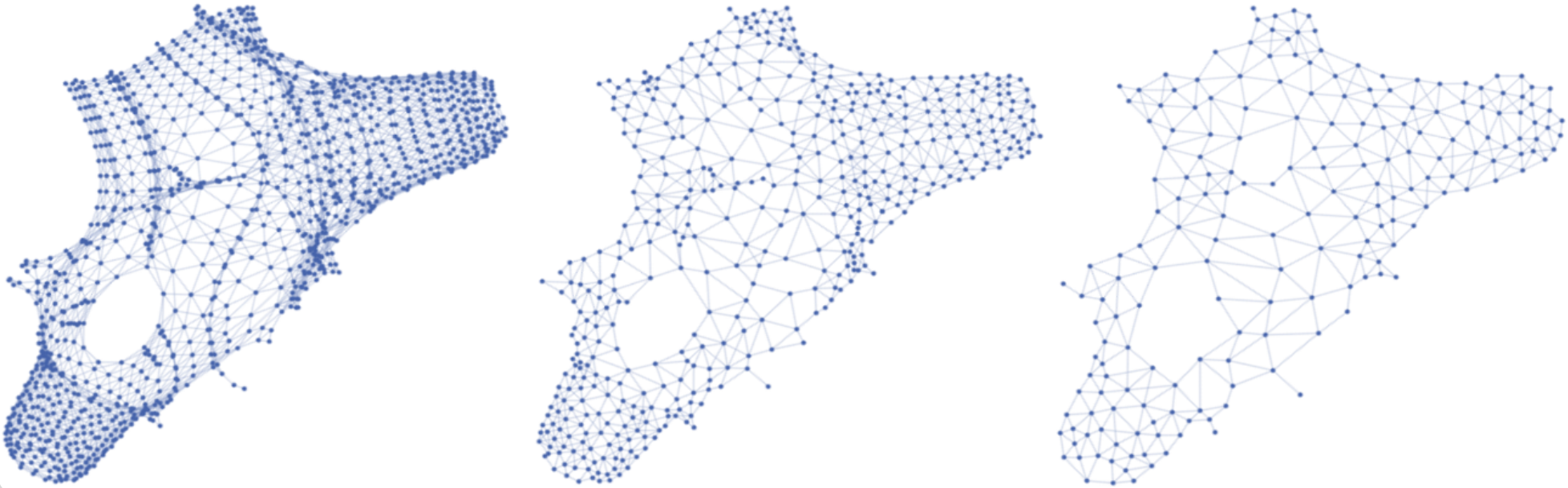
Algorithmen zur Visualisierung von Graphen

Kräftebasierte Verfahren

Marcus Krug | WS 2011/12

INSTITUTE OF THEORETICAL INFORMATICS
KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT)



Motivation

Was zeichnet ein gutes Layout aus?

- gute Verteilung der Knoten in der Ebene
- adjazente Knoten nah

Motivation

Was zeichnet ein gutes Layout aus?

- gute Verteilung der Knoten in der Ebene
- adjazente Knoten nah

Problem GRAPHEINBETTUNG

Instanz: Graph mit vorgegebenen Kantenlänge

Frage: Existiert Zeichnung, die Kantenlängen realisiert?

Motivation

Was zeichnet ein gutes Layout aus?

- gute Verteilung der Knoten in der Ebene
- adjazente Knoten nah

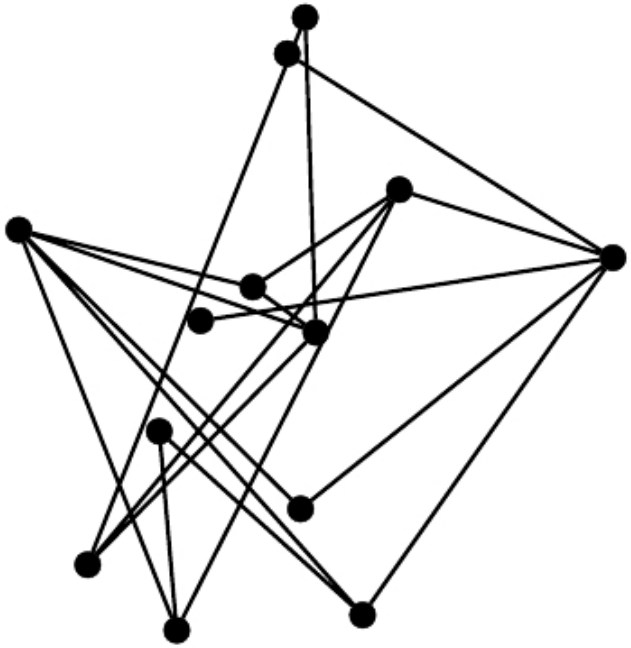
Problem GRAPHEINBETTUNG

Instanz: Graph mit vorgegebenen Kantenlänge

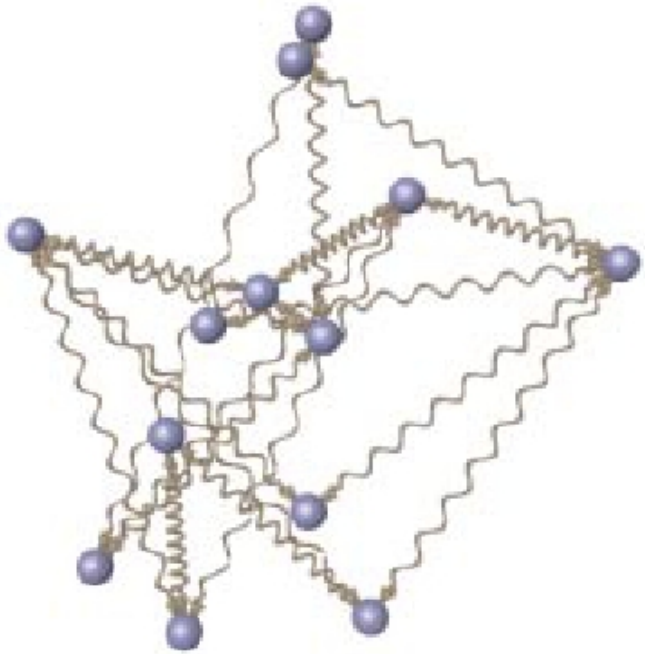
Frage: Existiert Zeichnung, die Kantenlängen realisiert?

- NP-schwer für Kantenlängen $\{1, 2\}$ sowie [Saxe, '80]
- Planar mit Einheitskantenlänge [Eades & Wormald, '85]

Modell

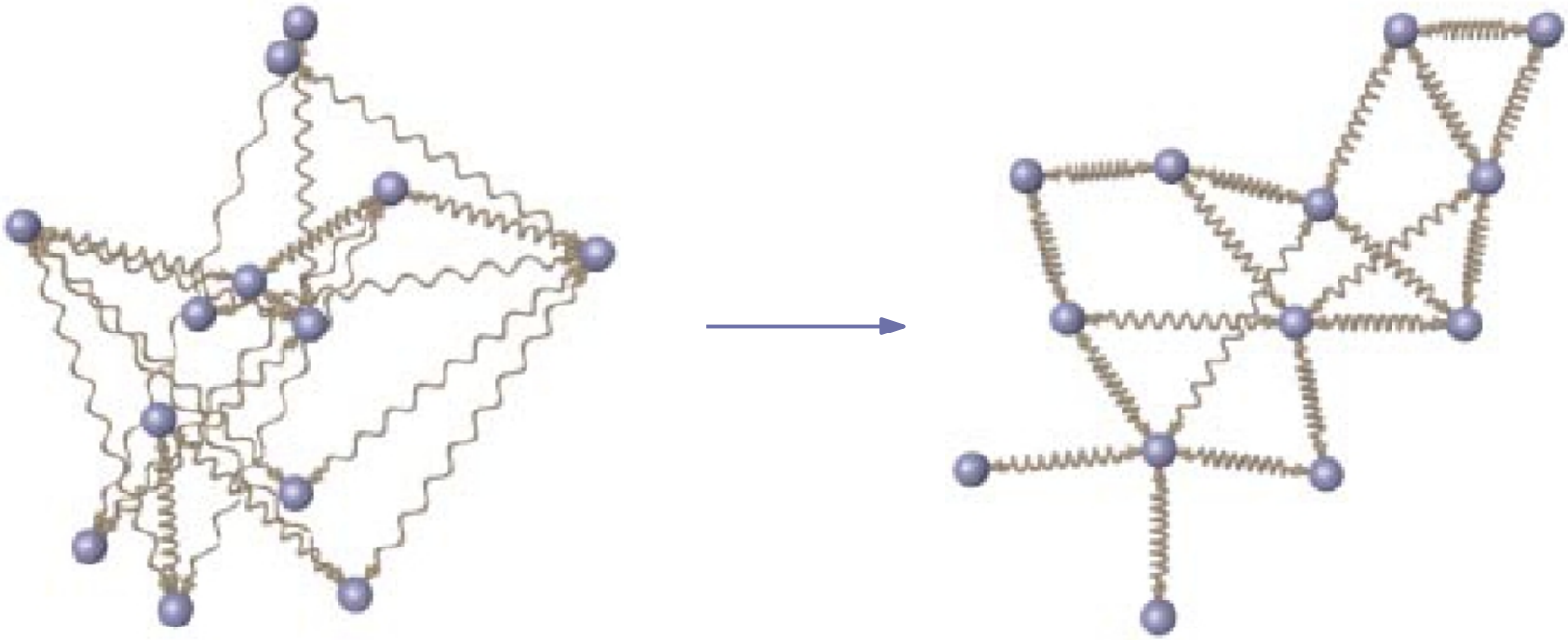


Modell



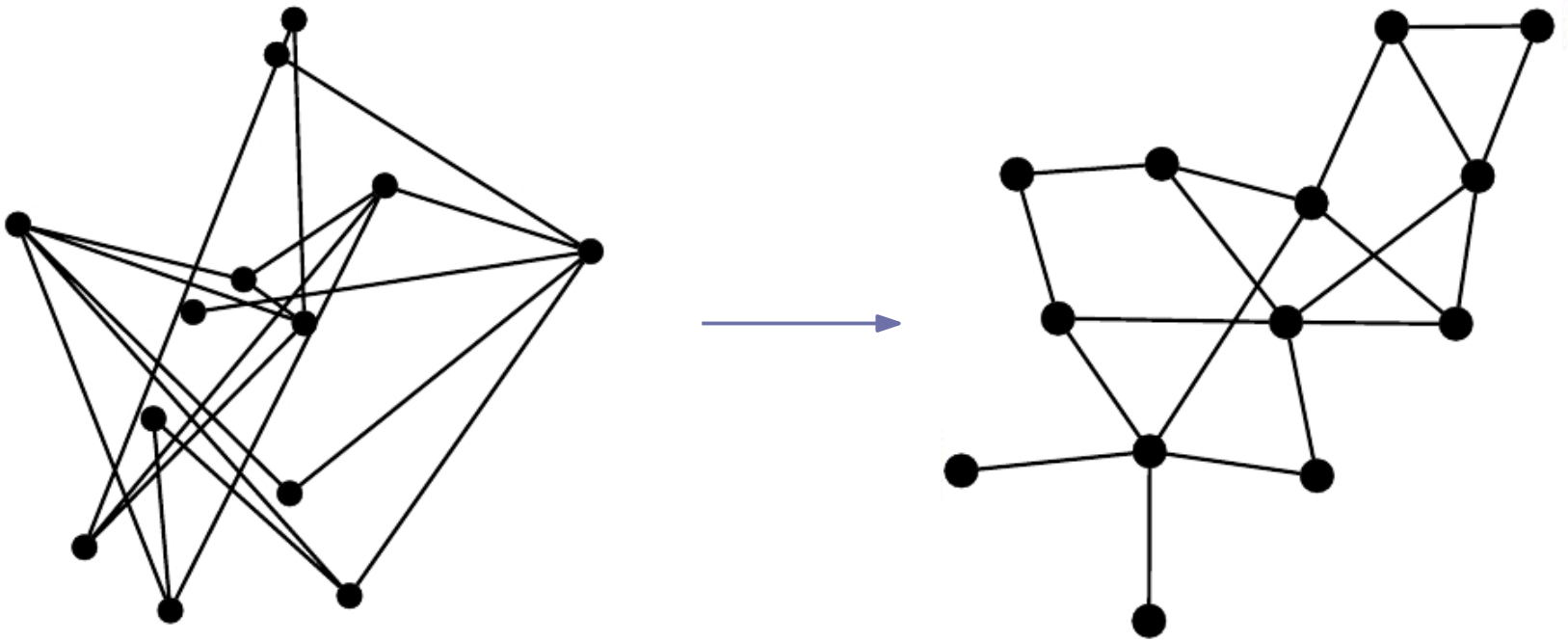
“To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system . . .

Modell



“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]

Modell



“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]

Terminologie im Folgenden

ℓ	Ideallänge der Feder
$p_v = (x_v, y_v)$	Position von Knoten v
$\|\|p_u - p_v\|\|$	Euklidischer Abstand zwischen u und v
$\overrightarrow{p_u p_v}$	Einheitsvektor von u nach v

Springembedder [Eades, ’84]

Modell

- abstoßende Kraft zwischen nicht adjazenten Knoten u und v

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

Springembedder [Eades, ’84]

Modell

- abstoßende Kraft zwischen 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

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

Springembedder [Eades, '84]

