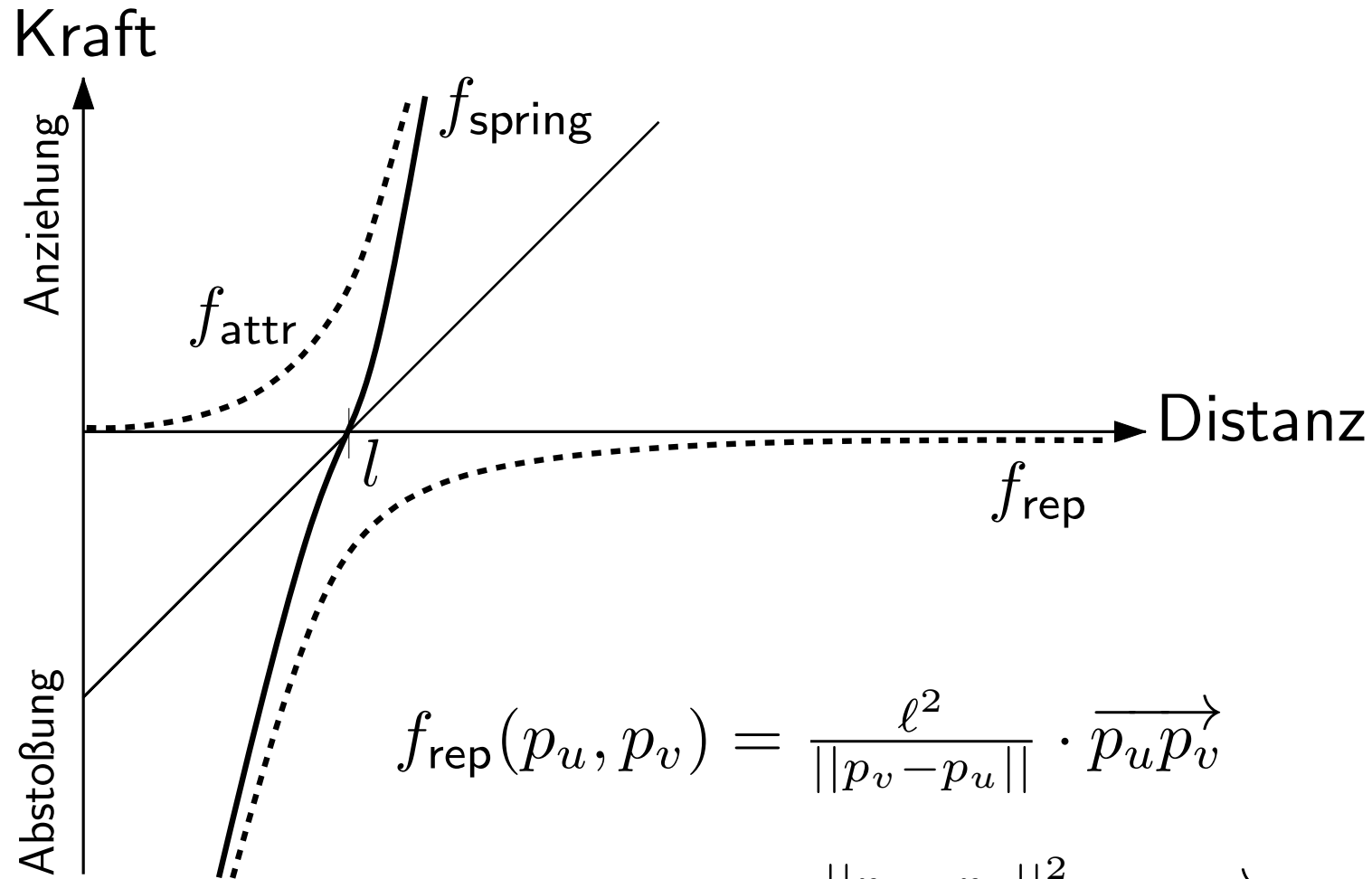
- anziehende Kraft zwischen adjazenten Knoten u und v

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

- resultierende Federkraft zw. adjazenten Knoten u und v

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

Kräfte diagramm Fruchtermann & Reingold

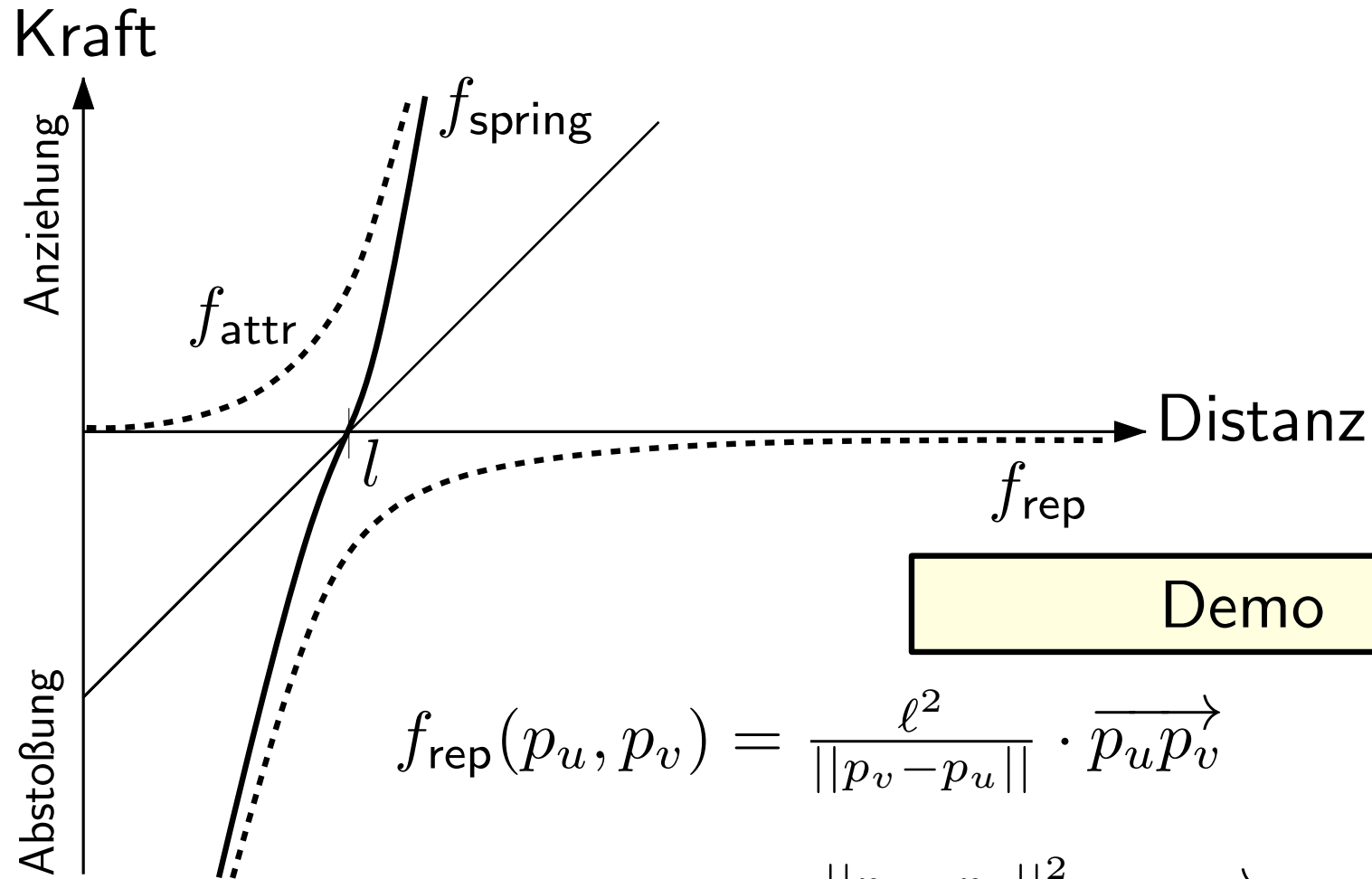


$$f_{rep}(p_u, p_v) = \frac{\ell^2}{\|p_v - p_u\|} \cdot \overrightarrow{p_u p_v}$$

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

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

Kräfte diagramm Fruchtermann & Reingold



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

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

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

Weitere mögliche Komponenten

- **Massenträgheit**
- **Gravitation**
- **magnetische Richtungskraft**

Ideen zur Modellierung?

Weitere mögliche Komponenten

- **Massenträgheit**

definiere Knotenmasse $\Phi(v) = 1 + \deg(v)/2$

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

- **Gravitation**

- **magnetische Richtungskraft**

Weitere mögliche Komponenten

- **Massenträgheit**

definiere Knotenmasse $\Phi(v) = 1 + \deg(v)/2$

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

- **Gravitation**

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

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

- **magnetische Richtungskraft**

Weitere mögliche Komponenten

■ Massenträgheit

definiere Knotenmasse $\Phi(v) = 1 + \deg(v)/2$

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

■ Gravitation

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

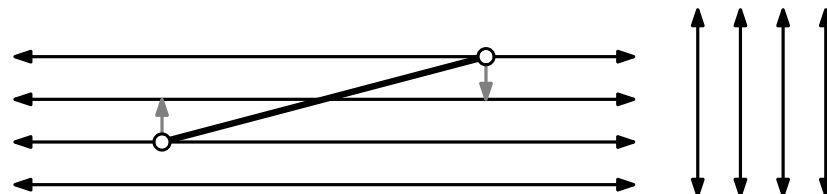
$f_{\text{grav}}(p_v) = c_{\text{grav}} \cdot \Phi(v) \cdot \overrightarrow{p_v p_{\text{bary}}}$

■ magnetische Richtungskraft

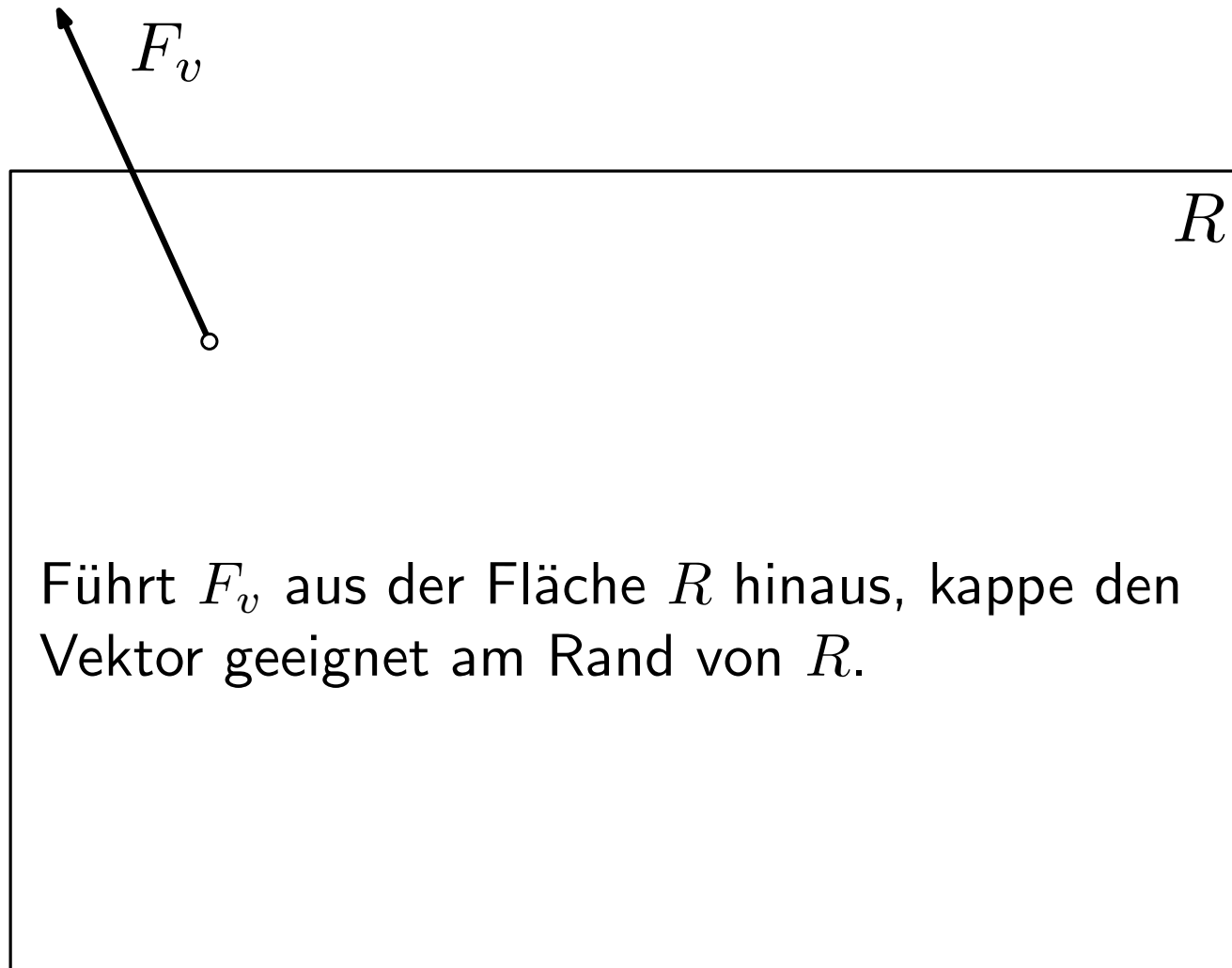
– definiere Magnetfeld (z.B. vertikal und horizontal)