Modell

- abstoßende Kraft zwischen nicht adjazenten Knoten u und v

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

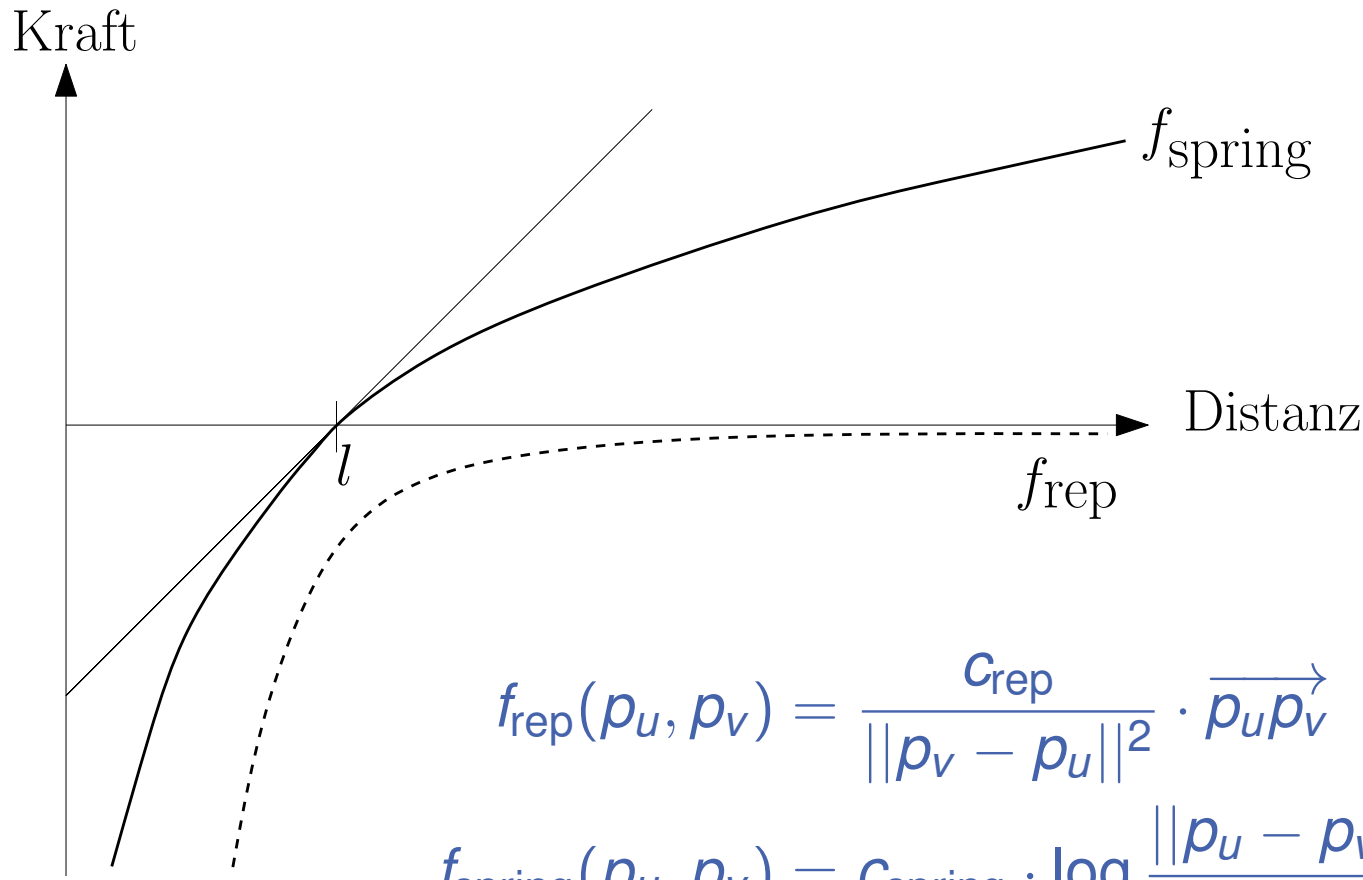
- anziehende Kraft zwischen adjazenten Knoten u und v

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

- Verschiebungsvektor

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

Springembedder [Eades, ’84]



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

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

Algorithmus

Algorithmus 1: Spring Embedder

Eingabe : $G = (V, E)$ zusammenhängender ungerichteter Graph mitAnfangslayout $p = (p_v)_{v \in V}$ und $K \in \mathbb{N}$, $\varepsilon \in \mathbb{R}$

Ausgabe : Layout p mit „niedriger innerer Anspannung“

1 $t \leftarrow 1$

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

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

4 $F_v(t) := \sum_{u:\{u,v\} \notin E} f_{rep}(p_u, p_v) + \sum_{u:\{u,v\} \in E} f_{spring}(p_u, p_v)$

Algorithmus

Algorithmus 1: Spring Embedder

Eingabe : $G = (V, E)$ zusammenhängender ungerichteter Graph mitAnfangslayout $p = (p_v)_{v \in V}$ und $K \in \mathbb{N}$, $\varepsilon \in \mathbb{R}$

Ausgabe : Layout p mit „niedriger innerer Anspannung“

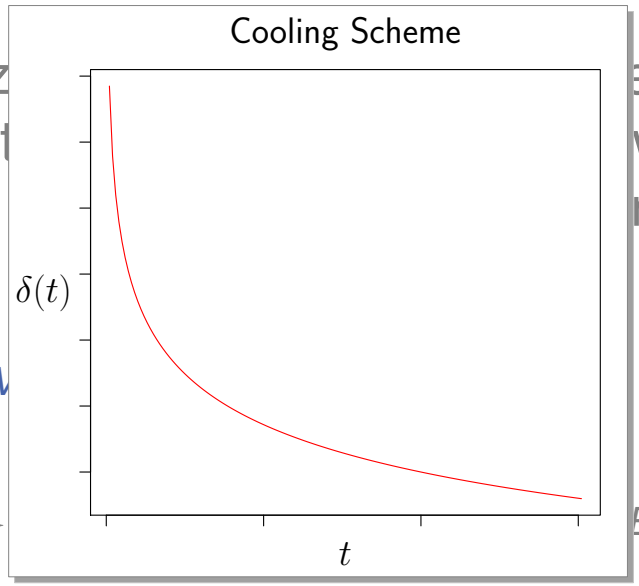
1 $t \leftarrow 1$
2 **while** $t < K$ **and** $\max_{v \in V} \|F_v(t)\| > \varepsilon$ **do**
3 **foreach** $v \in V$ **do**
4 $F_v(t) := \sum_{u:\{u,v\} \notin E} f_{rep}(p_u, p_v) + \sum_{u:\{u,v\} \in E} f_{spring}(p_u, p_v)$
5 **foreach** $v \in V$ **do**
6 setze $p_v := p_v + \delta \cdot F_v(t)$
7 $t \leftarrow t + 1$

Demo

Cooling

Algorithmus 1: Spring Embedder

Eingabe : $G = (V, E)$ z… erichteter Graph
mitAnfangslayout… wert ε
Ausgabe : Layout p mit … nung“



1 $t \leftarrow 1$
2 **while** $t < K$ **and** $\max_{v \in V}$…
3 **foreach** $v \in V$ **do**
4 $F_v(t) := \sum_{u:\{u,v\} \in E} f_{spring}(p_u, p_v)$
5 **foreach** $v \in V$ **do**
6 setze $p_v := p_v + \delta(t) \cdot F_v(t)$
7 $t \leftarrow t + 1$

Diskussion

Vorteile

- Algorithmus ist simpel
- gute Ergebnisse für kleine und mittel-große Graphen
- Symmetrien, Struktur

Diskussion

Vorteile

- Algorithmus ist simpel
- gute Ergebnisse für kleine und mittel-große Graphen
- Symmetrien, Struktur

Nachteile

- System am Ende möglicherweise nicht stabil
- lokale Minima
- f_{spring} in $\mathcal{O}(|E|)$ und f_{rep} in $\mathcal{O}(|V|^2)$ inkl. $\sqrt{\cdot}$ und $\log(\cdot)$

Variante [Fruchterman & Reingold '91]

Modell

- abstoßende Kraft zwischen allen Knotenpaaren u und v

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

Variante [Fruchterman & Reingold '91]

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

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

Variante [Fruchterman & Reingold '91]

Modell

- abstoßende Kraft zwischen allen Knotenpaaren u und v

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

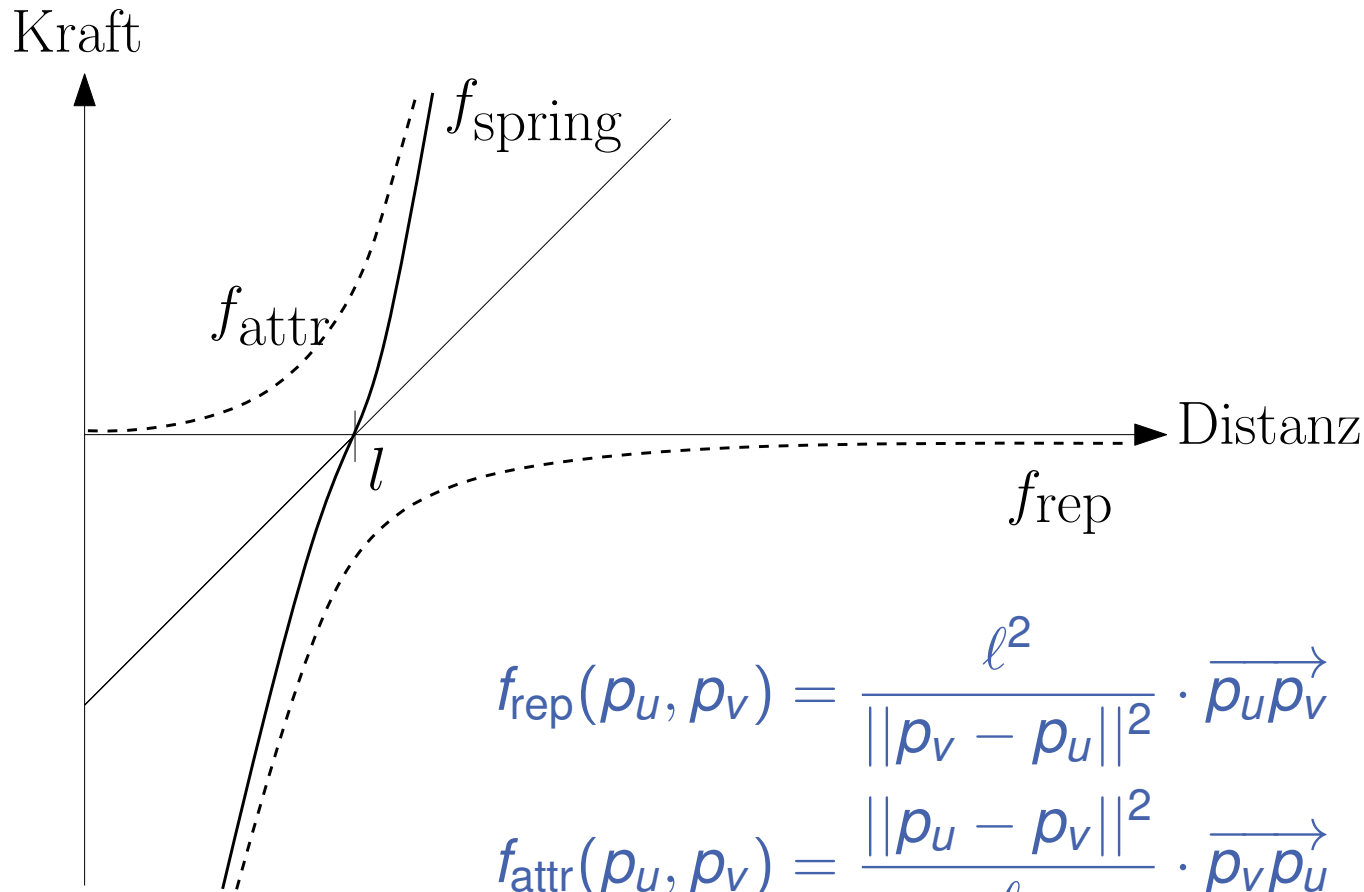
- anziehende Kraft zwischen adjazenten Knoten u und v

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

- resultierende Federkraft zwischen adjazenten Knoten u und v

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

Variante [Fruchterman & Reingold '91]

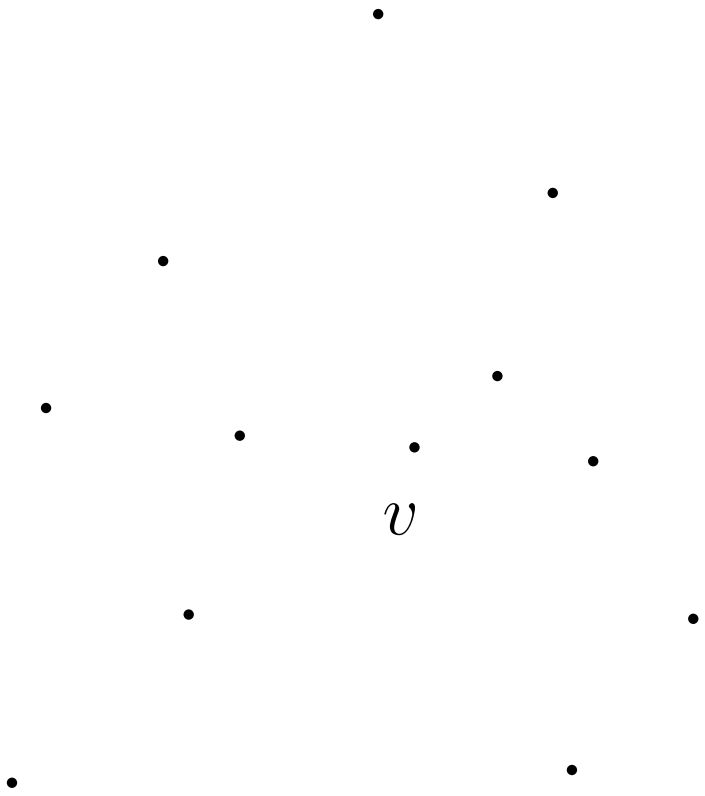


$$f_{\text{rep}}(p_u, p_v) = \frac{\ell^2}{||p_v - p_u||^2} \cdot \overrightarrow{p_u p_v}$$

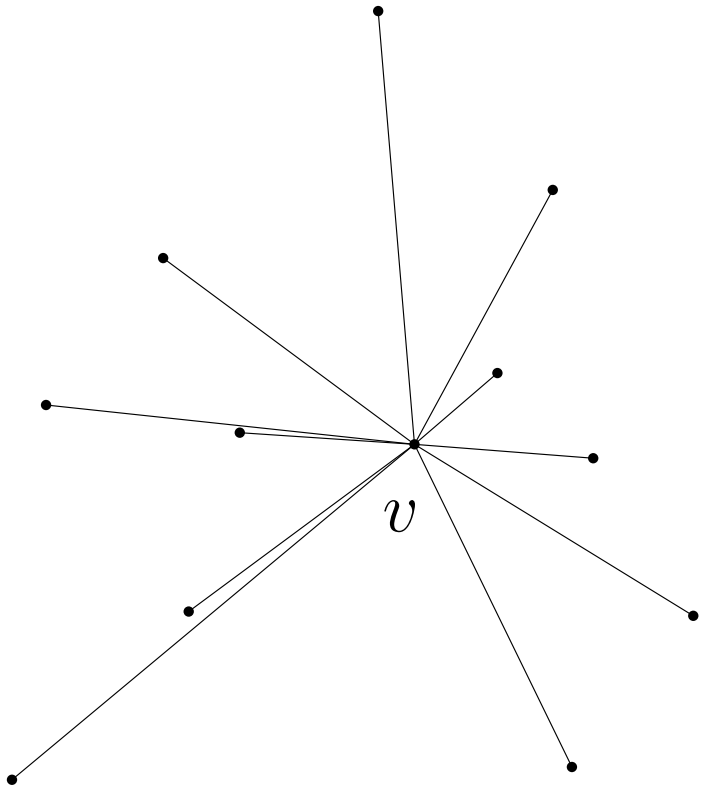
$$f_{\text{attr}}(p_u, p_v) = \frac{||p_u - p_v||^2}{\ell} \cdot \overrightarrow{p_v p_u}$$