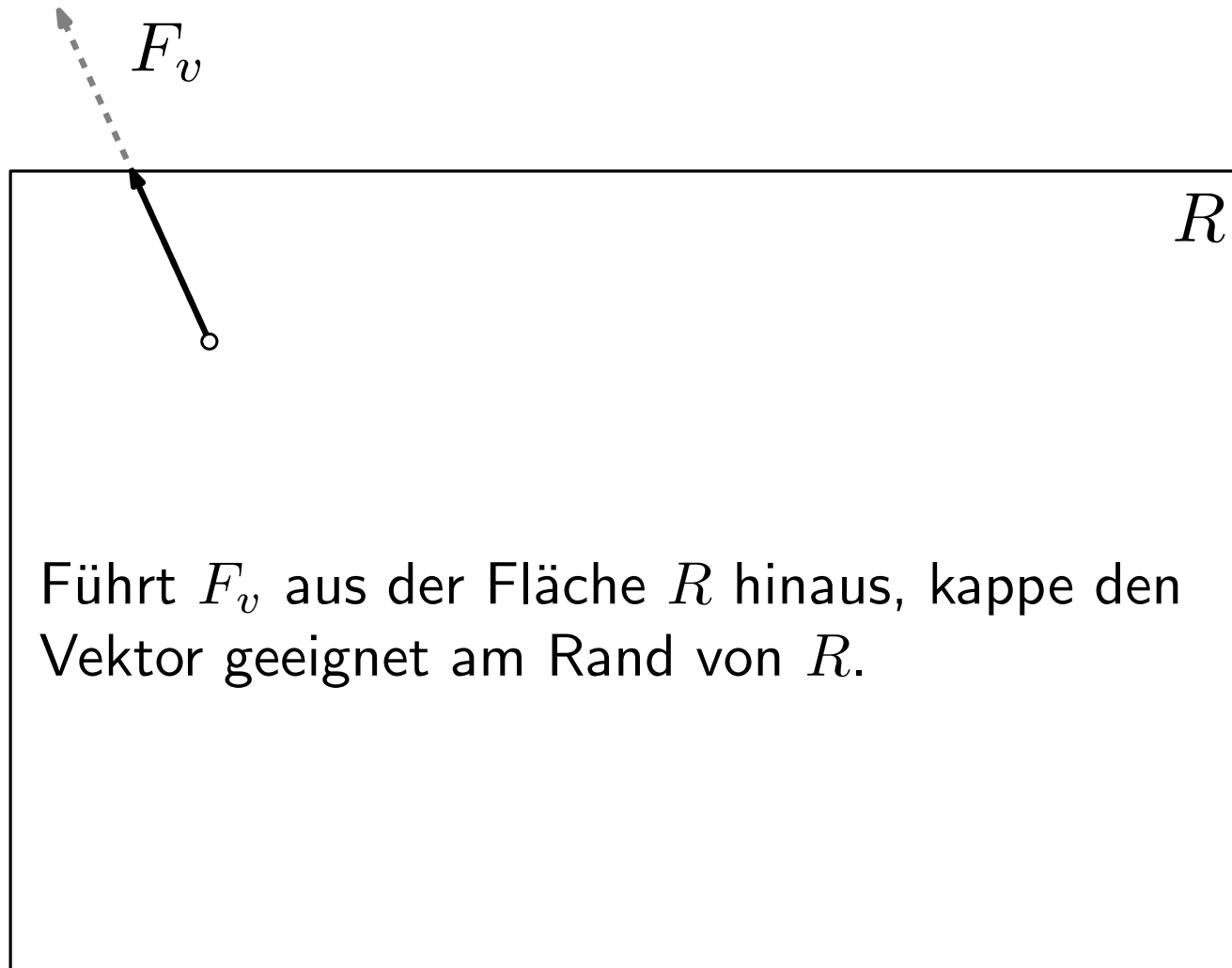
– projiziere Kanten auf nächste Richtung des Magnetfeldes

– definiere anziehende Kraft zu projizierten Knoten

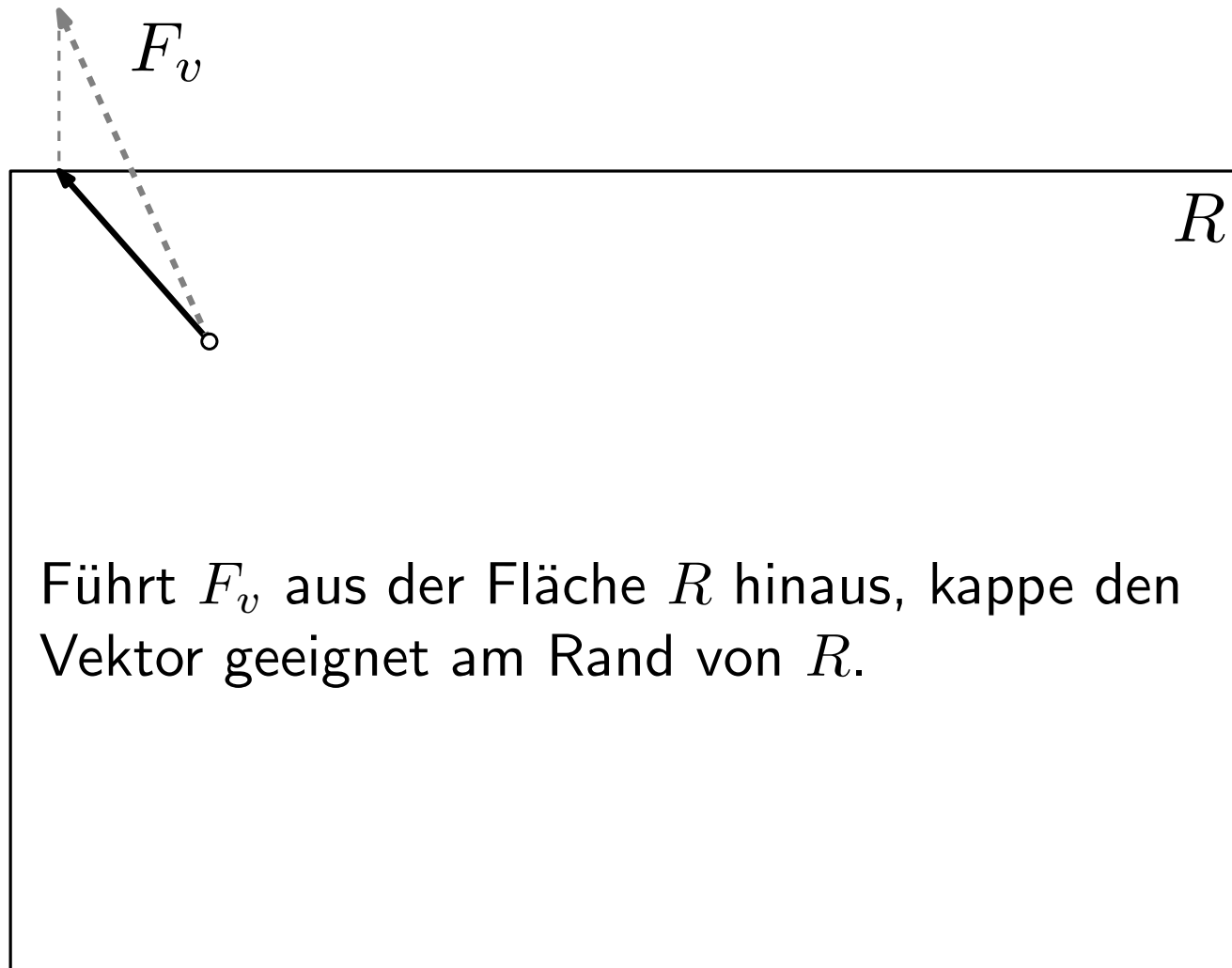


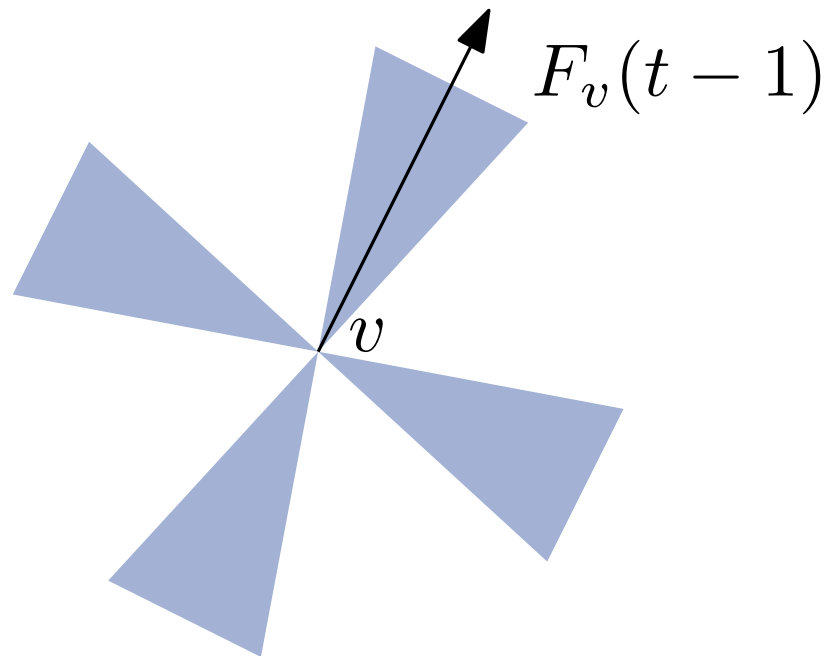
Beschränkte Zeichenfläche



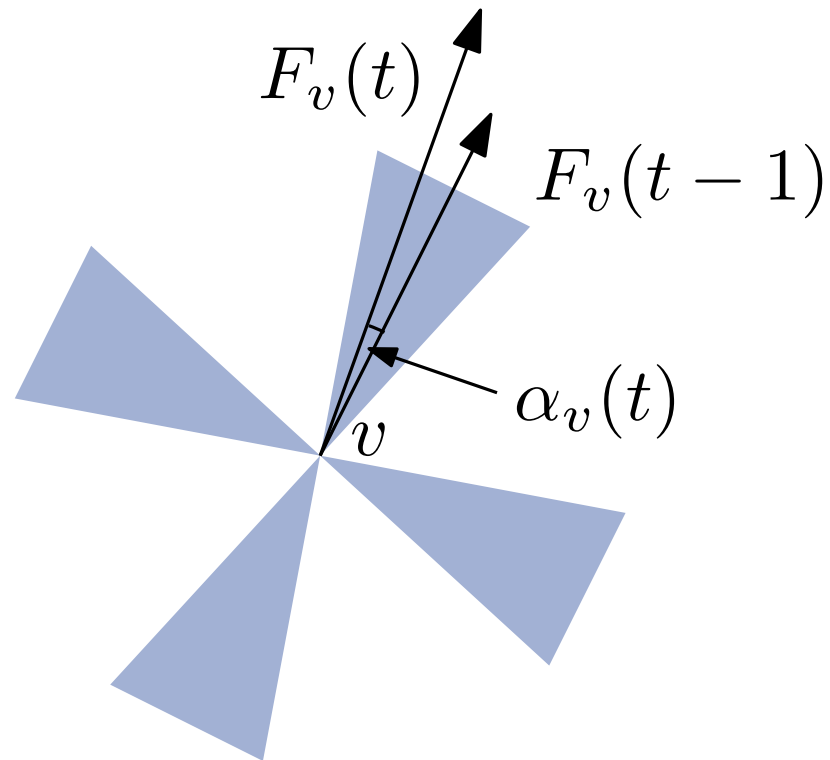


Beschränkte Zeichenfläche





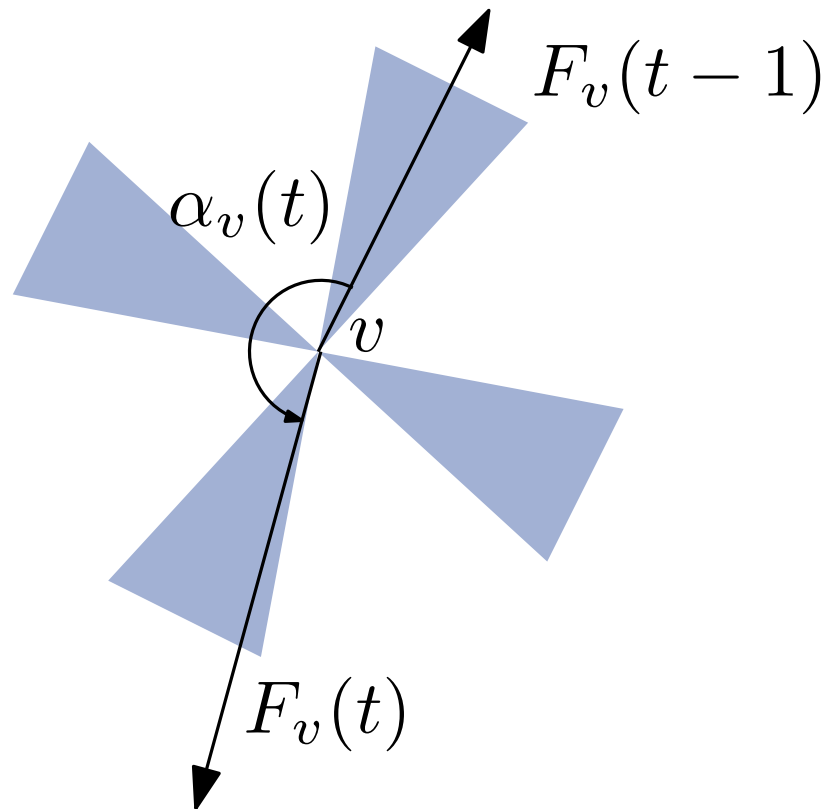
- speichere alten Verschiebevektor $F_v(t-1)$



- speichere alten Verschiebevektor $F_v(t-1)$

lokale Temperatur

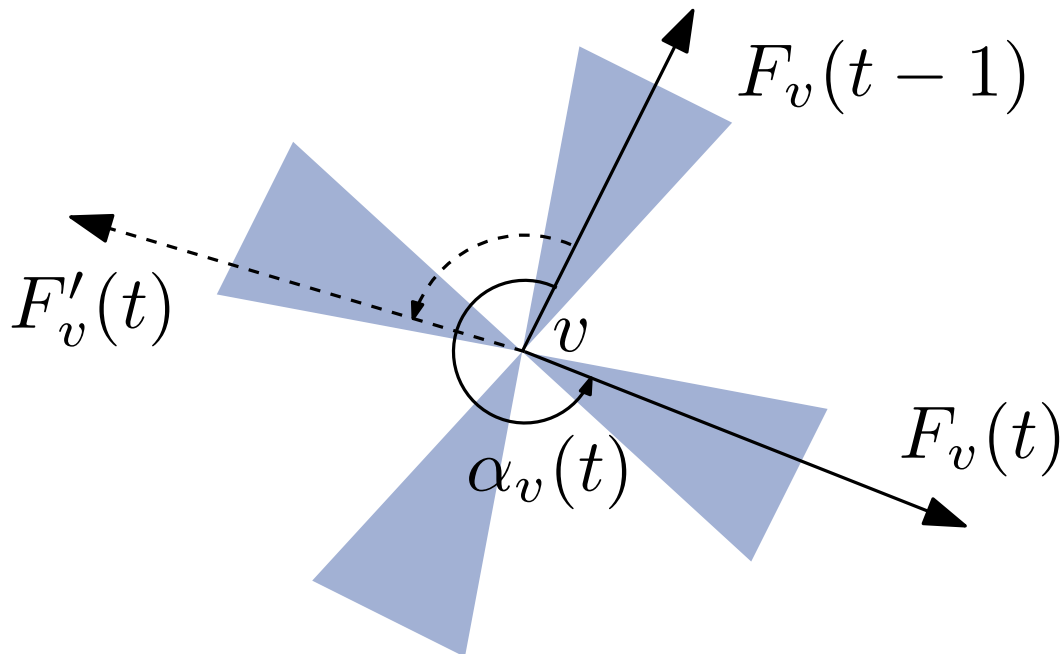
- $\cos(\alpha_v(t)) \approx 1$: gleiche Richtung
→ Temperatur erhöhen



- speichere alten Verschiebevektor $F_v(t-1)$

lokale Temperatur

- $\cos(\alpha_v(t)) \approx 1$:
gleiche Richtung
→ Temperatur erhöhen
- $\cos(\alpha_v(t)) \approx -1$:
Oszillation
→ Temperatur verringern



- speichere alten Verschiebevektor $F_v(t-1)$

lokale Temperatur

- $\cos(\alpha_v(t)) \approx 1$:
gleiche Richtung
→ Temperatur erhöhen
- $\cos(\alpha_v(t)) \approx -1$:
Oszillation
→ Temperatur verringern
- $\cos(\alpha_v(t)) \approx 0$:
Rotation
→ Rotationszähler updaten,
ggf. Temperatur verringern

Vorteile

- weiterhin sehr einfacher Algorithmus
- superlineare Kräfte \rightarrow schnellere Konvergenz
- weitere Modifikationen verbessern Layoutqualität und führen zu schnellerer Konvergenz

Vorteile

- weiterhin sehr einfacher Algorithmus
- superlineare Kräfte \rightarrow schnellere Konvergenz
- weitere Modifikationen verbessern Layoutqualität und führen zu schnellerer Konvergenz

Nachteile

- Stabilität weiterhin nicht garantiert
- lokale Minima möglich
- quadratischer Zeitaufwand für abstoßende Kräfte

Vorteile

- weiterhin sehr einfacher Algorithmus
- superlineare Kräfte \rightarrow schnellere Konvergenz
- weitere Modifikationen verbessern Layoutqualität und führen zu schnellerer Konvergenz

Nachteile

- Stabilität weiterhin nicht garantiert
- lokale Minima möglich
- quadratischer Zeitaufwand für abstoßende Kräfte

Einfluss

- Variante von Fruchterman und Reingold wohl populärste kräftebasierte Methode (2312-mal zitiert)

Vorteile

- weiterhin sehr einfacher Algorithmus
- superlineare Kräfte \rightarrow schnellere Konvergenz
- weitere Modifikationen verbessern Layoutqualität und führen zu schnellerer Konvergenz

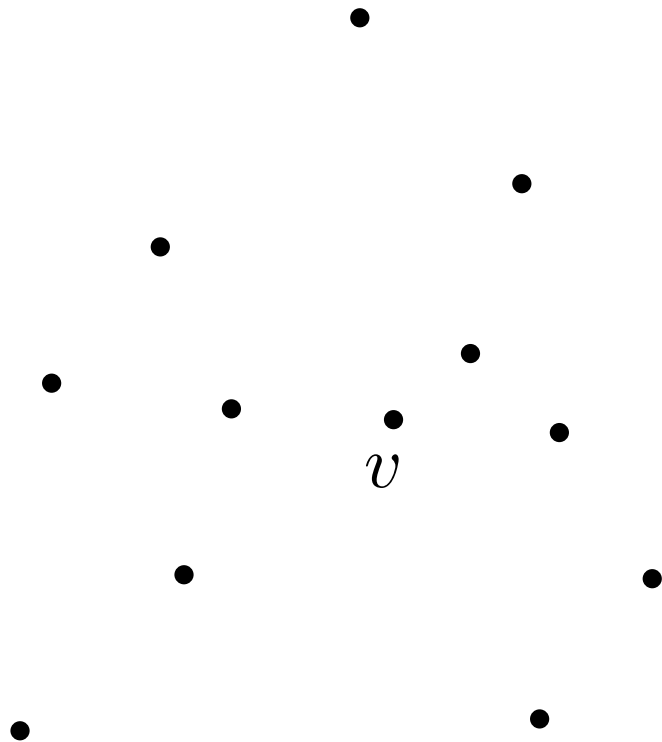
Nachteile

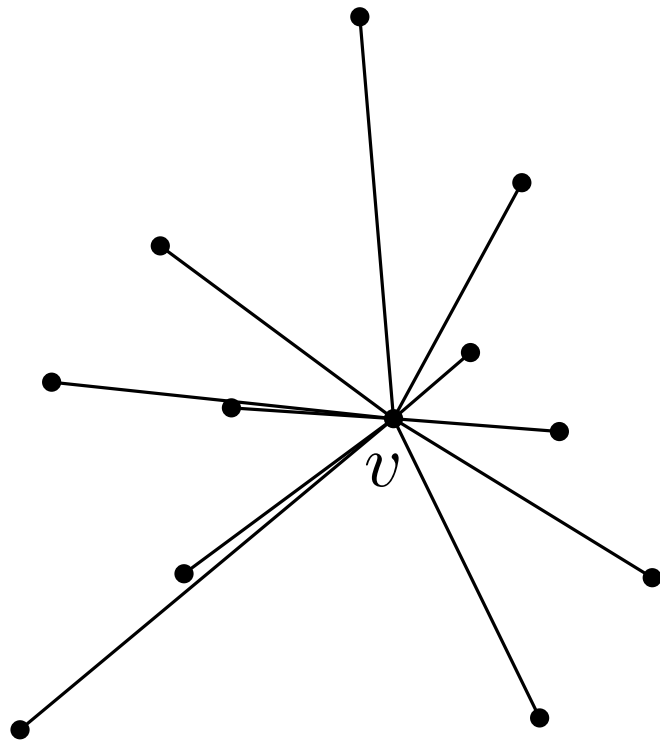
- Stabilität weiterhin nicht garantiert
- lokale Minima möglich
- quadratischer Zeitaufwand für abstoßende Kräfte

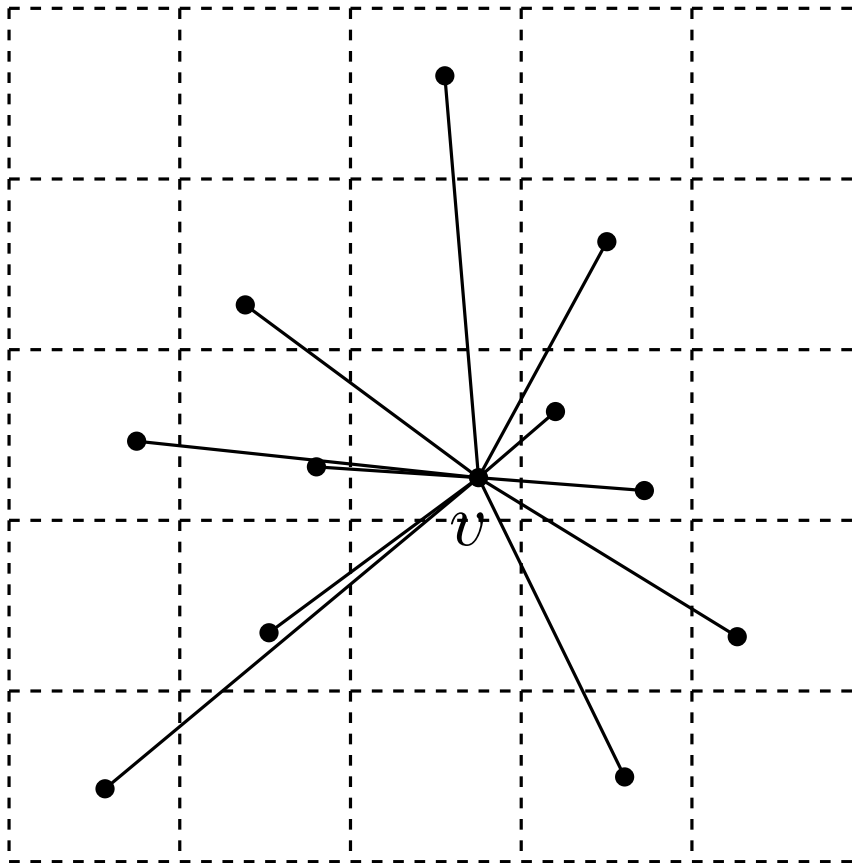
Wie könnte man das verringern?

Einfluss

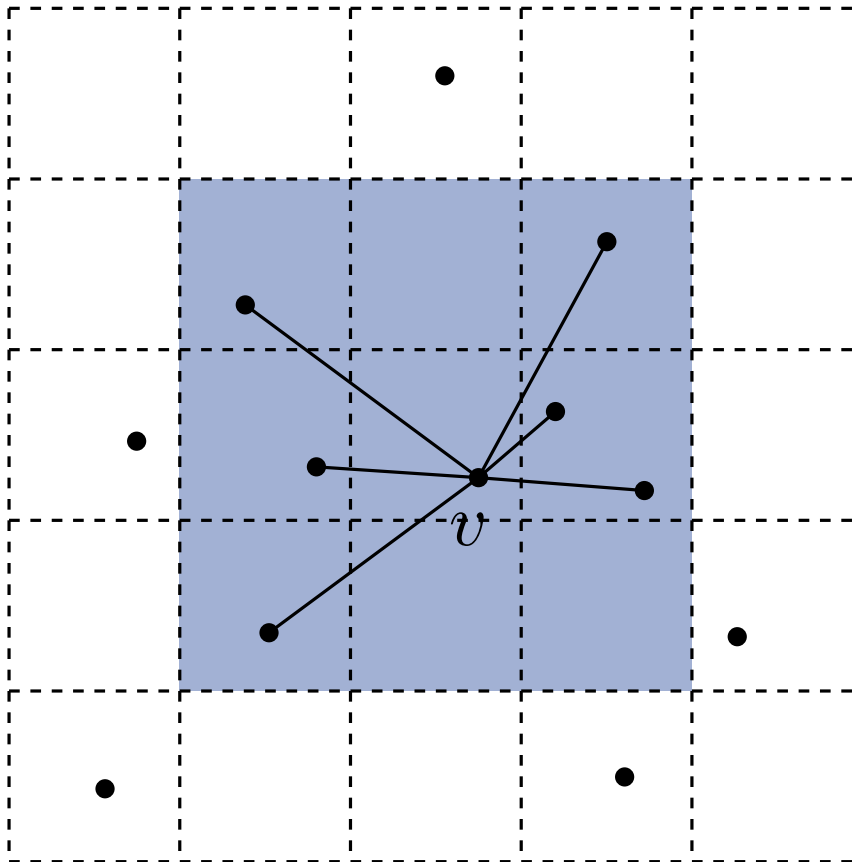
- Variante von Fruchterman und Reingold wohl populärste kräftebasierte Methode (2312-mal zitiert)



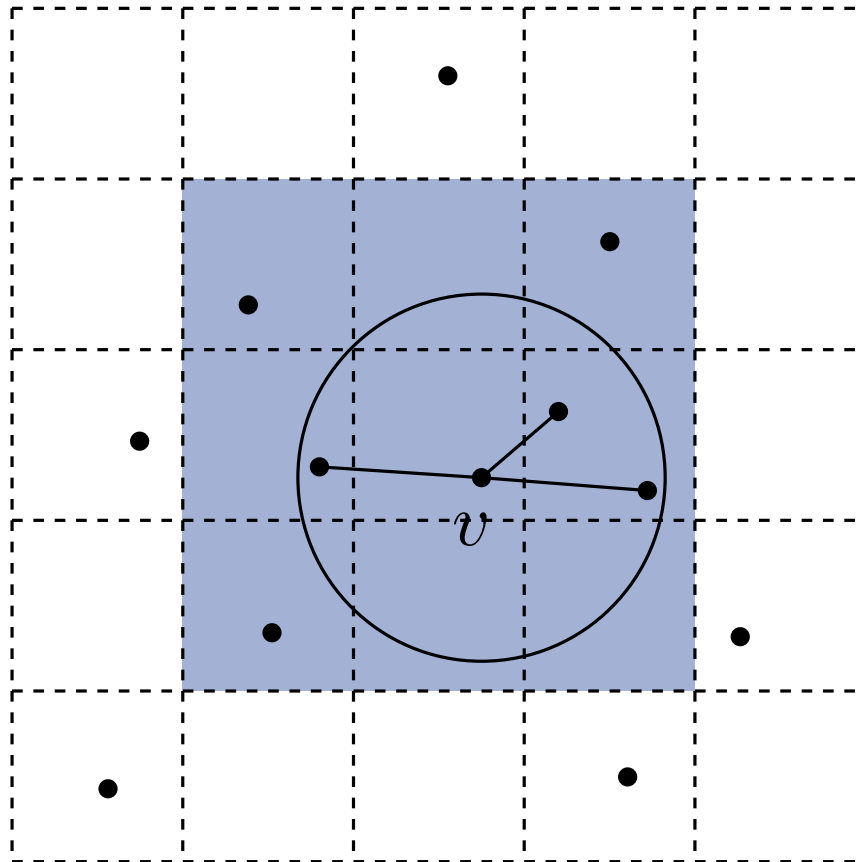




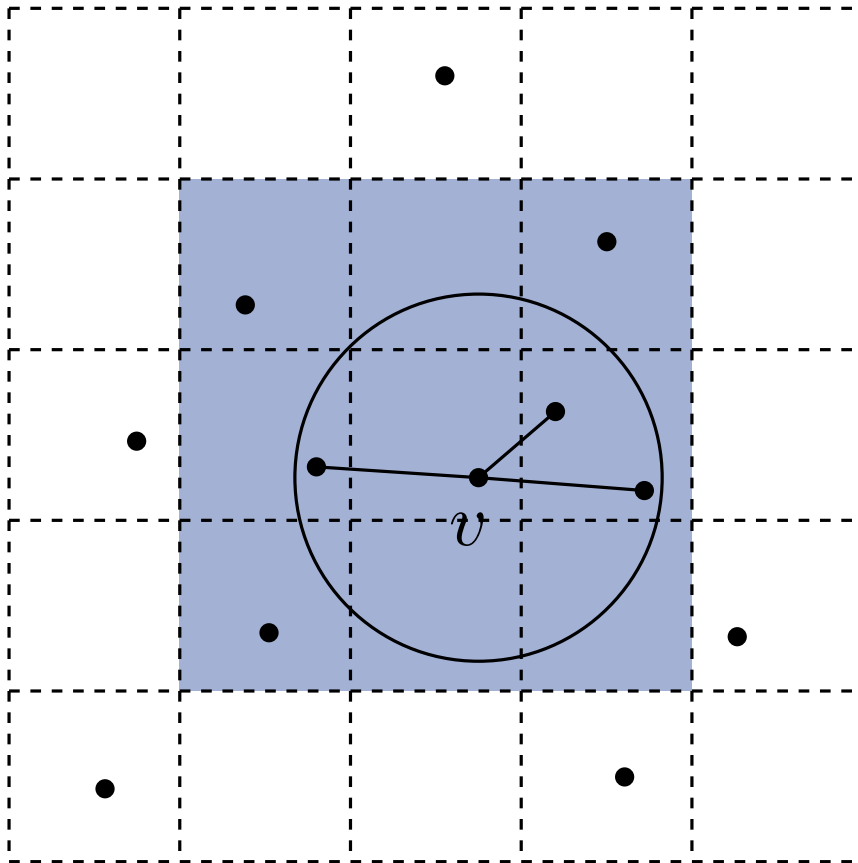
- zerlege Ebene in Gitterzellen



- zerlege Ebene in Gitterzellen
- betrachte abstoßende Kräfte nur zu Knoten in Nachbarzellen