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

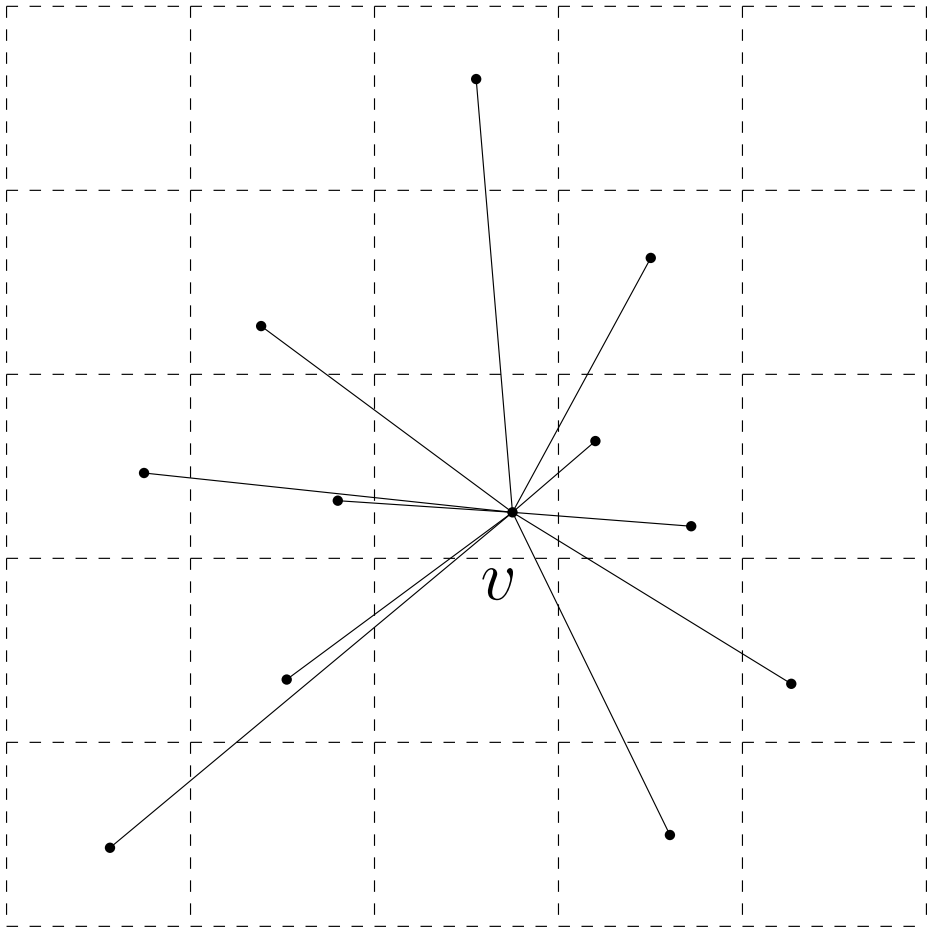
Grid-Technik



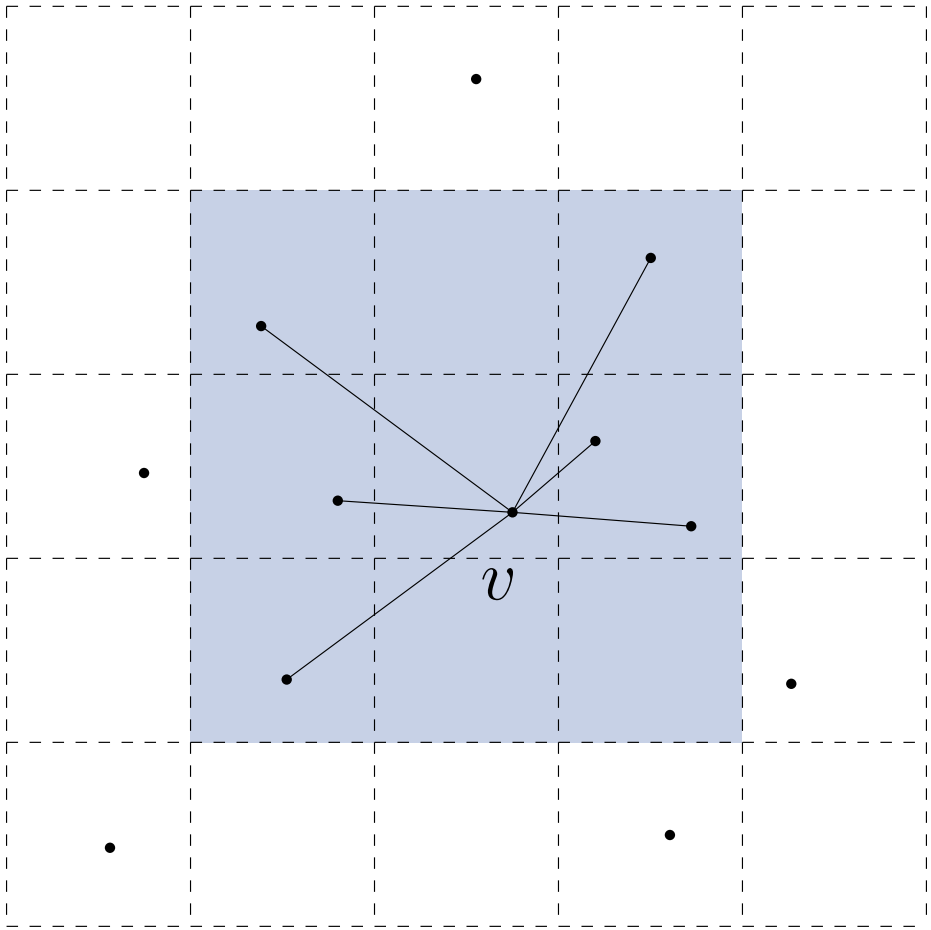
Grid-Technik



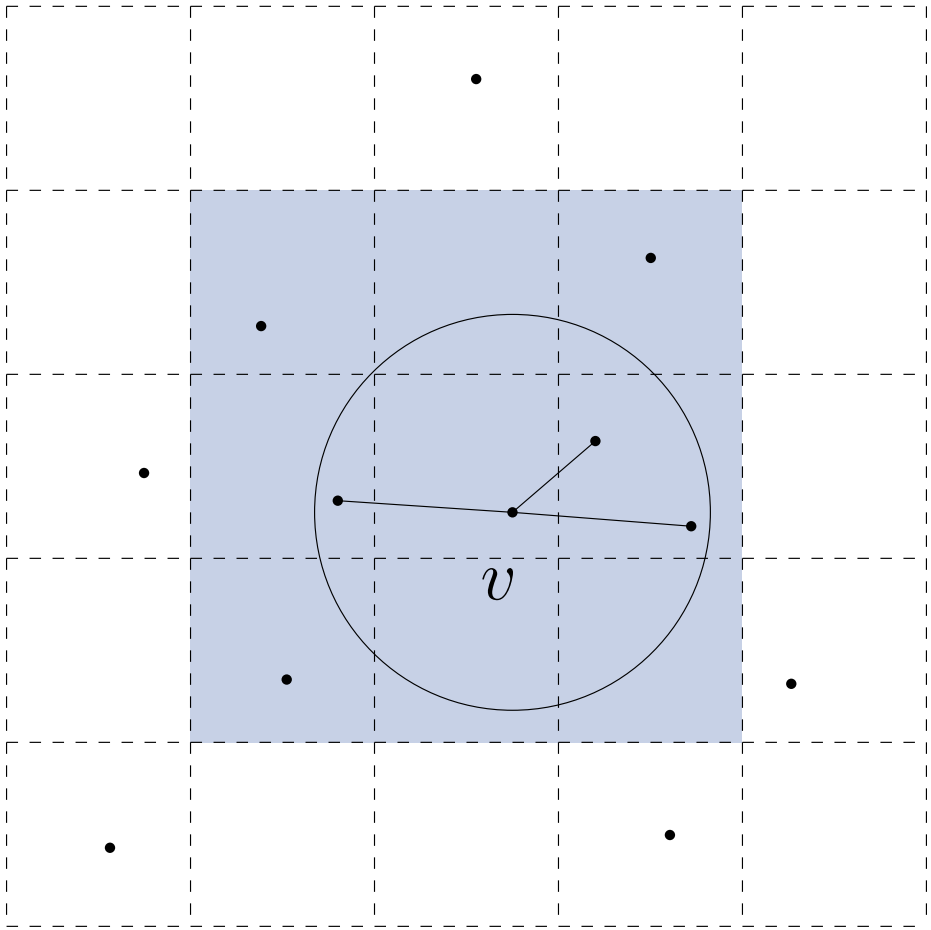
Grid-Technik



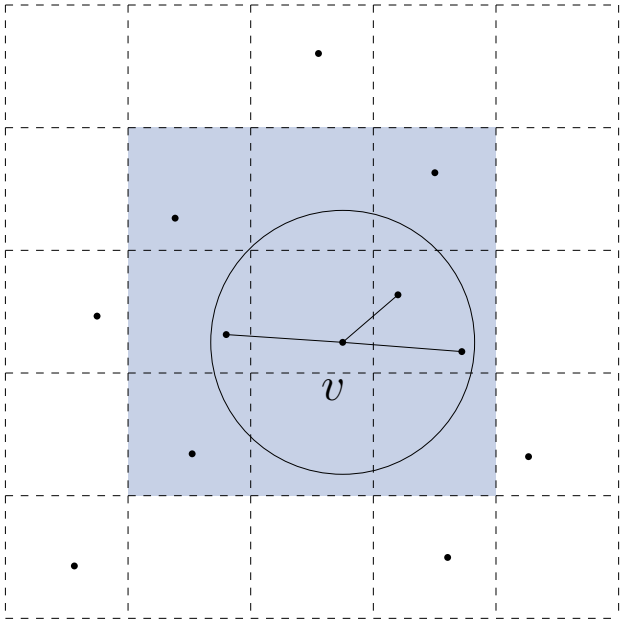
Grid-Technik



Grid-Technik



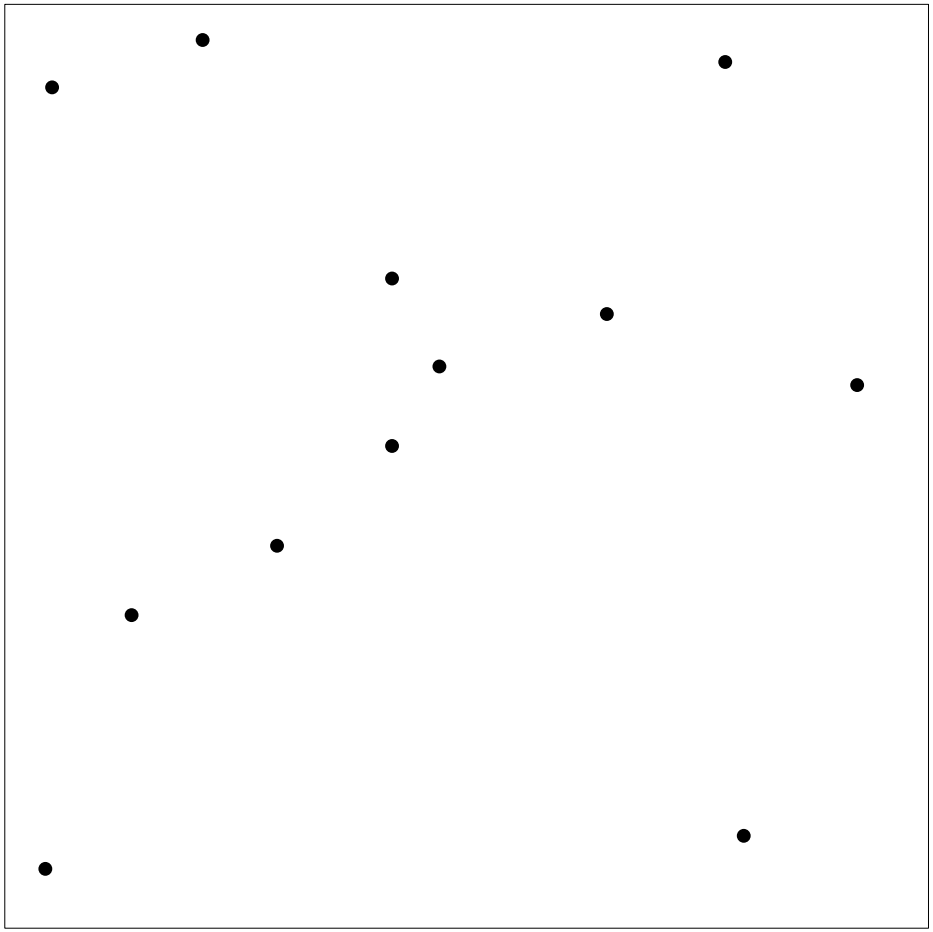
Grid-Technik



Nachteil

- keine Worst-Case Verbesserung
- Qualitätsverlust
  - Abschneiden der Kräfte
  - Oszillation

Quad-Tree

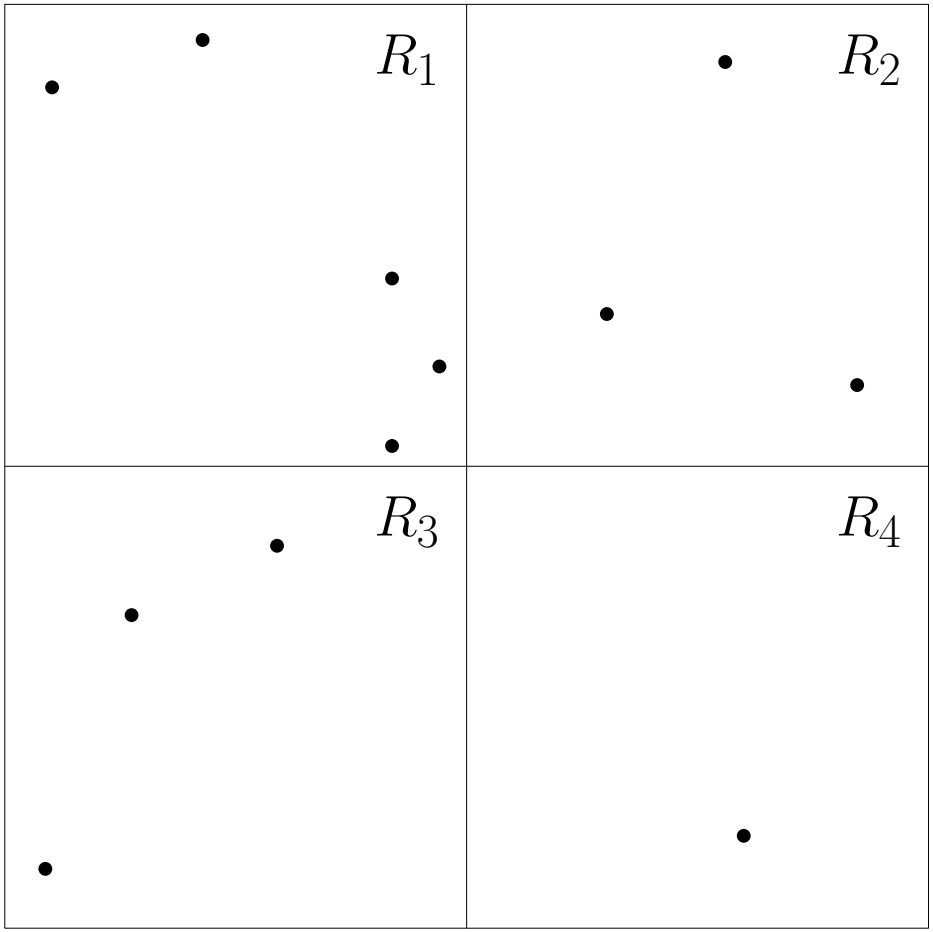


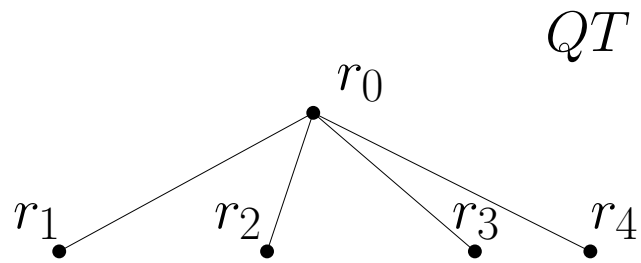
R_0

QT

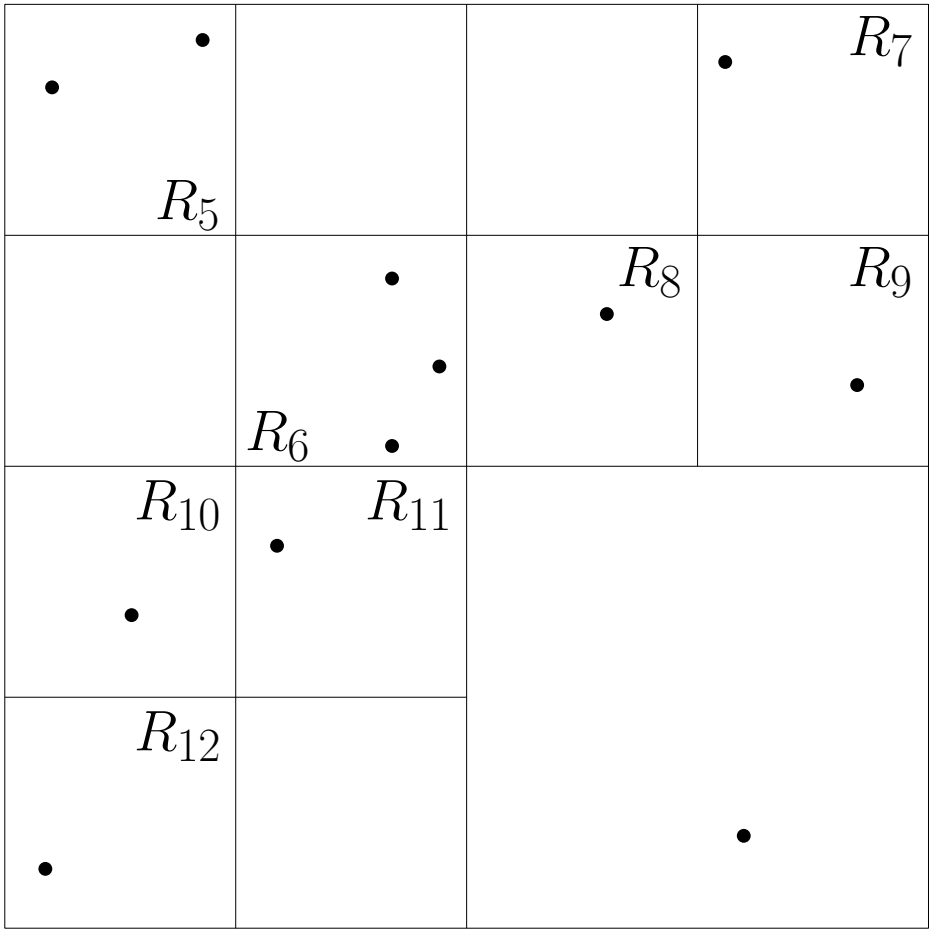
r_0

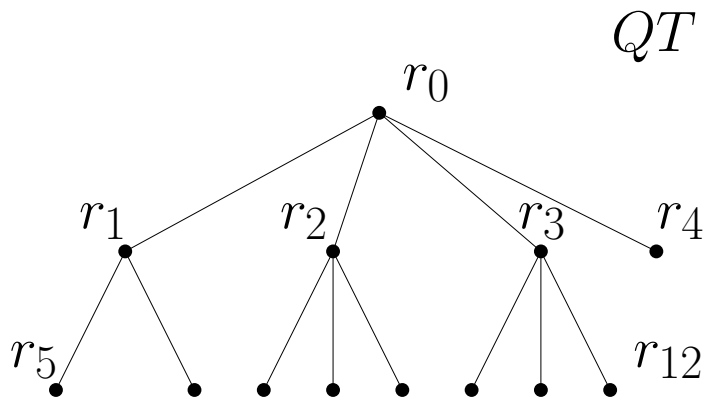
Quad-Tree



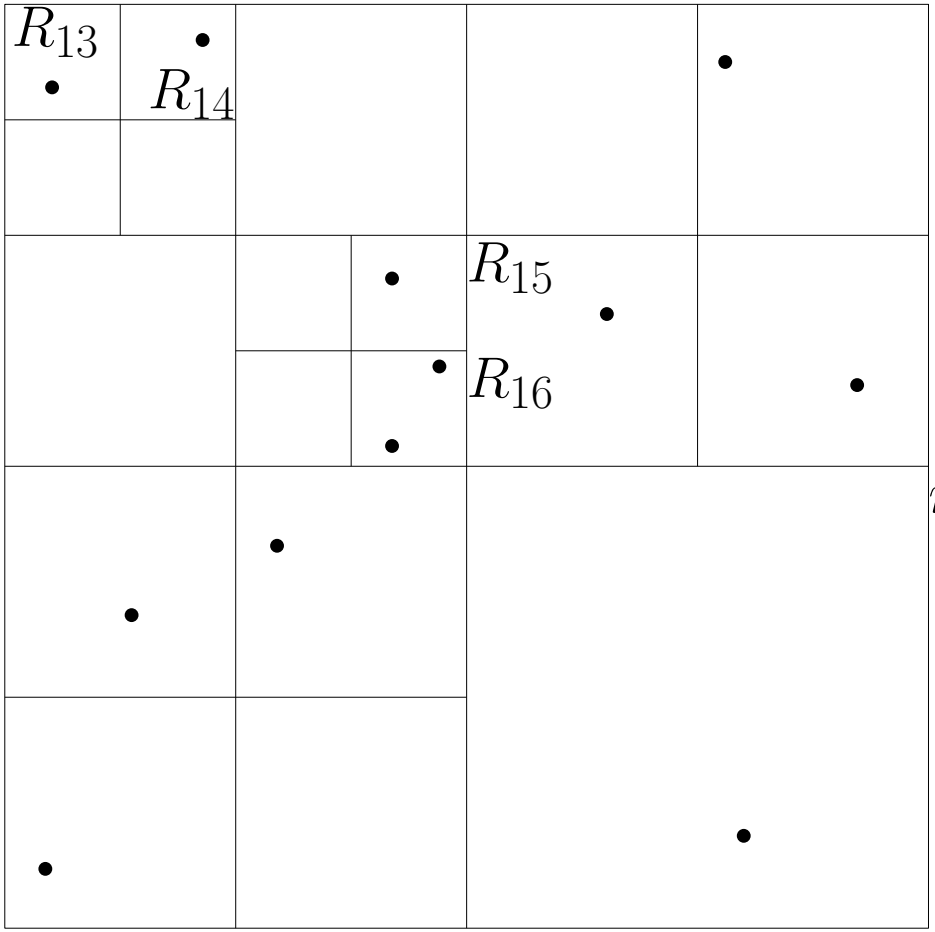


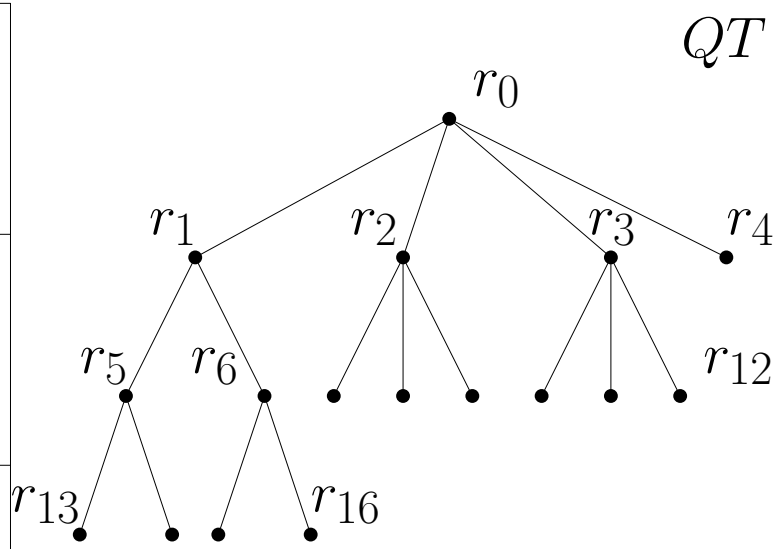
Quad-Tree



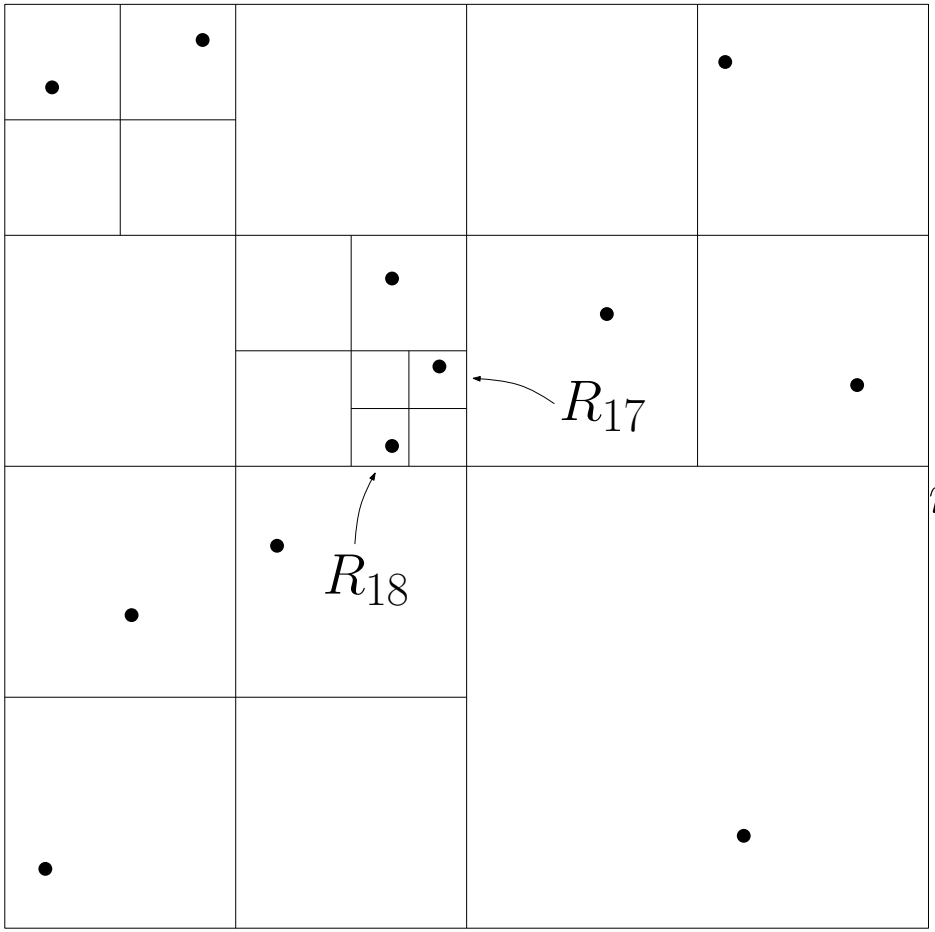


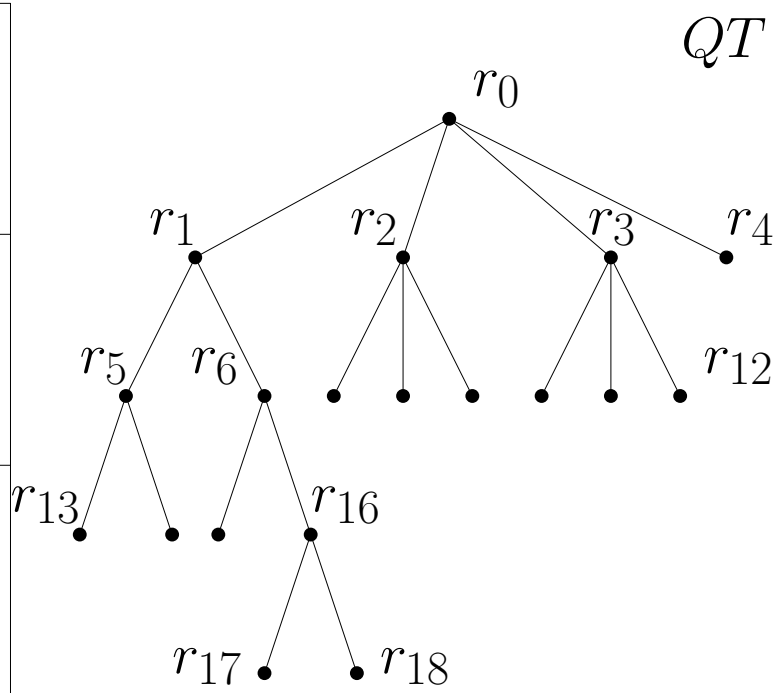
Quad-Tree





Quad-Tree



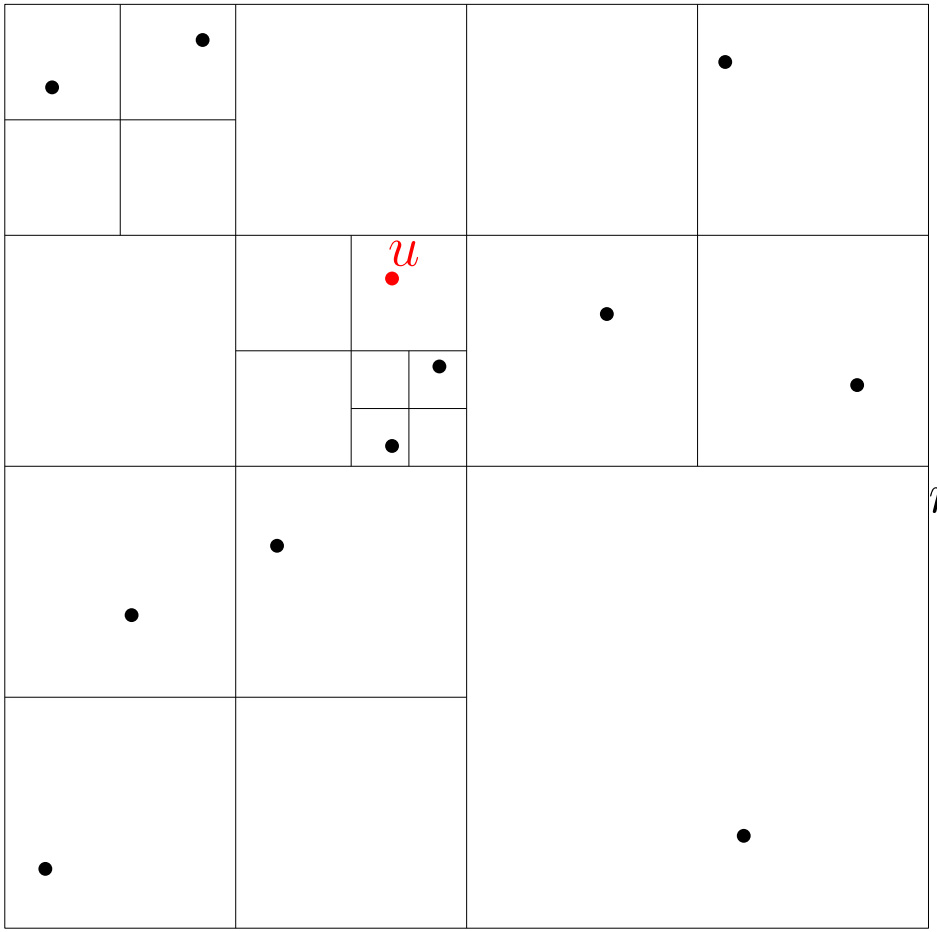


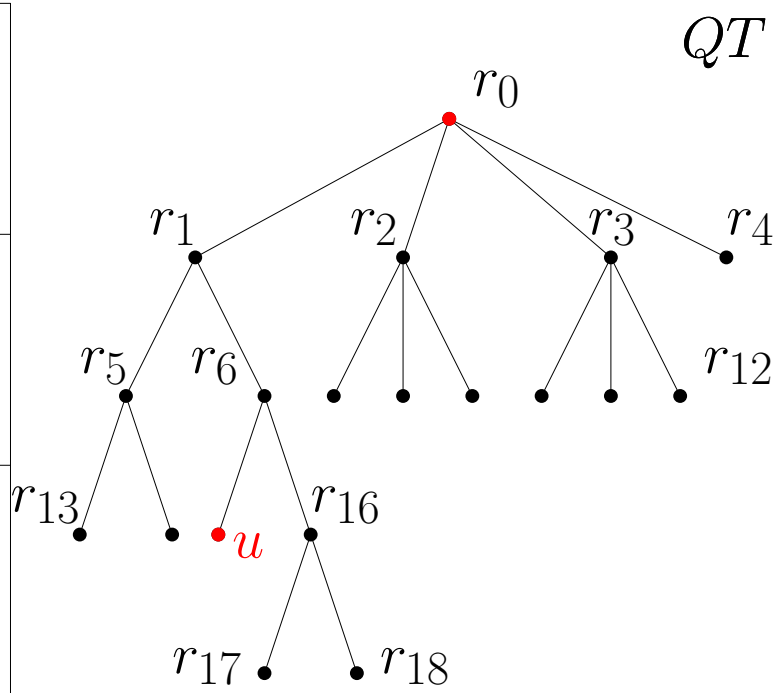
Quad-Tree

Eigenschaften

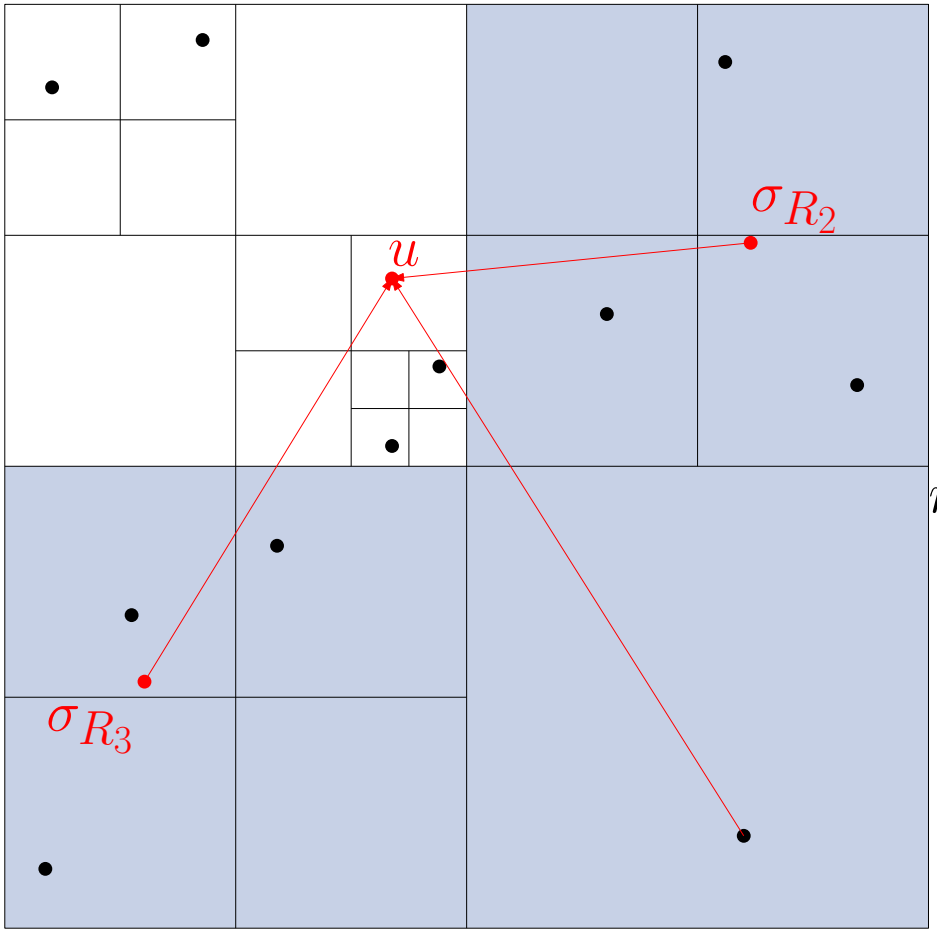