- zerlege Ebene in Gitterzellen
- betrachte abstoßende Kräfte nur zu Knoten in Nachbarzellen
- und nur falls kleiner als Maximalabstand d_{\max}

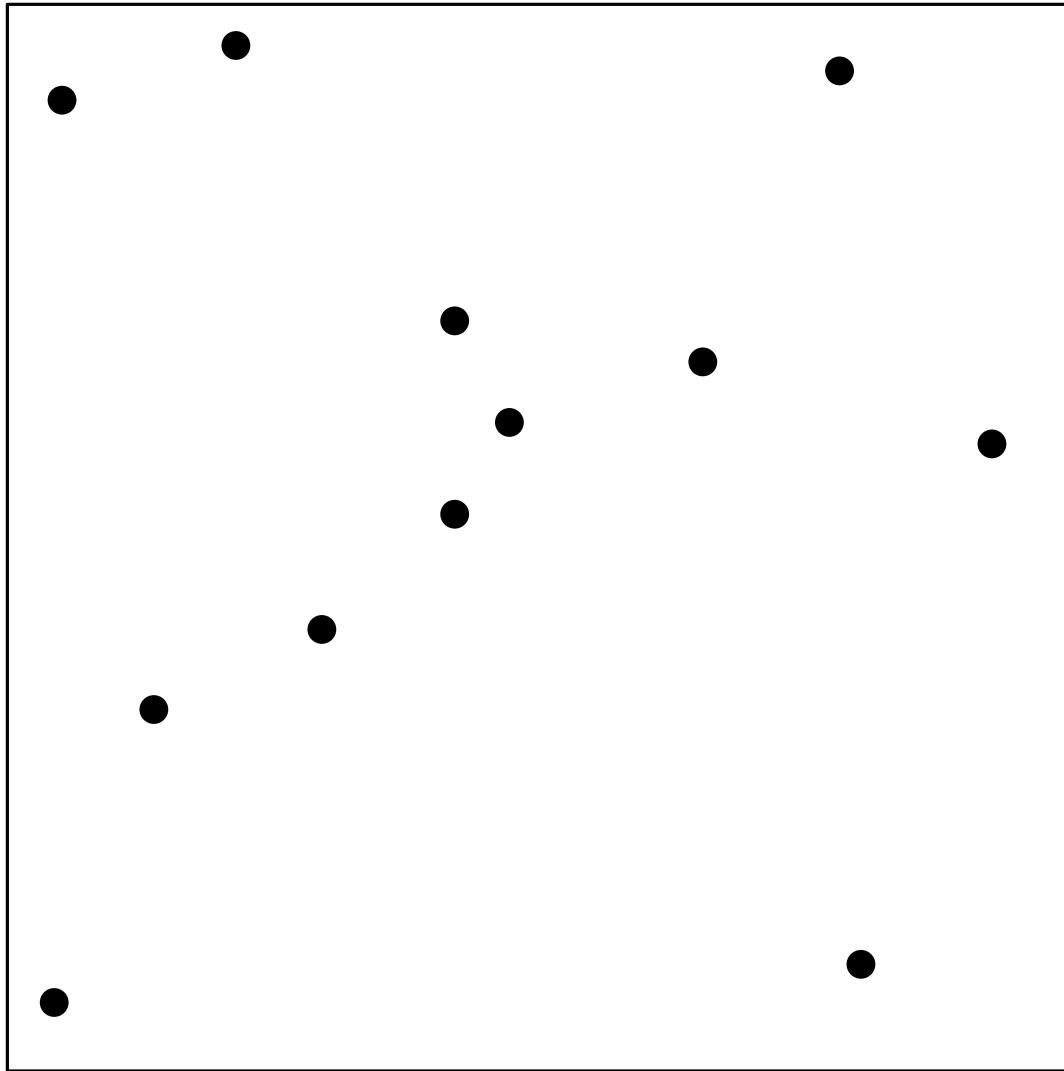


- zerlege Ebene in Gitterzellen
- betrachte abstoßende Kräfte nur zu Knoten in Nachbarzellen
- und nur falls kleiner als Maximalabstand d_{\max}

Diskussion

- sinnvolle Idee zur Laufzeitverbesserung
- worst-case kein Vorteil
- Qualitätsverlust (z.B. Oszillation um d_{\max})

Quad-Tree

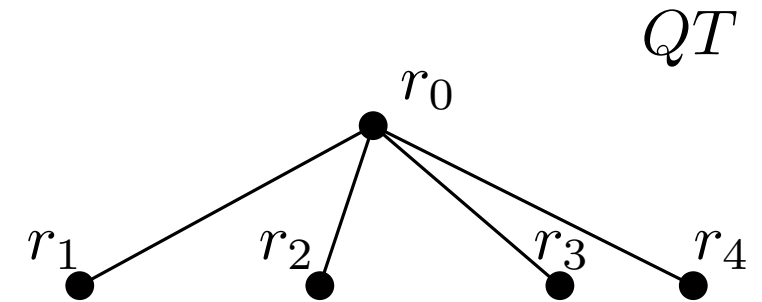
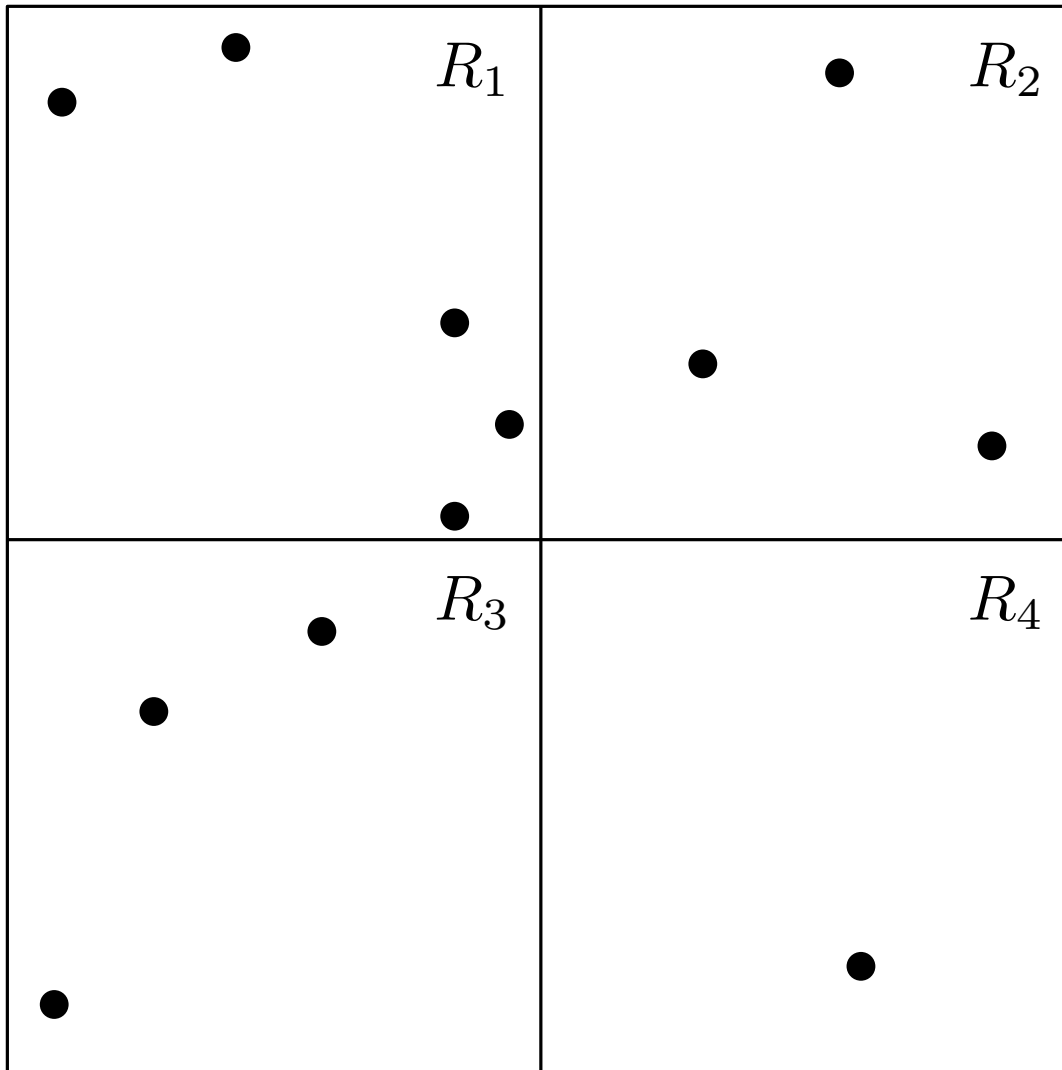


R_0

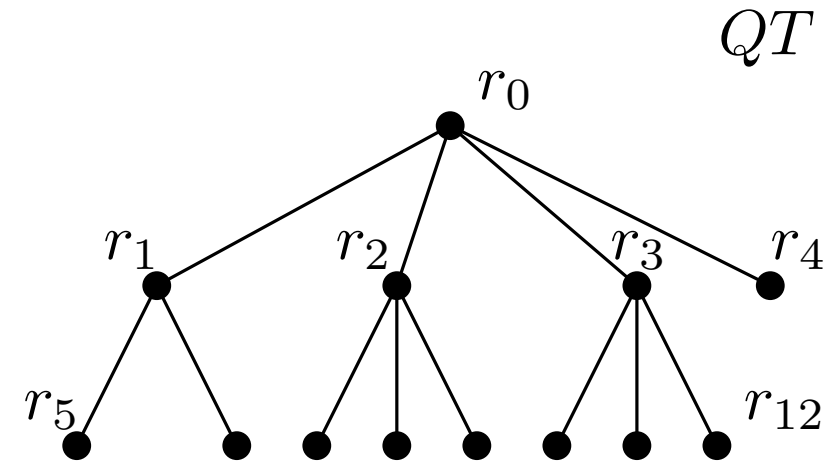
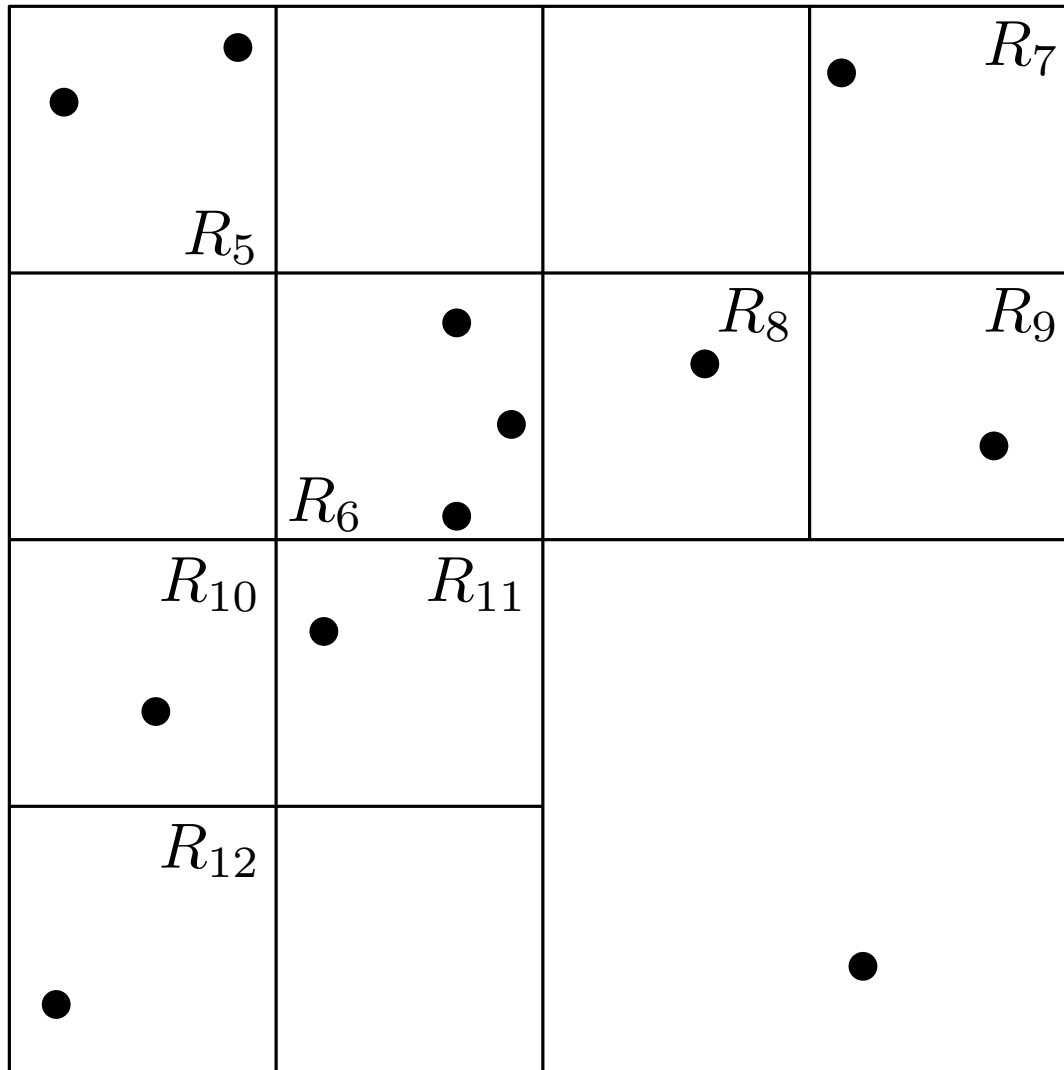


QT

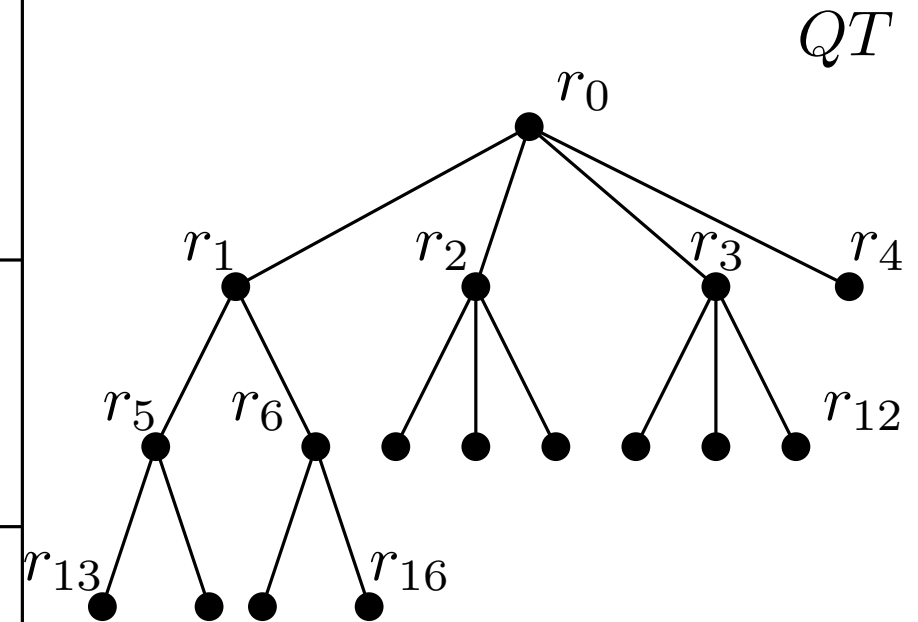
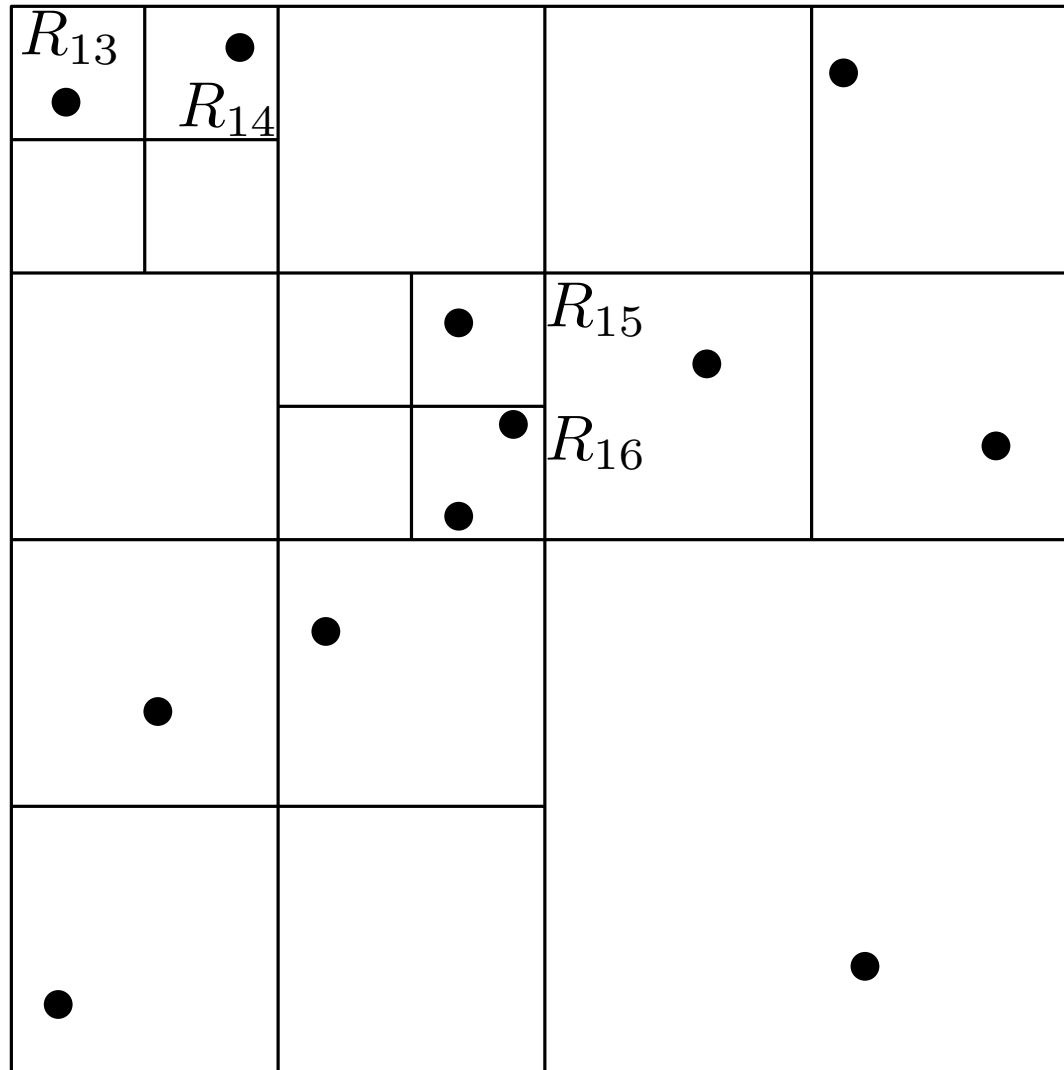
Quad-Tree



Quad-Tree

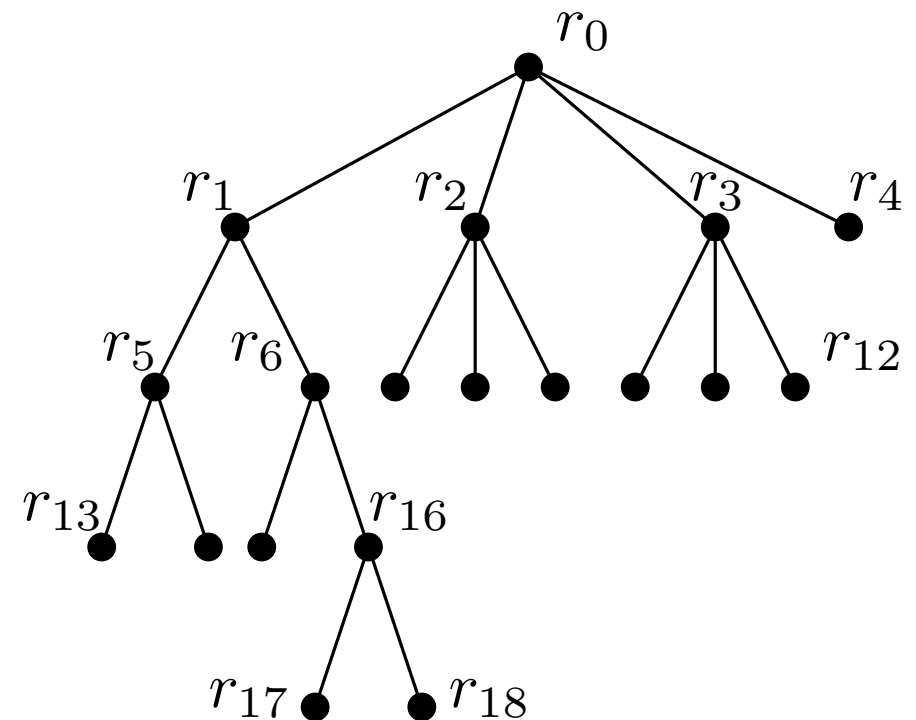
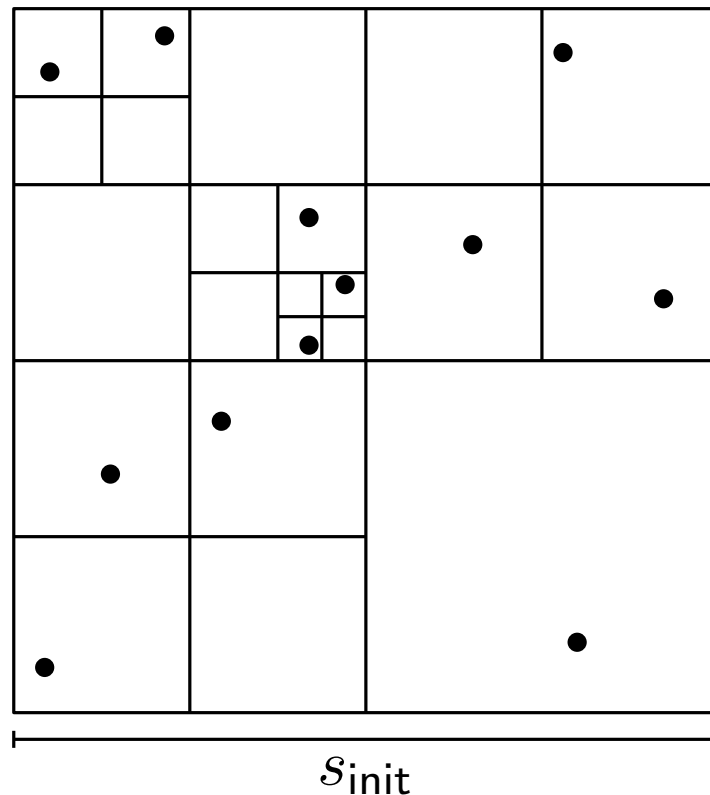


Quad-Tree

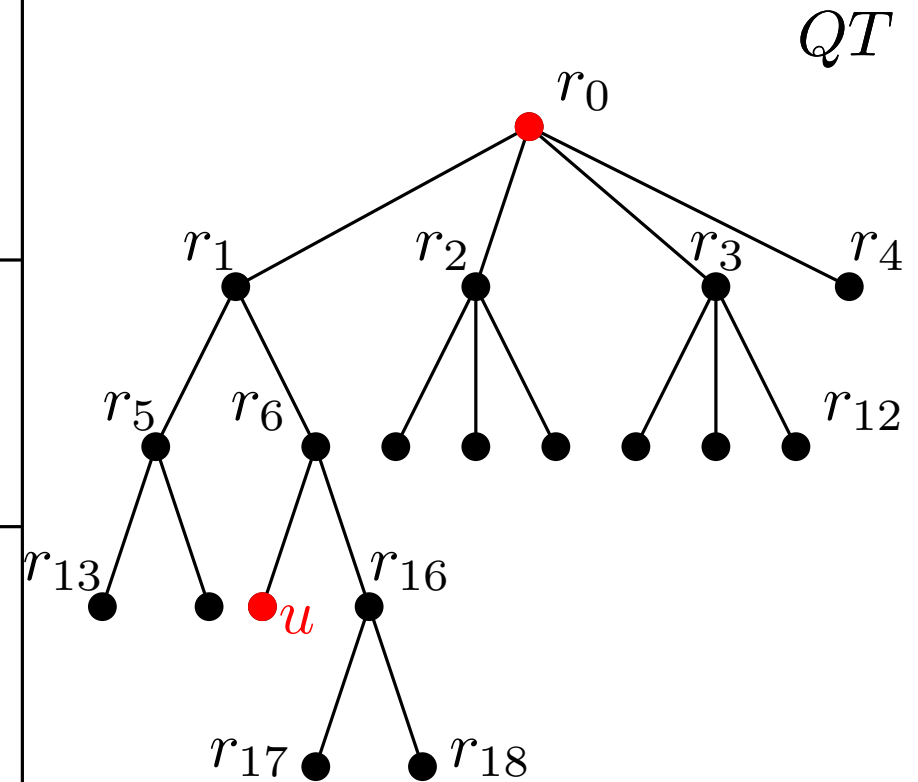
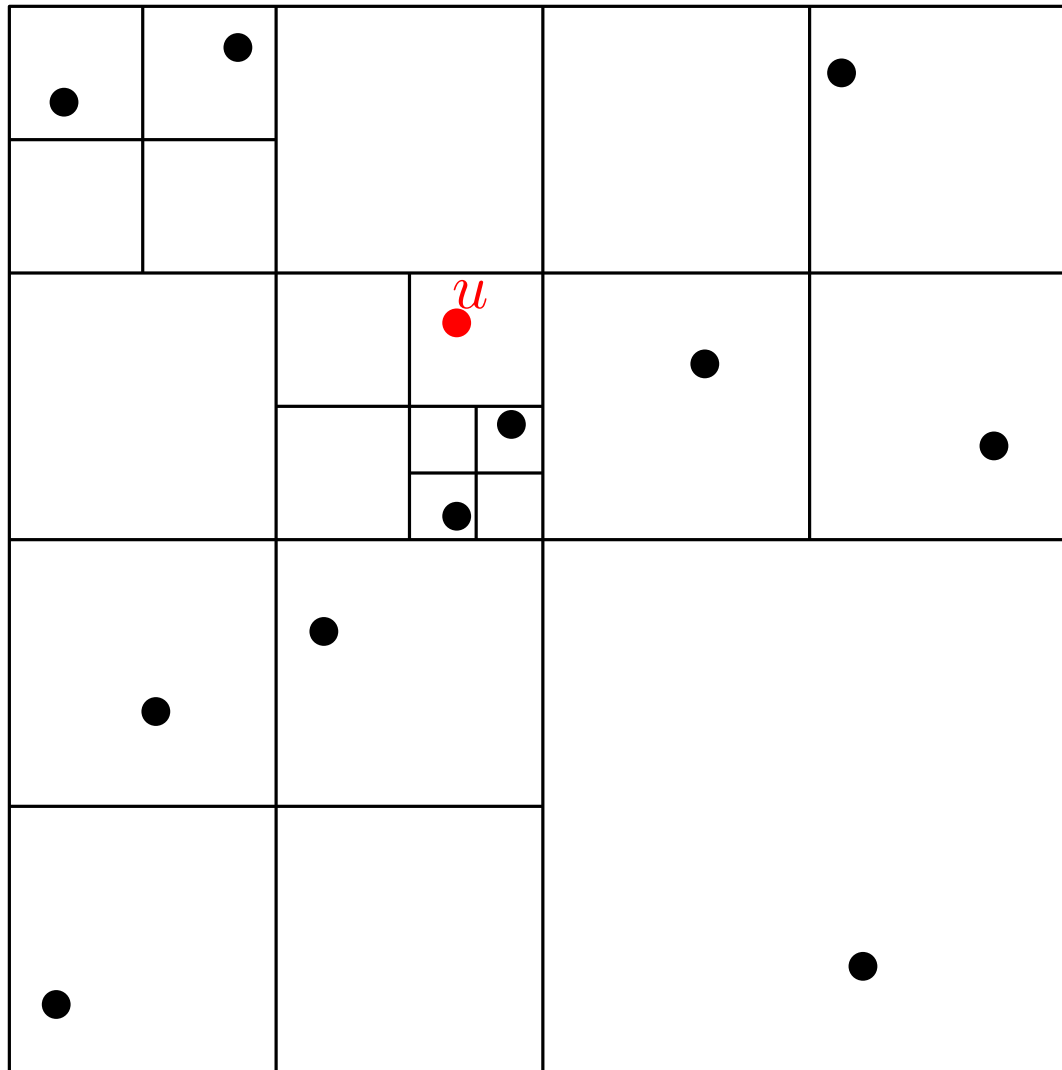