- Höhe $h \leq \log \frac{s_{\text{init}}}{d_{\min}} + \frac{3}{2}$
- Zeit-/Speicherbedarf $\mathcal{O}(hn)$
- komprimierter Quadtree in $\mathcal{O}(n \log n)$ berechenbar
- $h \in \mathcal{O}(\log n)$ bei gleichmäßiger Verteilung der Knoten

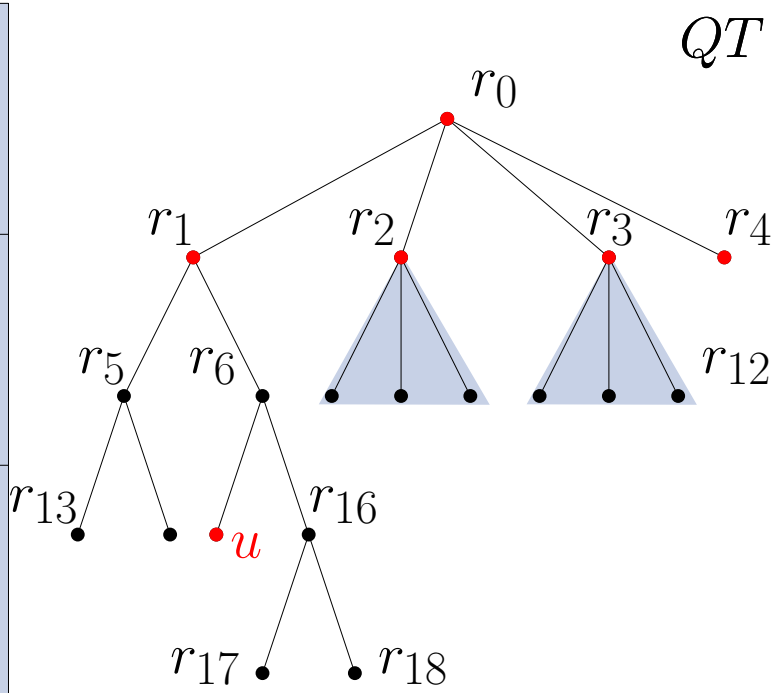
Quad-Tree



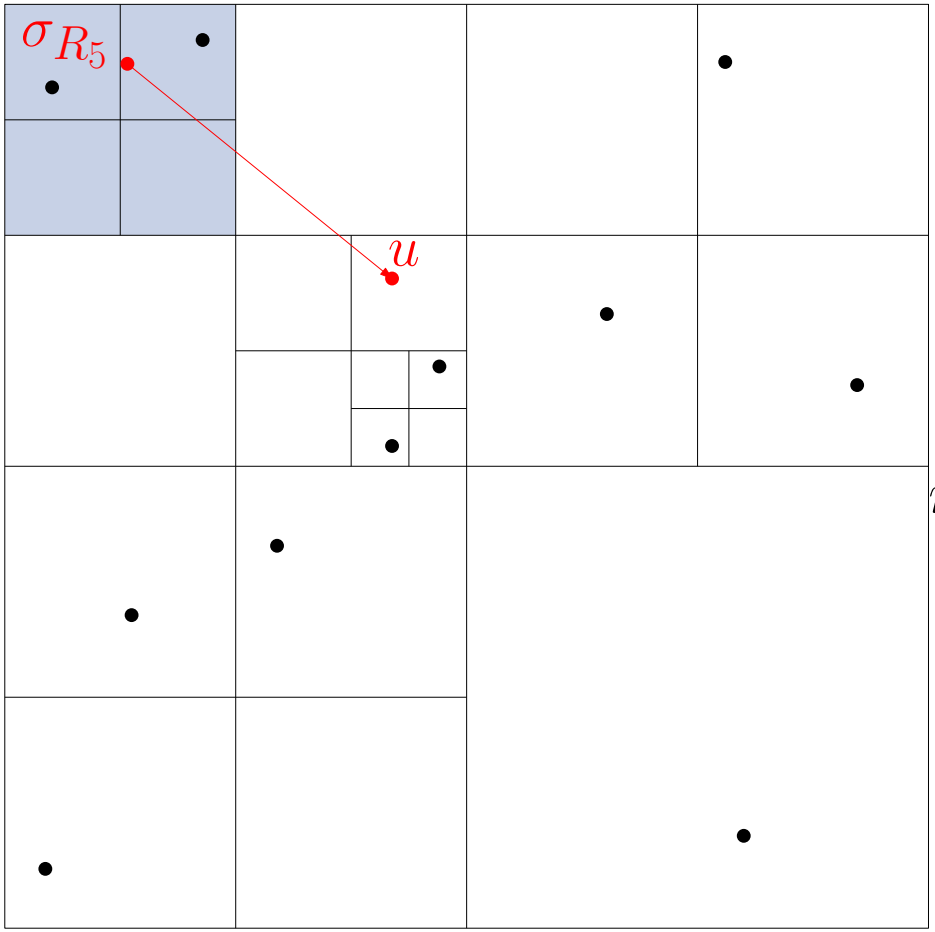


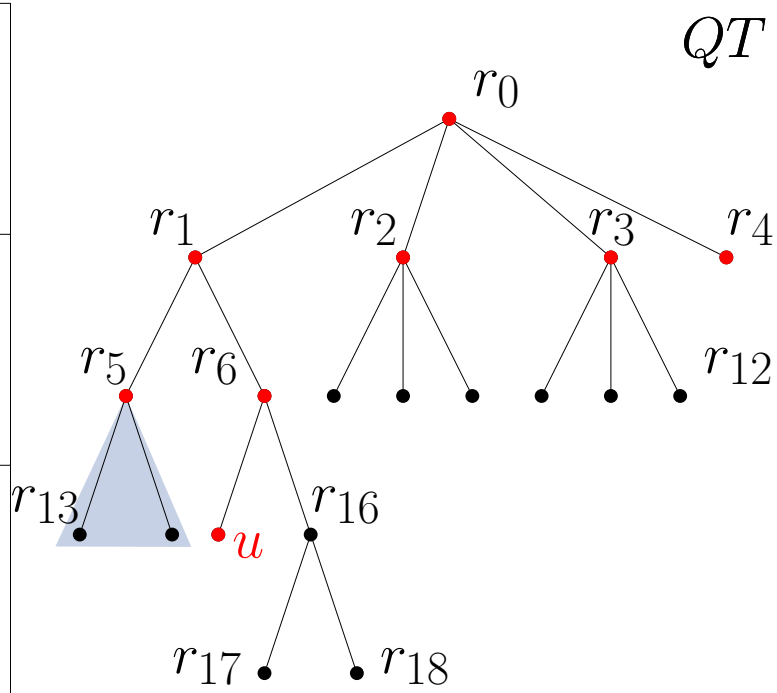
Quad-Tree



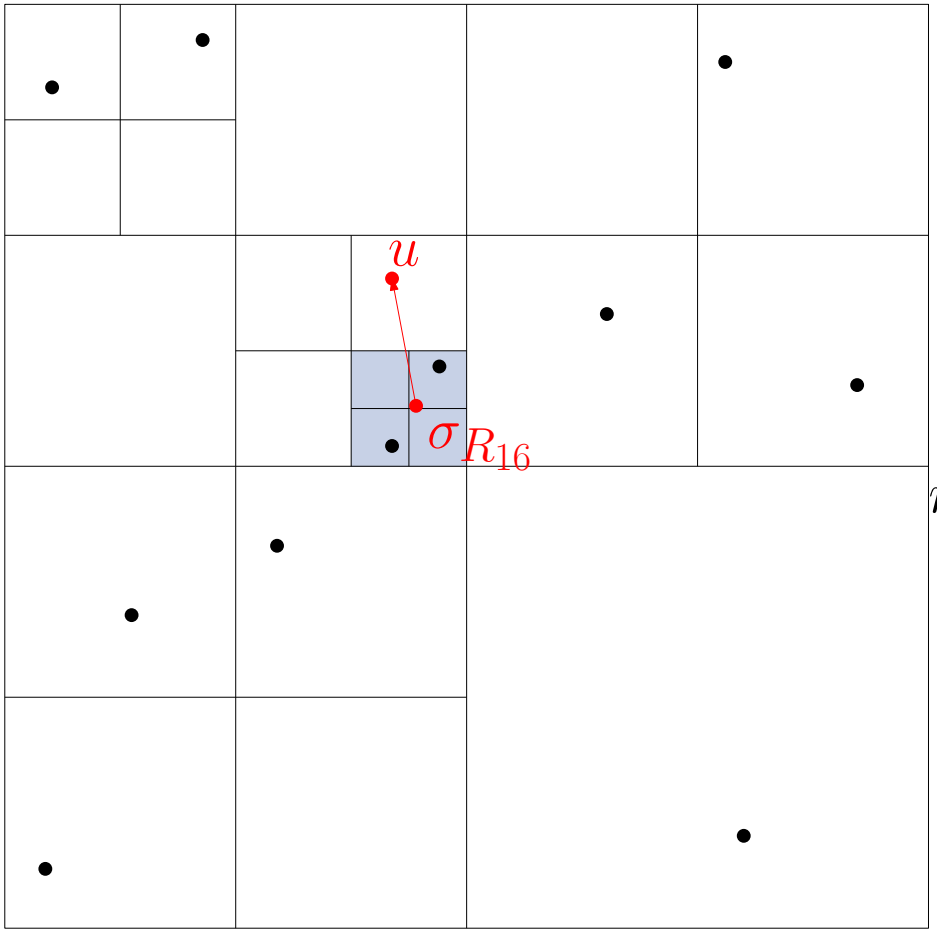


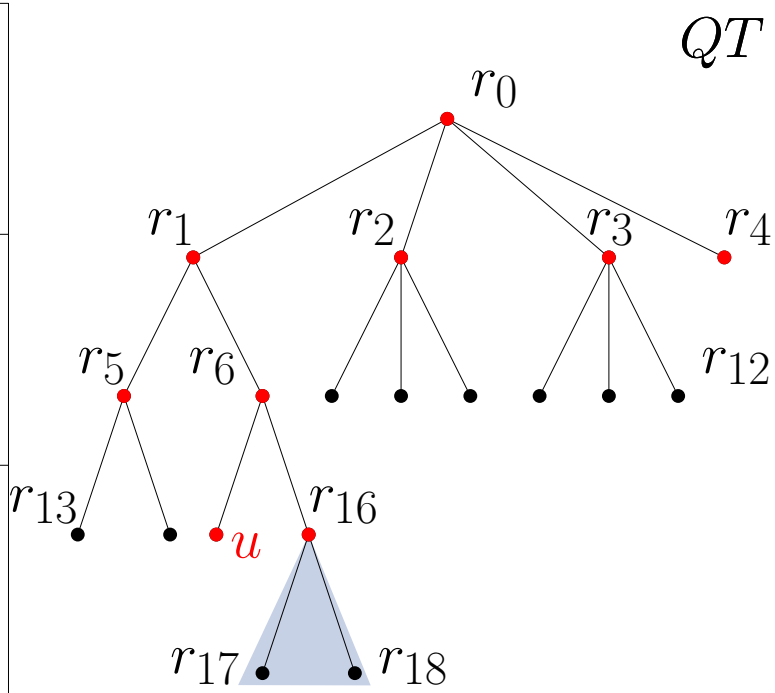
Quad-Tree





Quad-Tree





Variante GEM [Frick et al., ’95]

Modell

- abstoßende Kraft zwischen allen Knotenpaaren u und v

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

Variante GEM [Frick et al., ’95]

Modell

- abstoßende Kraft zwischen allen Knotenpaaren u und v

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

- anziehende Kraft zwischen adjazenten Knoten u und v, $\Phi(v) = 1 + \deg(v)/2$

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

Variante GEM [Frick et al., '95]

Modell

- abstoßende Kraft zwischen allen Knotenpaaren u und v

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

- anziehende Kraft zwischen adjazenten Knoten u und v, $\Phi(v) = 1 + \deg(v)/2$

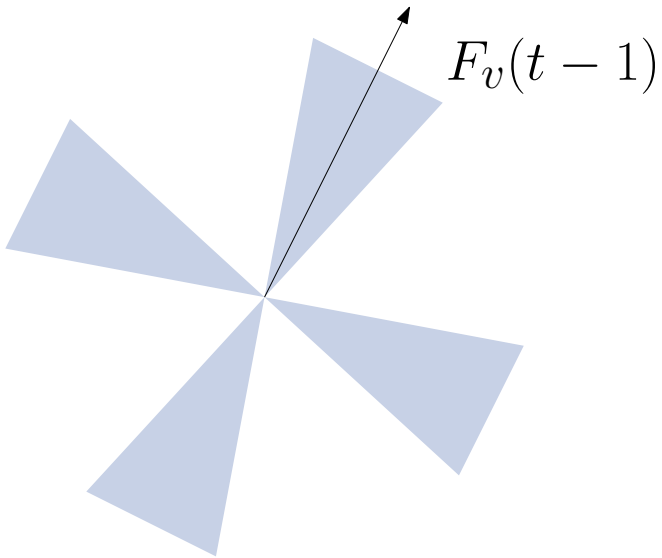
$$f_{\text{attr}}(p_u, p_v) = \frac{||p_u - p_v||^2}{\ell^2 \cdot \Phi(v)} \cdot \overrightarrow{p_v p_u}$$

- Gravitationskraft zum Baryzenter $p_{\text{bary}} = \sum_{v \in V} p_v / |V|$

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

30% Speedup laut Autoren

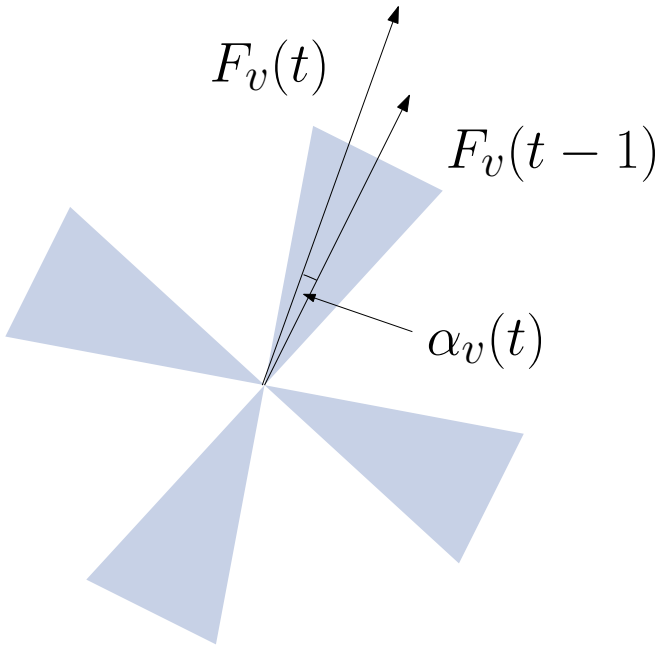
Adaptive Verschiebung



Modell Fortsetzung

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

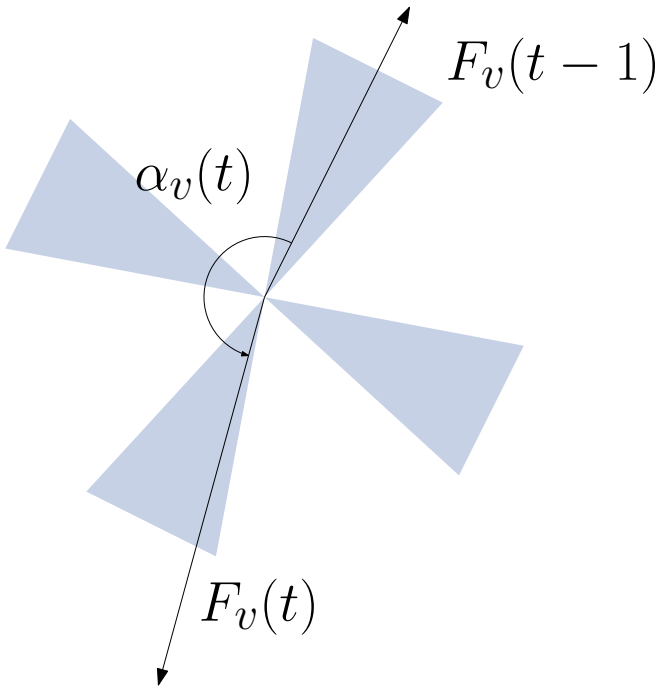
Adaptive Verschiebung



Modell Fortsetzung

- speichere alten Verschiebungs-vektor $F_v(t-1)$
- lokales Cooling
  - $\cos(\alpha_v(t)) \approx 1$ gleiche Richtung
    $\rightarrow$ Temperatur erhöhen

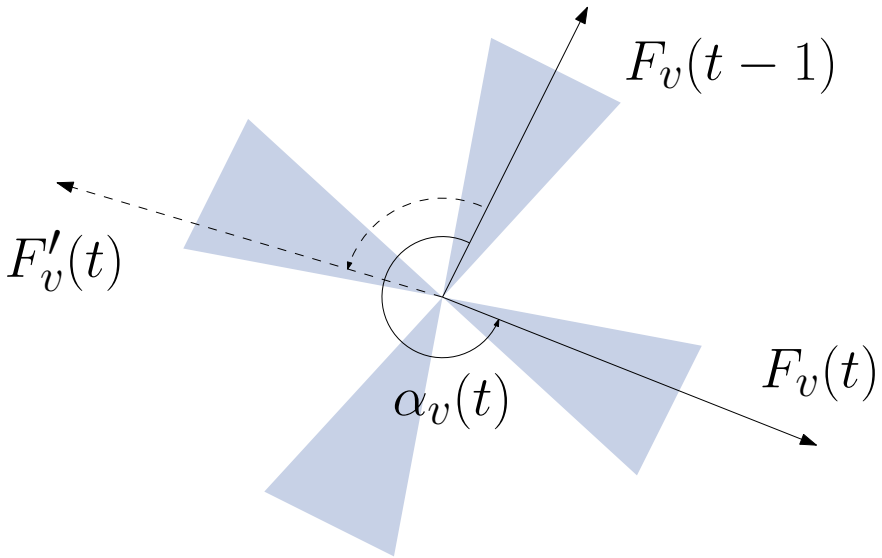
Adaptive Verschiebung



Modell Fortsetzung

- speichere alten Verschiebungs-vektor $F_v(t-1)$
- lokales Cooling
  - $\cos(\alpha_v(t)) \approx 1$ gleiche Richtung
    $\rightarrow$ Temperatur erhöhen
  - $\cos(\alpha_v(t)) \approx -1$ Oszillation
    $\rightarrow$ Temperatur verringern

Adaptive Verschiebung



Modell Fortsetzung

- speichere alten Verschiebungs-vektor $F_v(t-1)$
- lokales Cooling
  - $\cos(\alpha_v(t)) \approx 1$ gleiche Richtung
    $\rightarrow$ Temperatur erhöhen
  - $\cos(\alpha_v(t)) \approx -1$ Oszillation
    $\rightarrow$ Temperatur verringern
  - $\cos(\alpha_v(t)) \approx 0$ Rotation
    $\rightarrow$ Rotationszähler updaten, Temperatur verringern

GEM

Weitere Techniken

- Integer-Arithmetik
- Randomisierung
  - modifiziere Verschiebungsvektor zufällig
  - Entkommen aus lokalen Minima

GRIP – Graph dRawing with Intelligent Placement [Gajer et al., ’00]

Motivation

- Springembedder für große Graphen ($\gg$ 1000 Knoten) viel zu langsam
- anfälliger für lokale Minima / Initiallayout spielt größere Rolle

Techniken

- Top-Down Vergröberung (Filtration)
- Bottom-Up Berechnung
- iterative Wahl der Initialposition

GRIP Algorithmus

Algorithmus 2: GRIP

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

1 $\mathcal{V} \leftarrow$ Filtration $V_0 \supset V_1 \supset \cdots \supset V_k \supset \emptyset$

GRIP Algorithmus

Algorithmus 2: GRIP

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

1 $\mathcal{V} \leftarrow$ Filtration $V_0 \supset V_1 \supset \cdots \supset V_k \supset \emptyset$
2 **for** $i = k$ **to** 0 **do**
3 **foreach** $v \in V_i \setminus V_{i+1}$ **do**
4 berechne Nachbarschaften von v
5 berechne initiale Position von v

GRIP Algorithmus

Algorithmus 2: GRIP

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

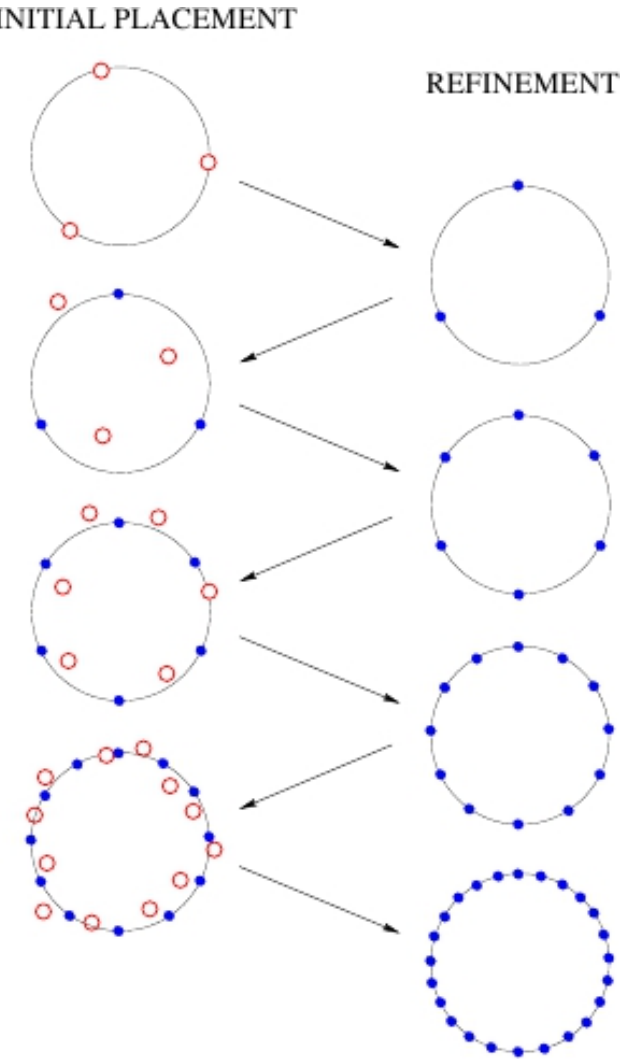
```
1 V ← Filtration V_0 ⊃ V_1 ⊃ ··· ⊃ V_k ⊃ ∅
2 for i = k to 0 do
3     foreach v ∈ V_i \ V_{i+1} do
4         berechne Nachbarschaften von v
5         berechne initiale Position von v
6     for j = 1 to rounds do
7         foreach v ∈ V_i do
8             kräftebasierte Relaxierung
```

GRIP Algorithmus

Algorithmus 2: GRIP

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

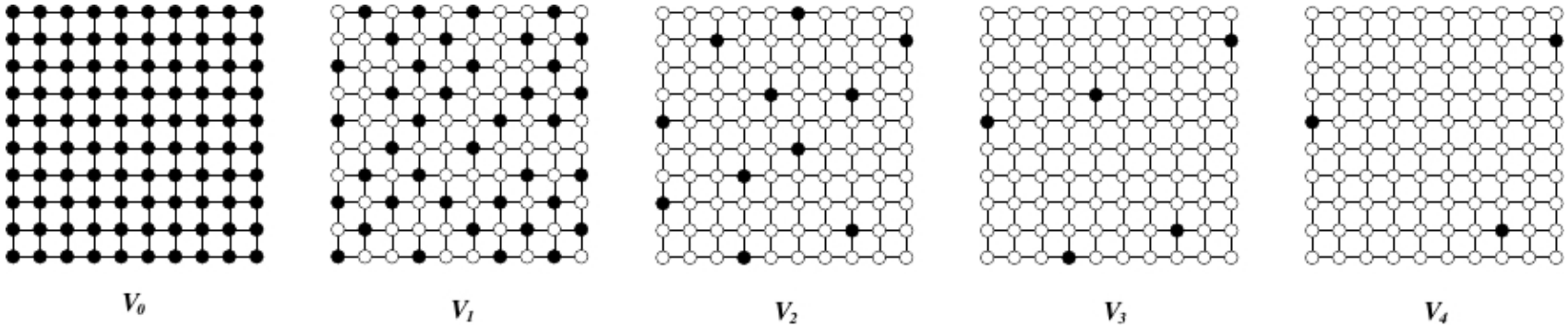
```
1 V ← Filtration V_0 ⊃ V_1 ⊃ ··· ⊃ V_k ⊃ ∅
2 for i = k to 0 do
3     foreach v ∈ V_i \ V_{i+1} do
4         berechne Nachbarschaften von v
5         berechne initiale Position von v
6     for j = 1 to rounds do
7         foreach v ∈ V_i do
8             kräftebasierte Relaxierung
```