Eigenschaften Quad-Tree

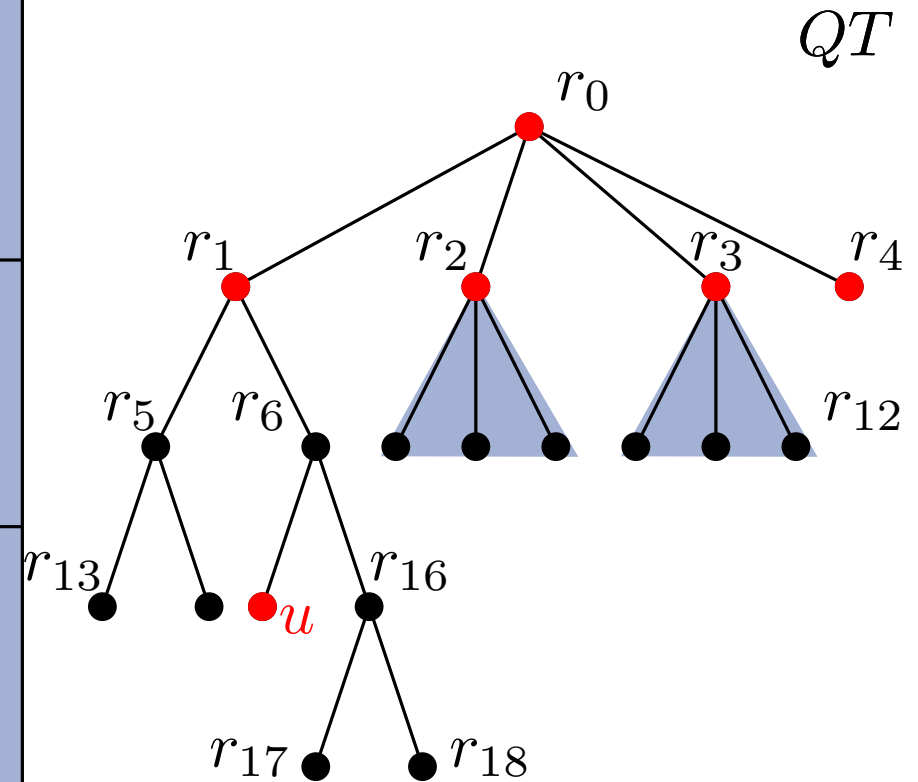
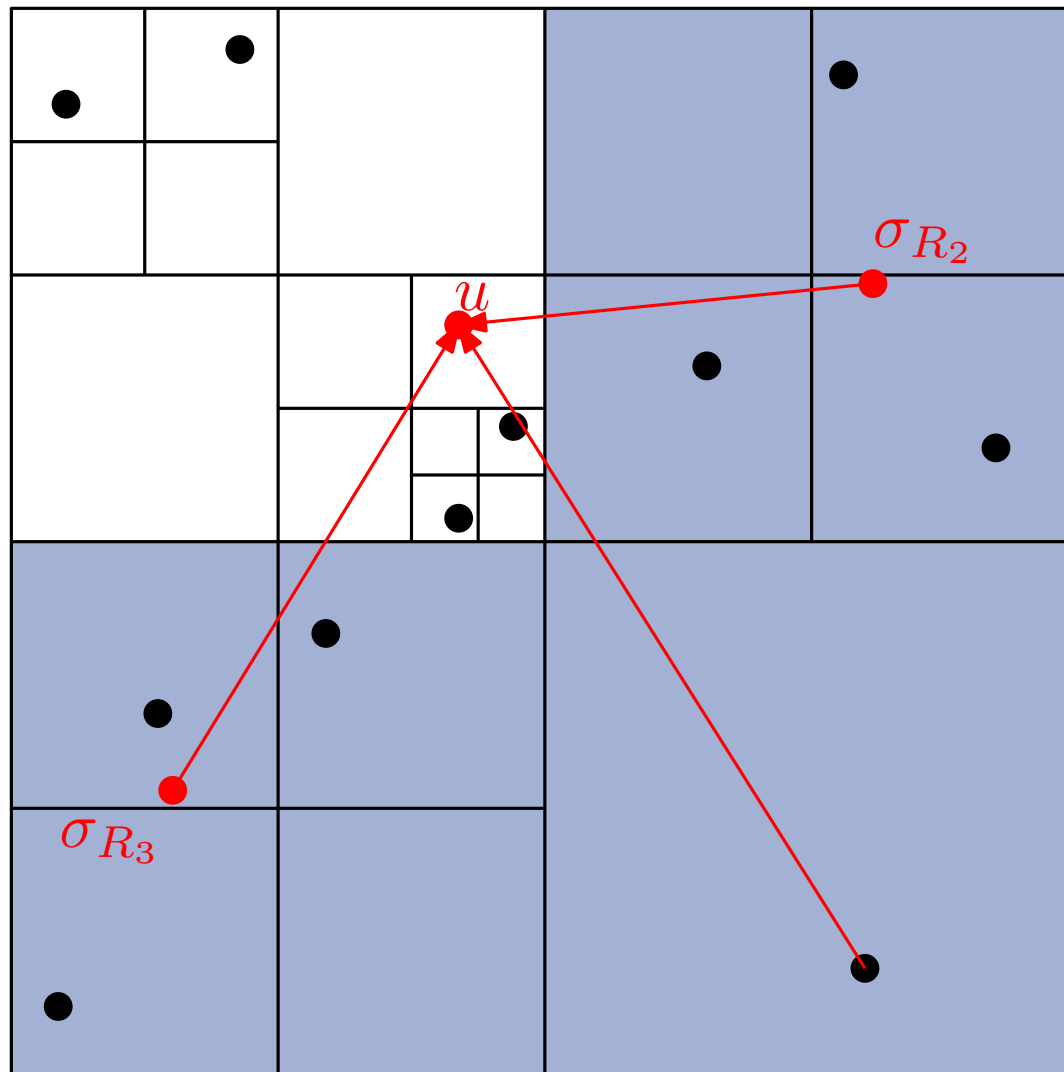
- Höhe $h \leq \log \frac{s_{\text{init}}}{d_{\text{min}}} + \frac{3}{2}$
- Zeit-/Speicherbedarf $O(hn)$
- komprimierter Quadtree in $O(n \log n)$ berechenbar
- $h \in O(\log n)$ bei gleichmäßiger Verteilung der Knoten



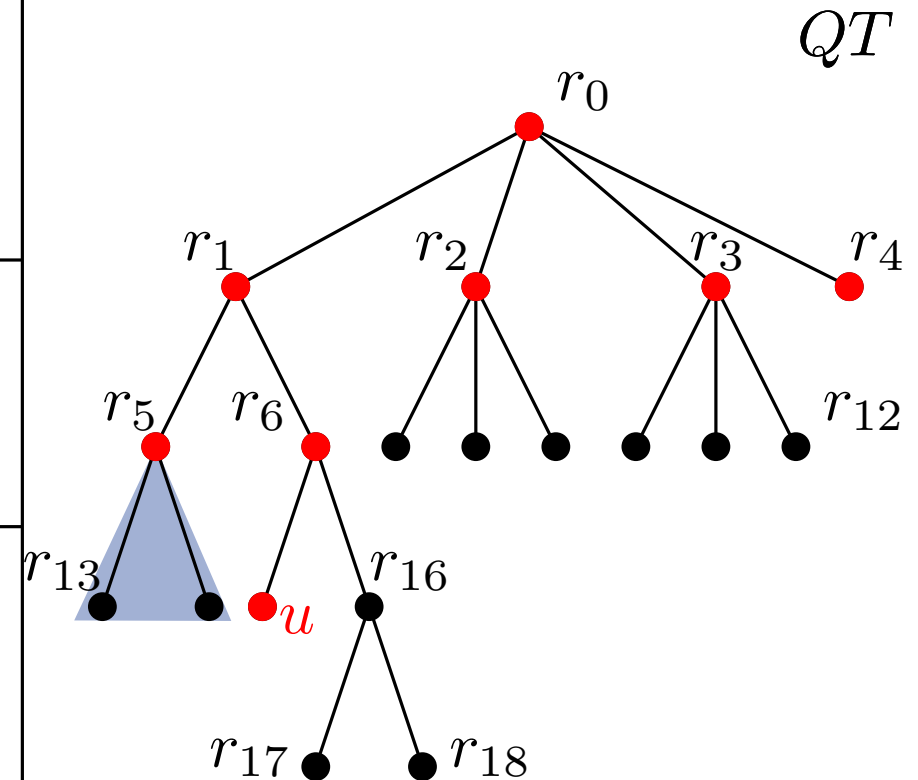
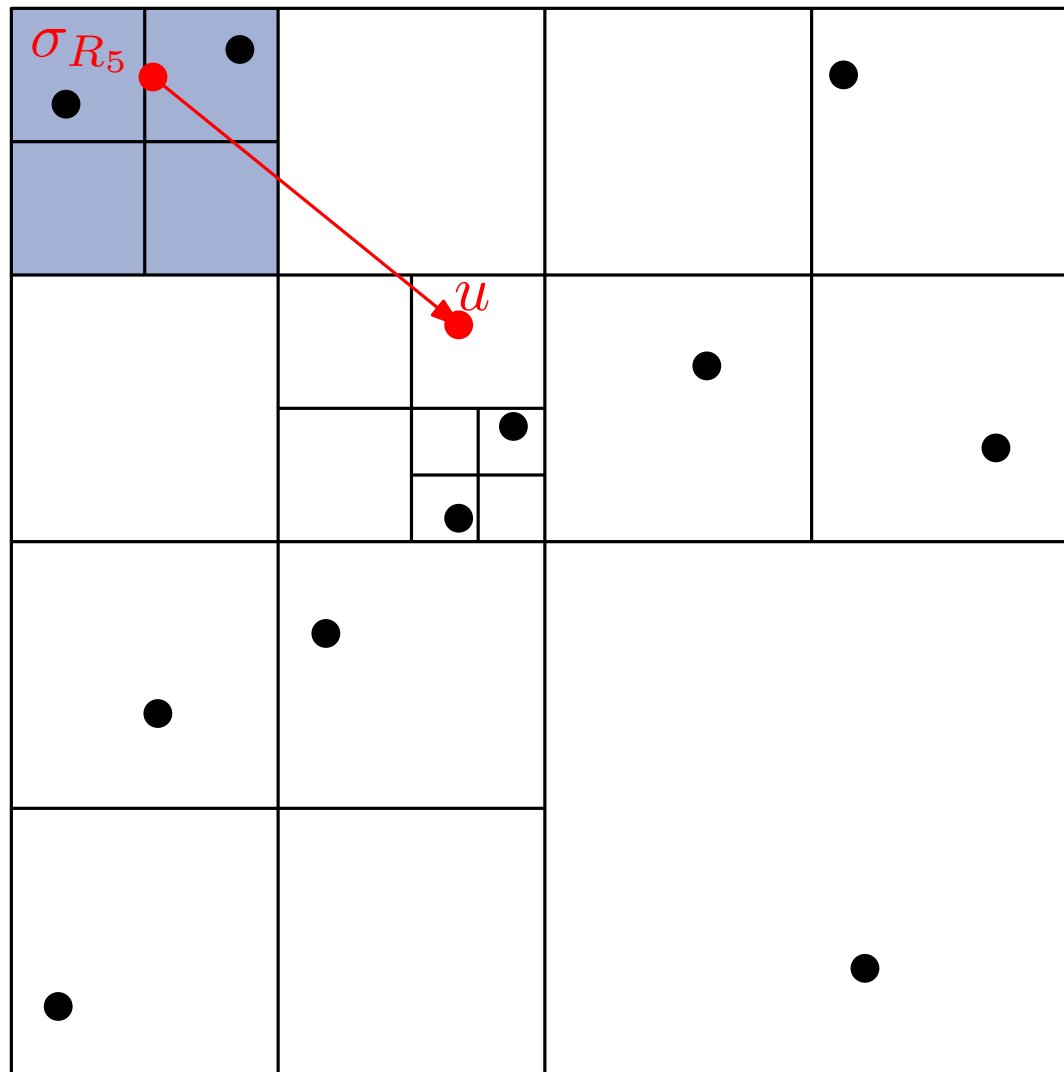
Kräfteberechnung mit Quad-Trees (Barnes, Hut, 1986)



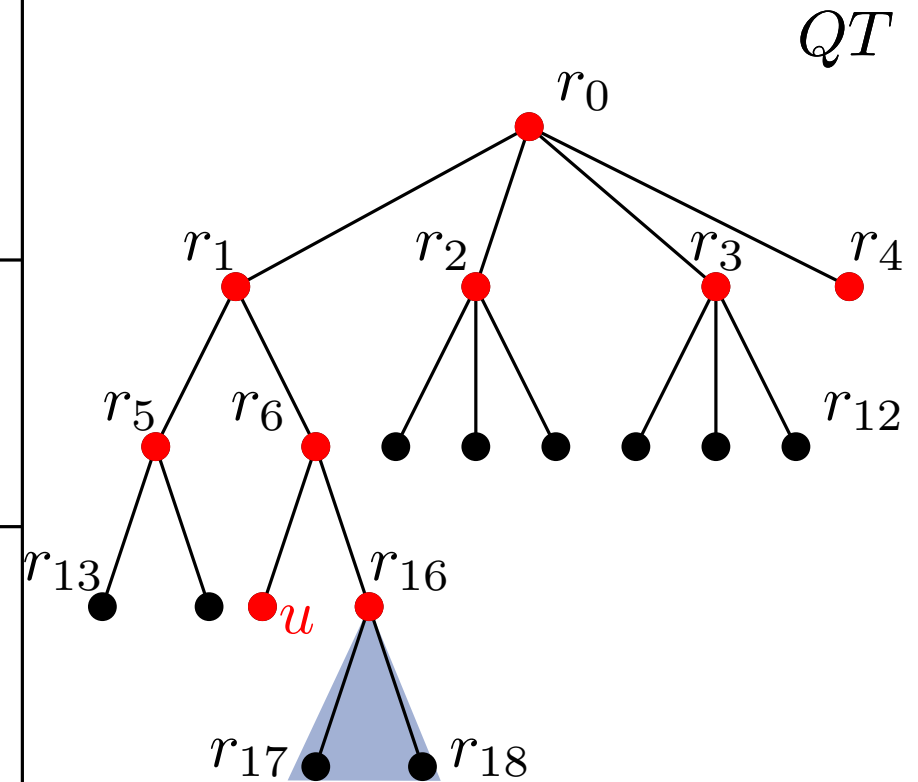
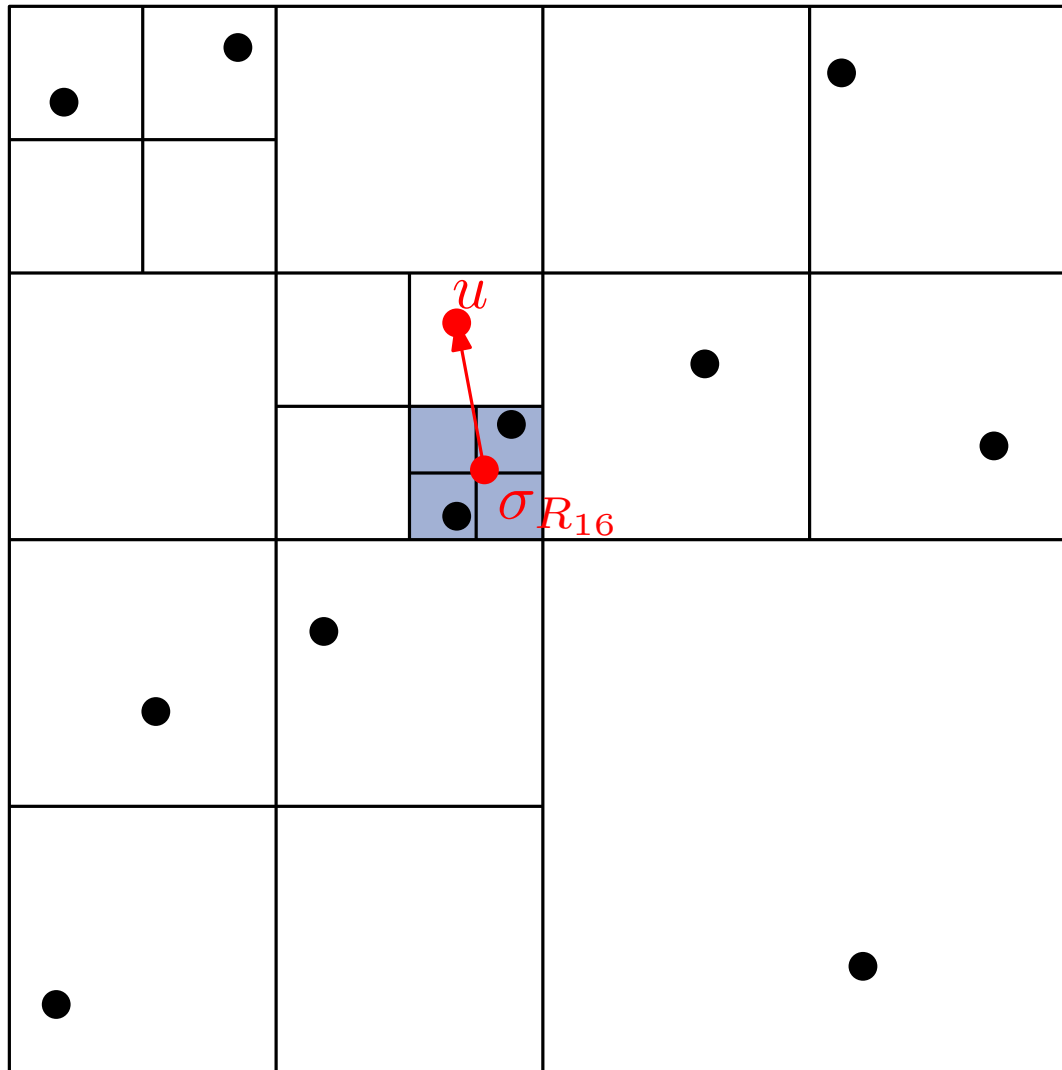
Kräfteberechnung mit Quad-Trees (Barnes, Hut, 1986)



Kräfteberechnung mit Quad-Trees (Barnes, Hut, 1986)



Kräfteberechnung mit Quad-Trees (Barnes, Hut, 1986)



Motivation

- klassischer Spring-Embedder für große Graphen zu langsam
- Schwachstelle zufällige Initialisierung der Knotenpositionen

Motivation

- klassischer Spring-Embedder für große Graphen zu langsam
- Schwachstelle zufällige Initialisierung der Knotenpositionen

GRIP – Graph dRawing with Intelligent Placement

(Gajer, Kobourov, 2004)

Ansatz

- Top-Down Vergrößerung des Graphen
- Bottom-Up Berechnung des Layouts
- sinnvolles Hinzufügen neuer Knoten
- kräftebasierte Verfeinerung

GRIP Algorithmus

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

$\mathcal{V} \leftarrow$ Filtrierung $V = V_0 \supset V_1 \supset \dots \supset V_k$

for $i = k$ **to** 0 **do**

foreach $v \in V_i \setminus V_{i+1}$ **do**

 berechne Nachbarschaften von v

 berechne initiale Position von v

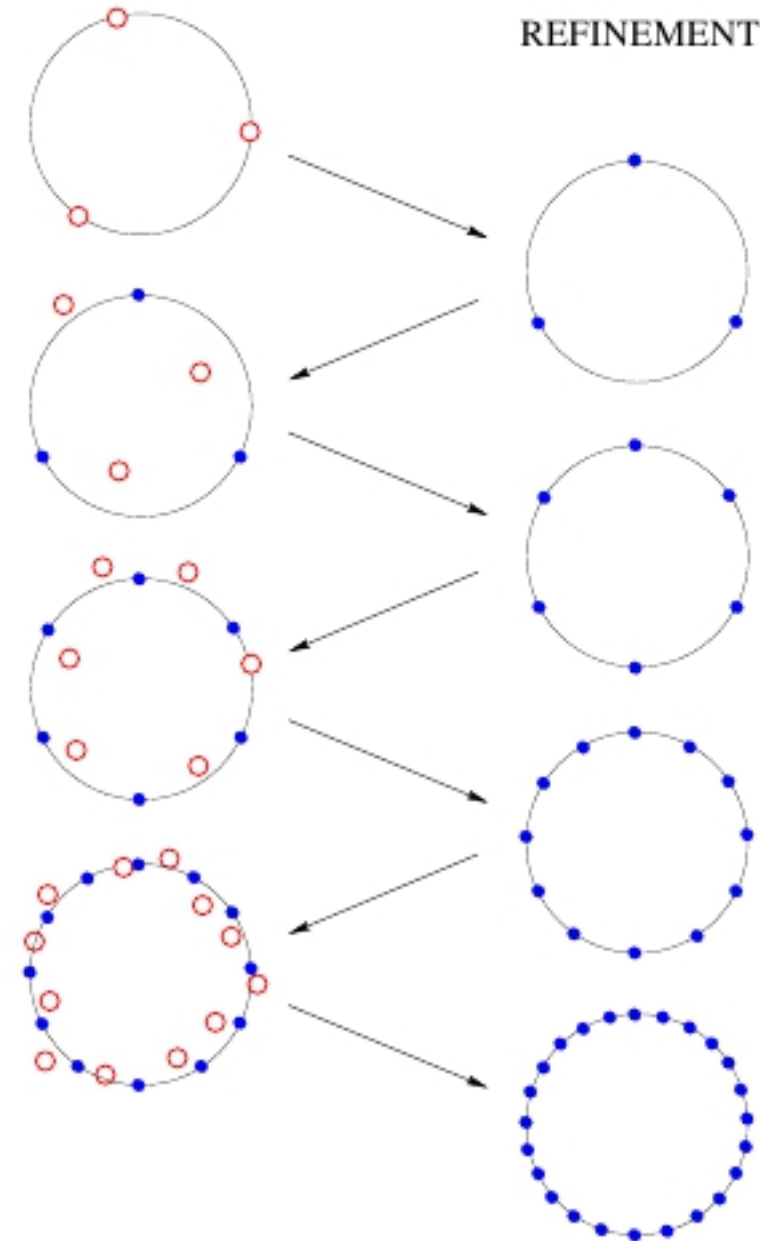
for $j = 1$ **to** rounds **do**

foreach $v \in V_i$ **do**

 kräftebasierte Relaxierung

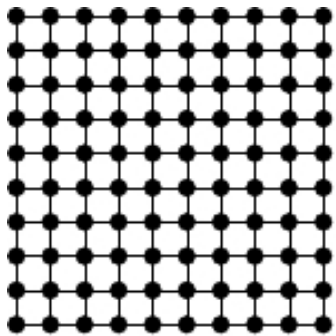
INITIAL PLACEMENT

REFINEMENT

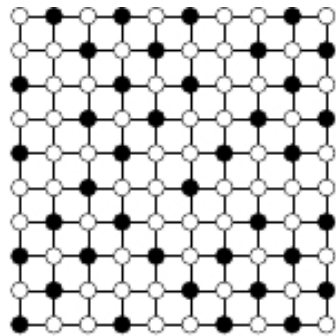


Maximal Independent Set (MIS) Filtrierung

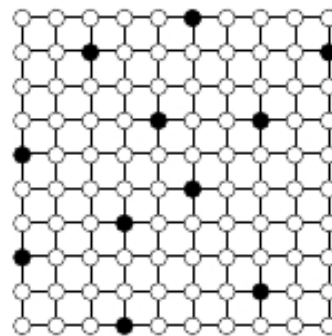
- Sequenz von Knotenteilmengen
 $V = V_0 \supset V_1 \supset \dots \supset V_k \supset \emptyset$
- V_i ist (inklusions-)maximale Knotenmenge, so dass
- Abstand in G zwischen Knoten in V_i ist $\geq 2^{i-1} + 1$
- gute Balance zwischen Tiefe und Größe der Level



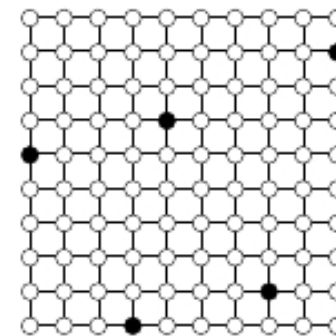
V_0



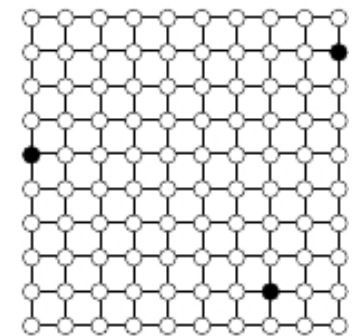
V_1



V_2



V_3



V_4

Algorithmus

- inkrementelles Vorgehen: gegeben V_i berechne V_{i+1}
- wähle zufälliges Element v in V_i
- entferne alle Elemente aus V_i mit Distanz $\leq 2^i$ zu v
- dazu BFS von v mit Suchtiefe 2^i
- wiederhole bis keine weiteren Knoten in V_i verbleiben

Algorithmus

- inkrementelles Vorgehen: gegeben V_i berechne V_{i+1}
- wähle zufälliges Element v in V_i
- entferne alle Elemente aus V_i mit Distanz $\leq 2^i$ zu v
- dazu BFS von v mit Suchtiefe 2^i
- wiederhole bis keine weiteren Knoten in V_i verbleiben

Tiefe der Filtrierung

- für letztes Level k gilt $2^k > \text{diam}(G)$
- Tiefe also $O(\log \text{diam}(G))$

Algorithmus

- inkrementelles Vorgehen: gegeben V_i berechne V_{i+1}
- wähle zufälliges Element v in V_i
- entferne alle Elemente aus V_i mit Distanz $\leq 2^i$ zu v
- dazu BFS von v mit Suchtiefe 2^i
- wiederhole bis keine weiteren Knoten in V_i verbleiben

Tiefe der Filtrierung

- für letztes Level k gilt $2^k > \text{diam}(G)$
- Tiefe also $O(\log \text{diam}(G))$

Alternative

- Matching-Vergrößerung, die rekursiv die Endknoten aller Matchingkanten verschmilzt

(Walshaw, JGAA 2003)

Phase 1

- bestimme für jeden Knoten $v \in V_i \setminus V_{i-1}$ optimale Position bzgl. der drei nächsten Knoten aus V_{i-1} (\rightarrow BFS)

Phase 2

- führe kräftebasierte Verfeinerung durch, wobei Kräfte nur lokal zu einer konstanten Anzahl nächster Nachbarn in V_i berechnet werden

Phase 1

- bestimme für jeden Knoten $v \in V_i \setminus V_{i-1}$ optimale Position bzgl. der drei nächsten Knoten aus V_{i-1} (\rightarrow BFS)