MIS-Filtration

Maximum Independent Set (MIS)-Filtration

- Sequenz $V = V_0 \supset V_1 \supset \cdots \supset V_k \supset \emptyset$
- V_i is (inklusions-)maximale Knotenmenge
- Abstand zwischen Knoten in V_i is mindestns $2^{i-1} + 1$
- gute Balance zwischen Tiefe und Größe der “Layer”



MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen

MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen
- sei $V^* = V_i$

MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen
- sei $V^* = V_i$
- wähle iterativ zufälliges Element v in V^*

MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen
- sei $V^* = V_i$
- wähle iterativ zufälliges Element v in V^*
- entferne alle Elemente in V^* mit Distanz $< 2^i$

MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen
- sei $V^* = V_i$
- wähle iterativ zufälliges Element v in V^*
- entferne alle Elemente in V^* mit Distanz $< 2^i$
- beschränkte Breitensuche von v, Aufwand linear

MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen
- sei $V^* = V_i$
- wähle iterativ zufälliges Element v in V^*
- entferne alle Elemente in V^* mit Distanz $< 2^i$
- beschränkte Breitensuche von v, Aufwand linear

MIS-Filtration

Berechnung einer MIS Filtration

- gegeben V_i, wollen V_{i+1} berechnen
- sei $V^* = V_i$
- wähle iterativ zufälliges Element v in V^*
- entferne alle Elemente in V^* mit Distanz $< 2^i$
- beschränkte Breitensuche von v, Aufwand linear

Tiefe

- letztes Level $k \Rightarrow 2^k > \delta(G)$ (Durchmesser)
- daher Tiefe in $\mathcal{O}(\log \delta(G))$

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$
- Tradeoff: wenig Nachbarn $\rightarrow$ schlechte Qualität, viele Nachbarn $\rightarrow$ zeitaufwendig

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$
- Tradeoff: wenig Nachbarn $\rightarrow$ schlechte Qualität, viele Nachbarn $\rightarrow$ zeitaufwendig
- wähle Schranke $\text{nbrs}(i) = \deg_{\varnothing}(G)n/|V_i|$

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$
- Tradeoff: wenig Nachbarn $\rightarrow$ schlechte Qualität, viele Nachbarn $\rightarrow$ zeitaufwendig
- wähle Schranke $\text{nbrs}(i) = \deg_{\varnothing}(G)n/|V_i|$
- wähle $\text{nbrs}(i)$ viele Nachbarn mit Breitensuche von v

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$
- Tradeoff: wenig Nachbarn $\rightarrow$ schlechte Qualität, viele Nachbarn $\rightarrow$ zeitaufwendig
- wähle Schranke $\text{nbrs}(i) = \deg_{\varnothing}(G)n/|V_i|$
- wähle $\text{nbrs}(i)$ viele Nachbarn mit Breitensuche von v
- Tiefe eines Knotens v in Filtration: größtes d mit $v \in V_d$

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$
- Tradeoff: wenig Nachbarn $\rightarrow$ schlechte Qualität, viele Nachbarn $\rightarrow$ zeitaufwendig
- wähle Schranke $\text{nbrs}(i) = \deg_{\varnothing}(G)n/|V_i|$
- wähle $\text{nbrs}(i)$ viele Nachbarn mit Breitensuche von v
- Tiefe eines Knotens v in Filtration: größtes d mit $v \in V_d$
- neuer Knoten mit Tiefe d wird in alle $N_j(v)$ mit $j \leq d$ eingefügt, die weniger als $\text{nbrs}(j)$ viele Nachbarn haben (Distanz zu v bekannt!)

MIS-Filtration

Berechnung von Nachbarschaften

- für $v \in V_i$ wähle Nachbarschaften $N_j(v)$ für alle $j \leq i$
- Tradeoff: wenig Nachbarn $\rightarrow$ schlechte Qualität, viele Nachbarn $\rightarrow$ zeitaufwendig
- wähle Schranke $\text{nbrs}(i) = \deg_{\varnothing}(G)n/|V_i|$
- wähle $\text{nbrs}(i)$ viele Nachbarn mit Breitensuche von v
- Tiefe eines Knotens v in Filtration: größtes d mit $v \in V_d$
- neuer Knoten mit Tiefe d wird in alle $N_j(v)$ mit $j \leq d$ eingefügt, die weniger als $\text{nbrs}(j)$ viele Nachbarn haben (Distanz zu v bekannt!)
- Zeit-/Speicheraufwand $\mathcal{O}(k^2 \deg_{\varnothing}(G) \cdot n)$, $k \in \mathcal{O}(\log n)$?

MIS-Filtration

Initiale Platzierung

- starte mit drei Knoten u, v, w, deren Abstände bekannt sind

MIS-Filtration

Initiale Platzierung

- starte mit drei Knoten u, v, w, deren Abstände bekannt sind
- Löse Gleichungssystem

MIS-Filtration

Initiale Platzierung

- starte mit drei Knoten u, v, w, deren Abstände bekannt sind
- Löse Gleichungssystem
- für Knoten $v \in V_i \setminus V_{i+1}$ wähle paare von Nachbarn, berechne mögliche Positionen aus Graphdistanz und plaziere Knoten in Baryzenter der nächsten Lösungen (siehe Tafel)

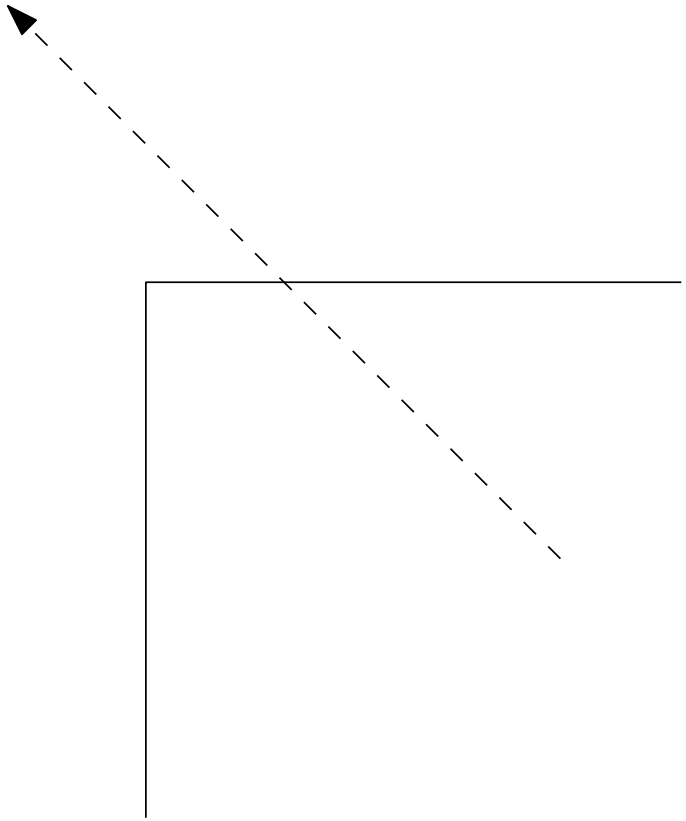
MIS-Filtration

Initiale Platzierung

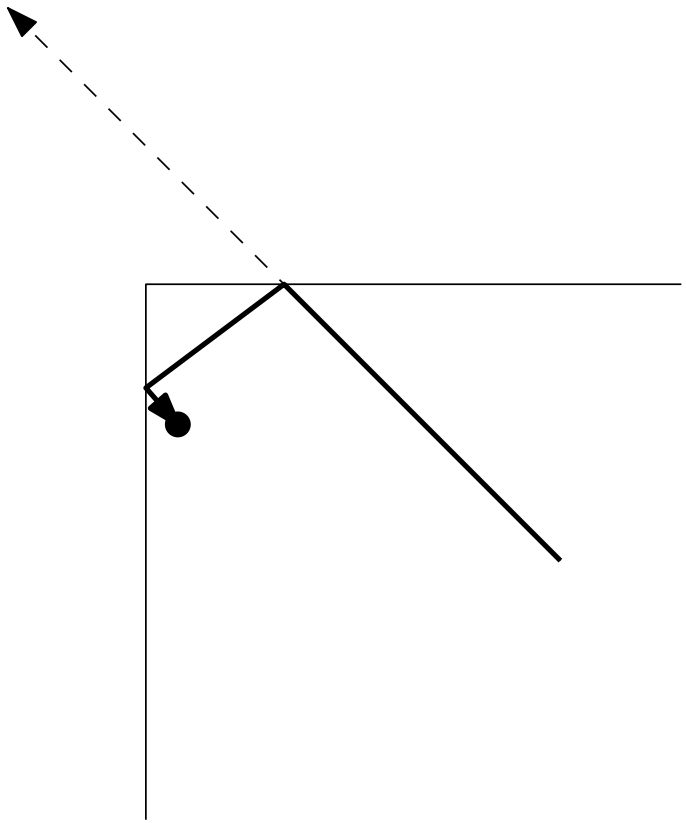
- starte mit drei Knoten u, v, w, deren Abstände bekannt sind
- Löse Gleichungssystem
- für Knoten $v \in V_i \setminus V_{i+1}$ wähle paare von Nachbarn, berechne mögliche Positionen aus Graphdistanz und plaziere Knoten in Baryzenter der nächsten Lösungen (siehe Tafel)
- lokale kräftebasierte Relaxierung