Phase 2

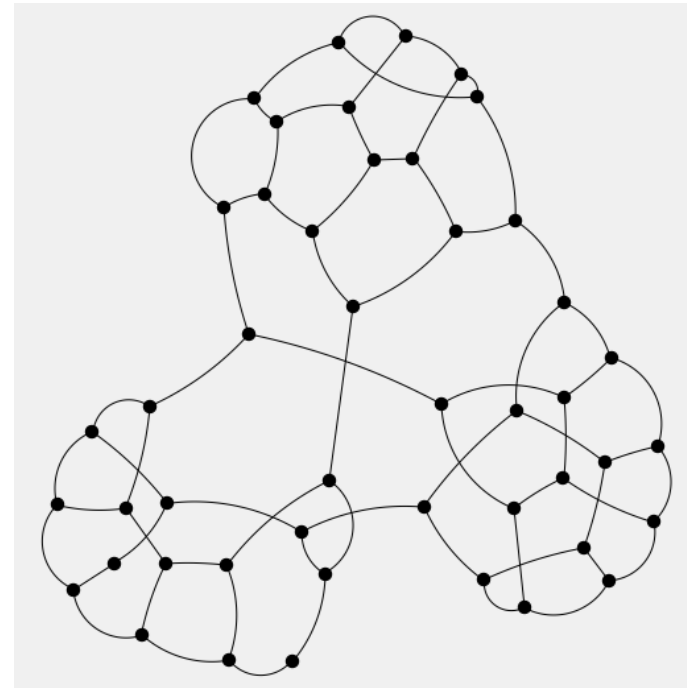
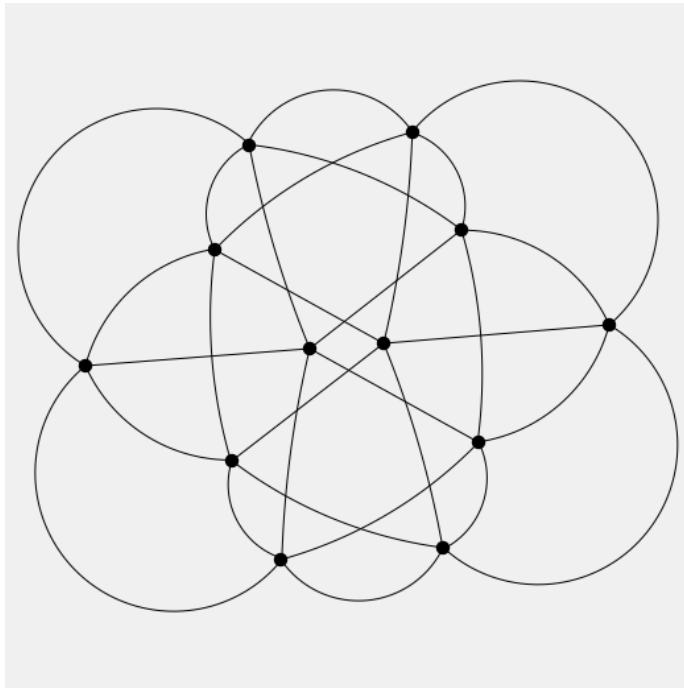
- führe kräftebasierte Verfeinerung durch, wobei Kräfte nur lokal zu einer konstanten Anzahl nächster Nachbarn in V_i berechnet werden

Eigenschaften GRIP

- intelligente schrittweise Berechnung eines guten Startlayouts durch MIS-Vergrößerung
- deutlich schnellere Konvergenz
- Graphen mit > 10.000 Knoten in wenigen Sekunden (2004)

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

- Kanten sind Kreisbögen
- optimale Winkelauflösung $2\pi / \deg(v)$ in jedem Knoten v
- zusätzliche Rotations- und Tangentialkräfte

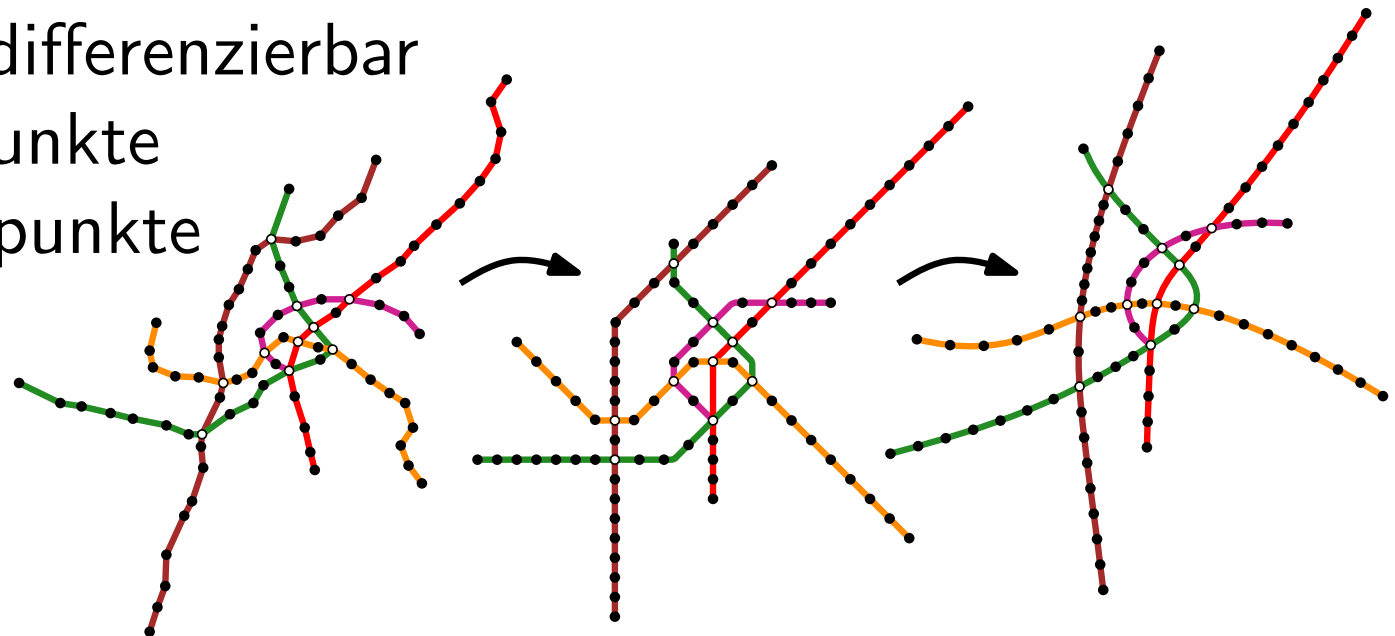


Lombardi-Spring-Embedder (Chernobelskiy et al. 2012)

- Kanten sind Kreisbögen
- optimale Winkelauflösung $2\pi / \deg(v)$ in jedem Knoten v
- zusätzliche Rotations- und Tangentialkräfte

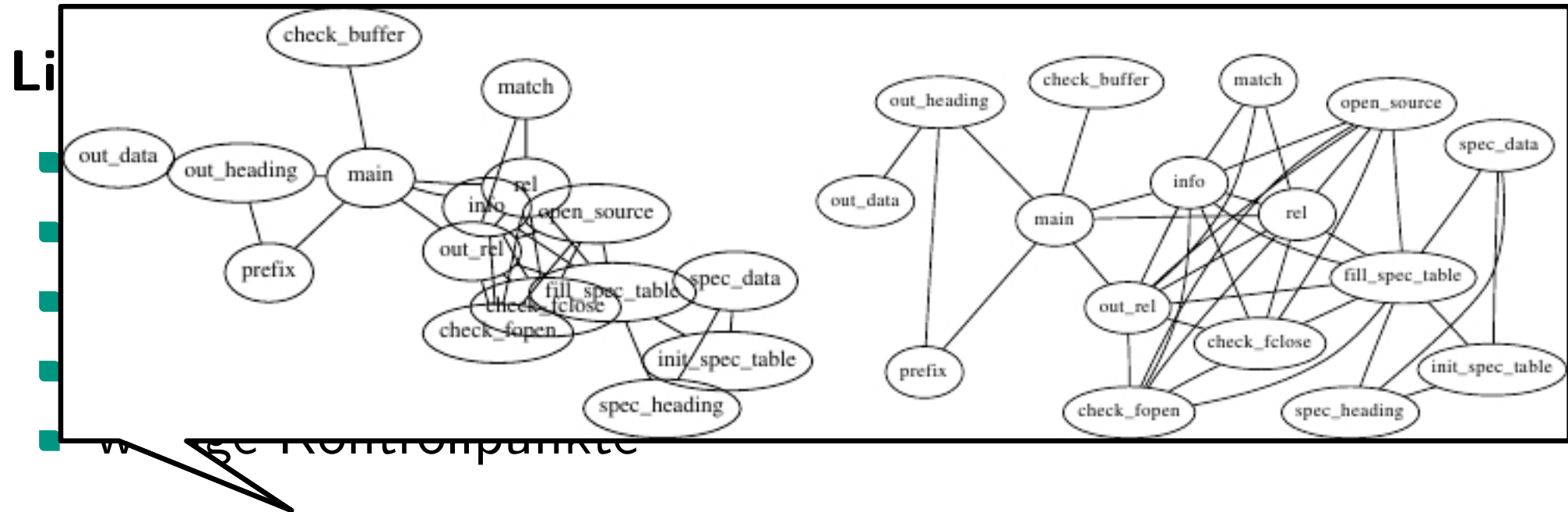
Linienpläne mit Bézierkurven (Fink et al. 2013)

- modelliere Kantenfolgen als Bézierkurve
- Kräfte auf Graph-Knoten und Kontroll-Knoten
- einzelne Linien differenzierbar
- wenige Wendepunkte
- wenige Kontrollpunkte



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

- Kanten sind Kreisbögen
- optimale Winkelauflösung $2\pi / \deg(v)$ in jedem Knoten v
- zusätzliche Rotations- und Tangentialkräfte



Realistische Knotengrößen (Gansner, North 1998)

- Abstoßung berücksichtigt Knotengrößen

Kräftebasierte Verfahren sind

- leicht verständlich und implementierbar
- keine besonderen Anforderungen an Eingabegraph
- je nach Graph (klein & dünn) erstaunlich gute Layouts (Symmetrien, Clusterung, ...)
- leicht adaptierbar und konfigurierbar
- robust
- skalierbar

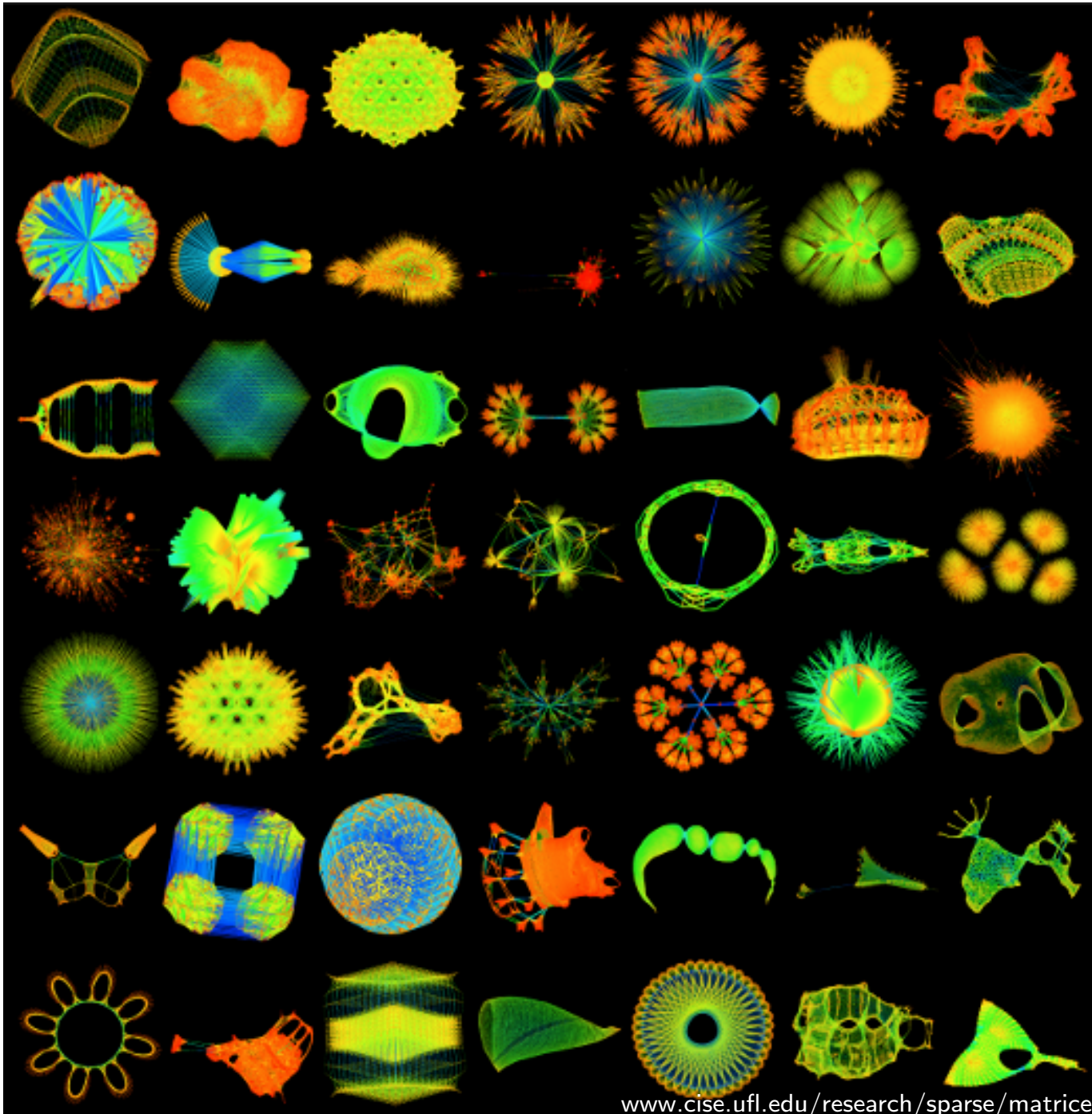
Kräftebasierte Verfahren sind

- leicht verständlich und implementierbar
- keine besonderen Anforderungen an Eingabegraph
- je nach Graph (klein & dünn) erstaunlich gute Layouts (Symmetrien, Clusterung, ...)
- leicht adaptierbar und konfigurierbar
- robust
- skalierbar

Aber...

- üblicherweise keine Qualitäts- und Laufzeitgarantien
- schlechtes Startlayout → langsame Konvergenz
- evtl. langsam für große Graphen
- oft fine-tuning durch Experten nötig

Galerie



www.cise.ufl.edu/research/sparse/matrices

©Davis & Hu