Demo

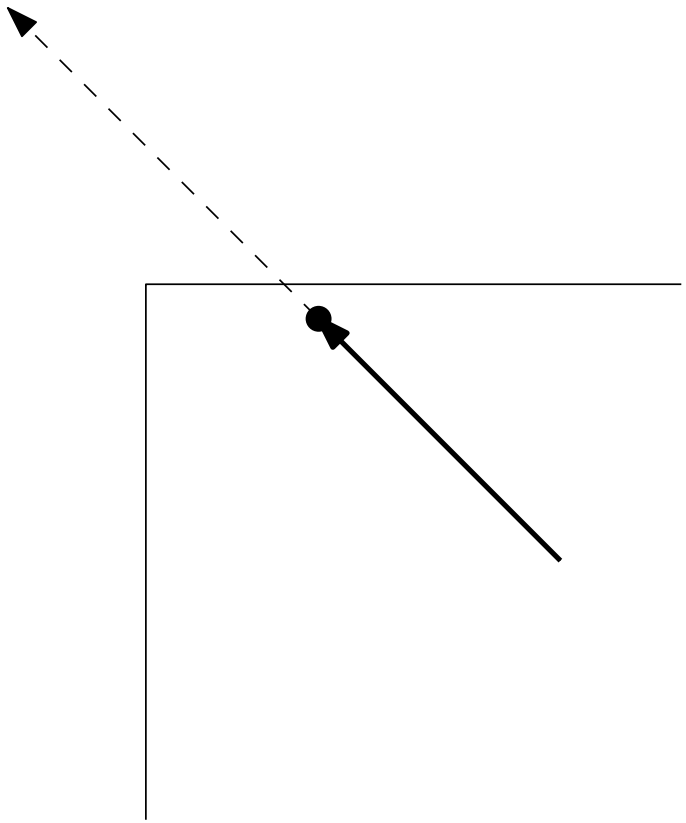
Modellierung – Rahmen



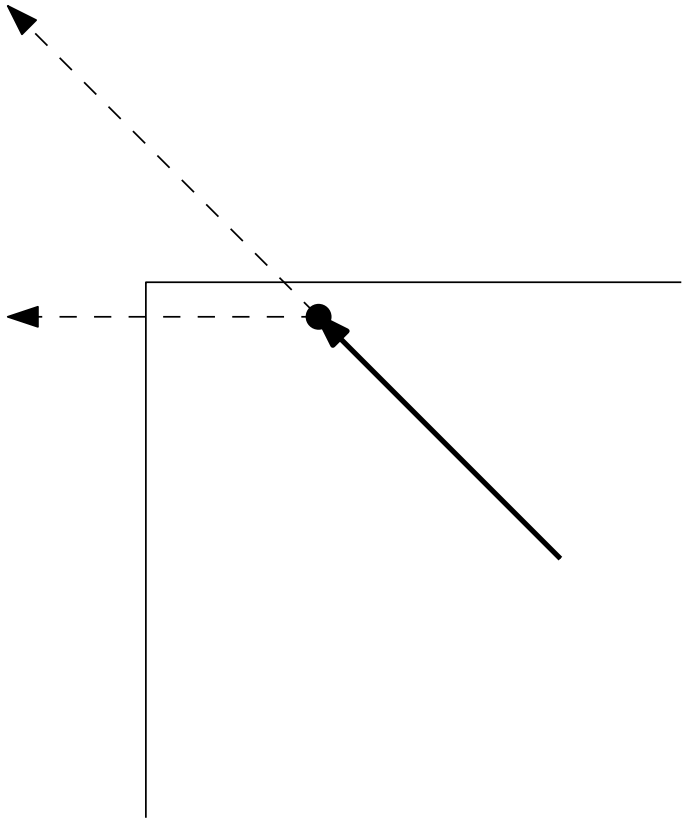
Modellierung – Rahmen



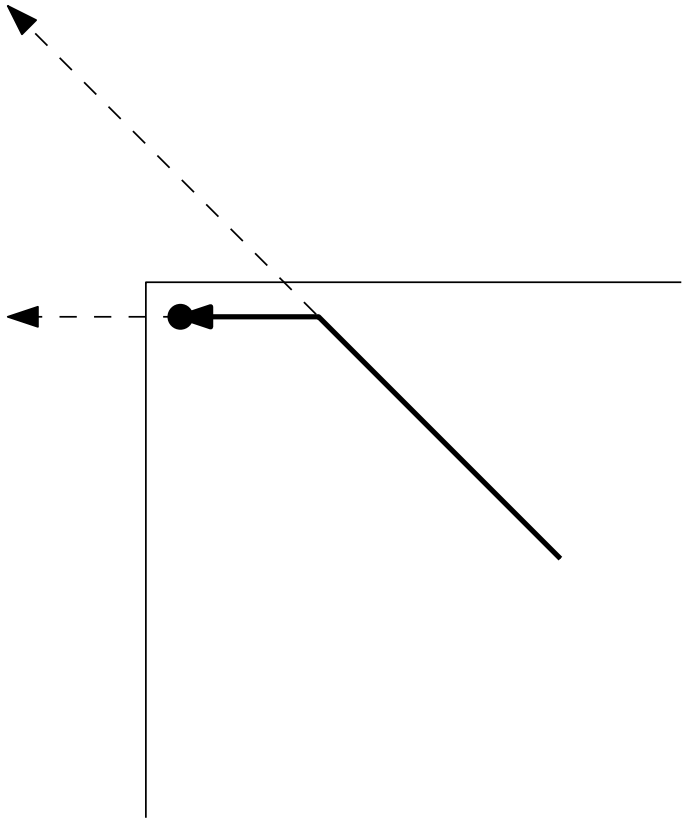
Modellierung – Rahmen



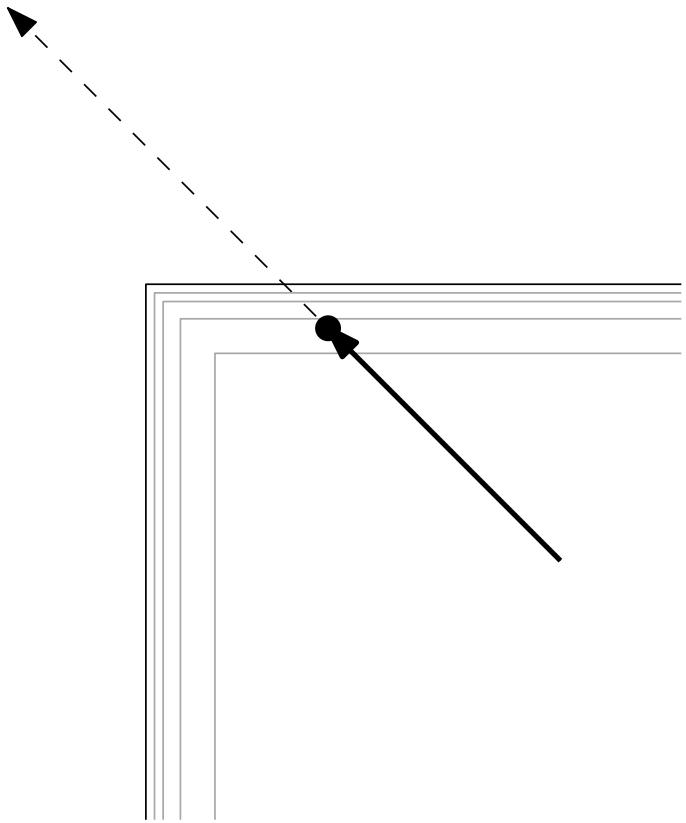
Modellierung – Rahmen



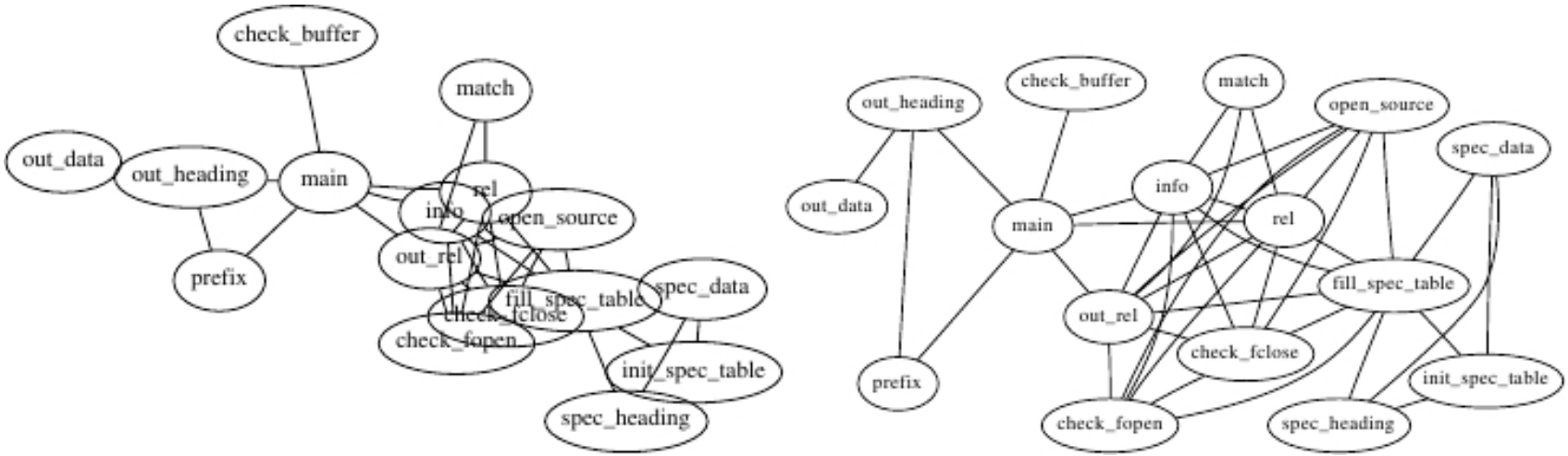
Modellierung – Rahmen



Modellierung – Rahmen

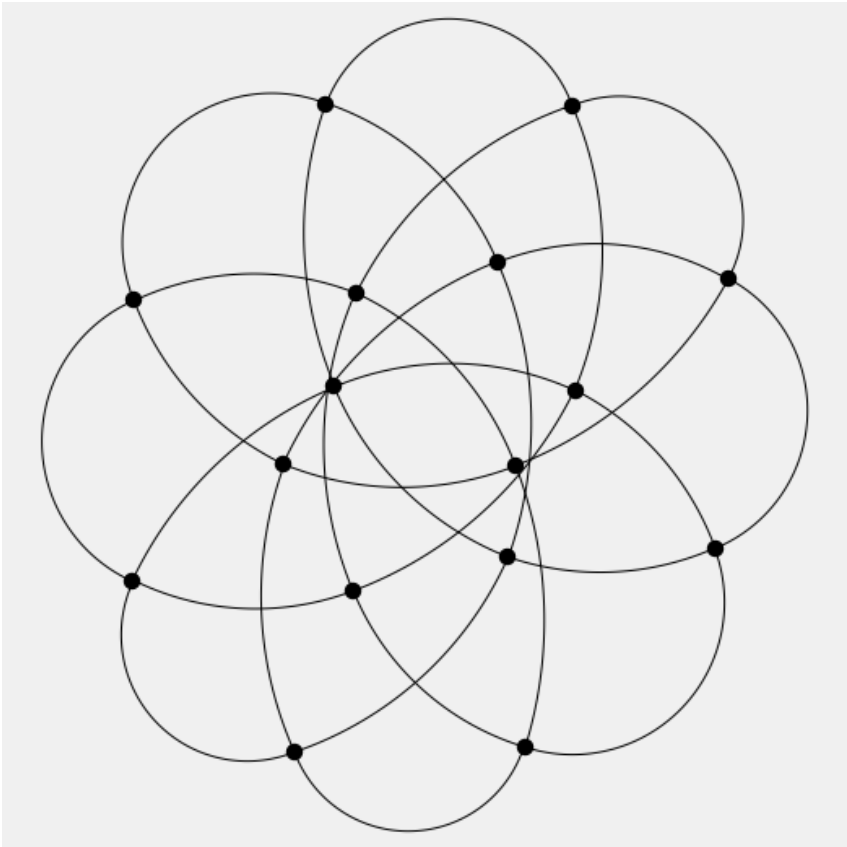


Modellierung – realistischere Knoten



[Gansner & North, ’98]

Modellierung – Nichtgeradlinige Kanten



[http://lombardi.cs.arizona.edu]

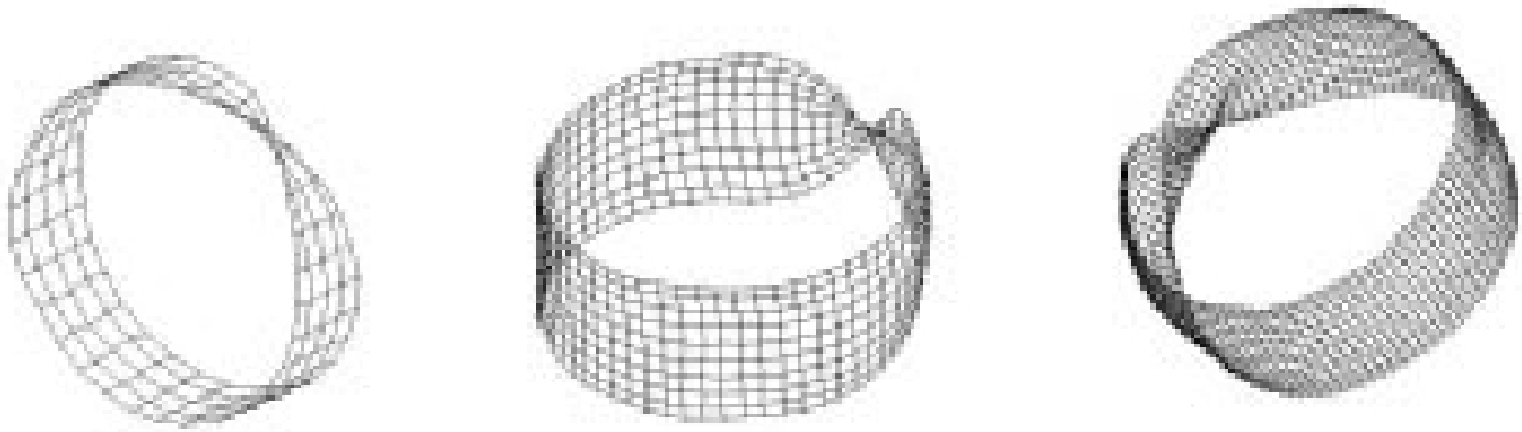


Fig. 5. Moebius strips on 150, 300, and 1500 vertices drawn in directly 3D. Note the rough "twists."

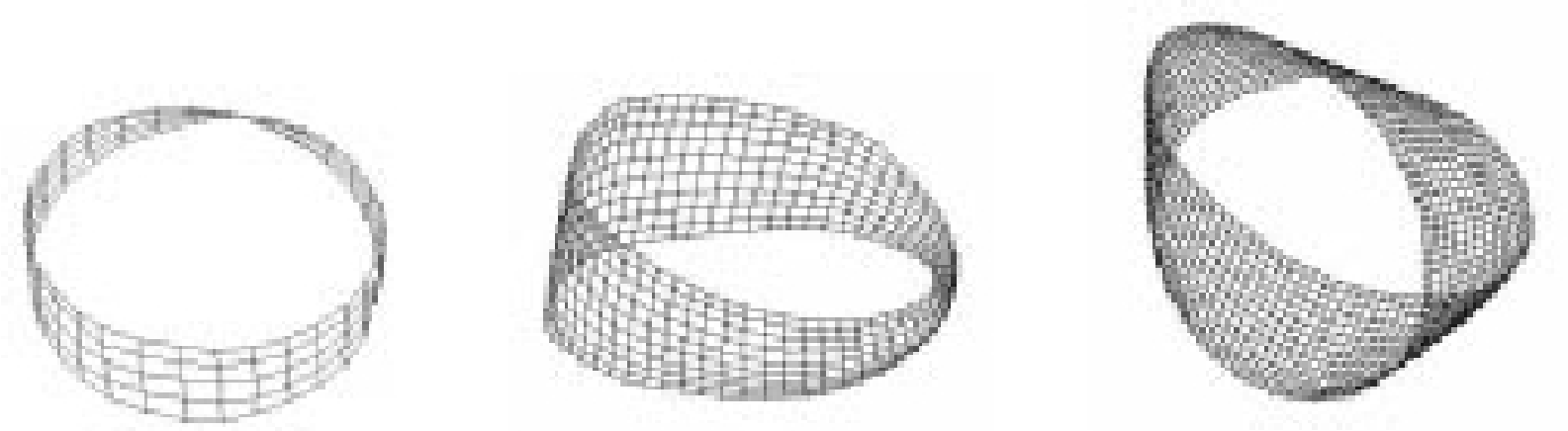
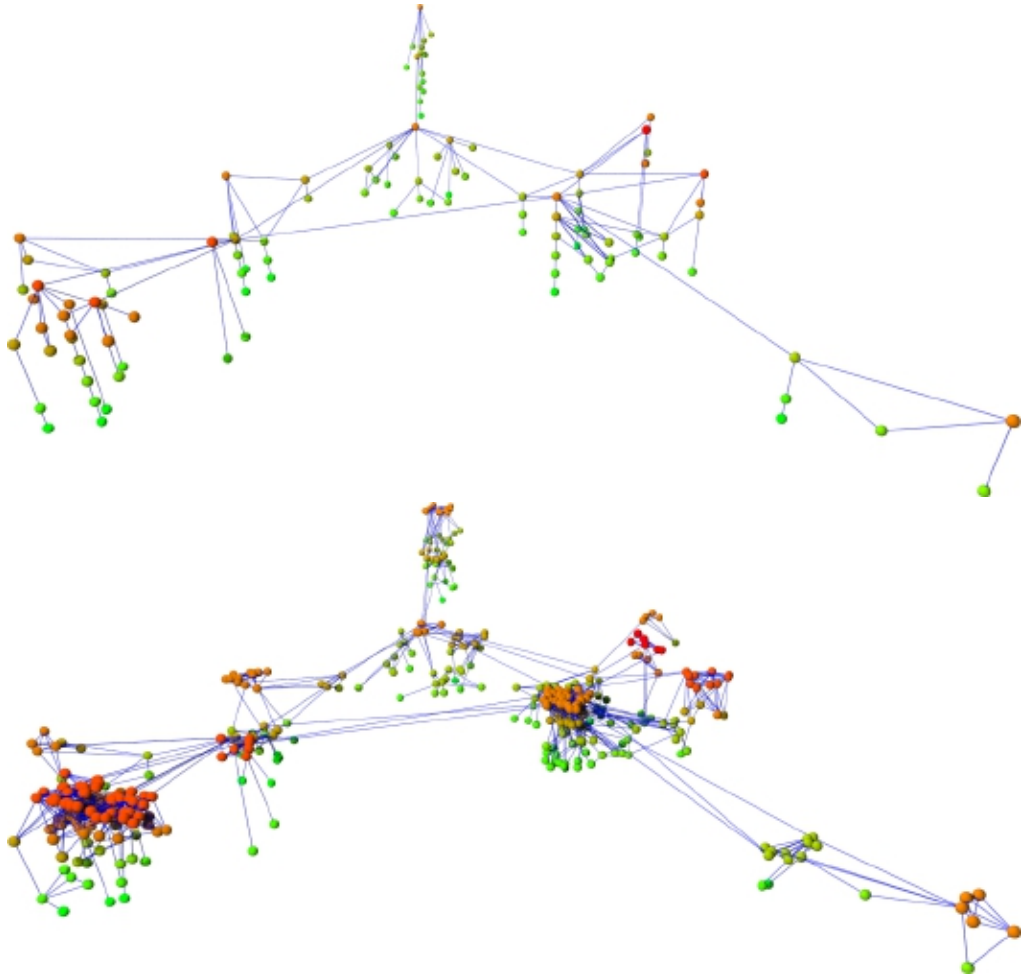
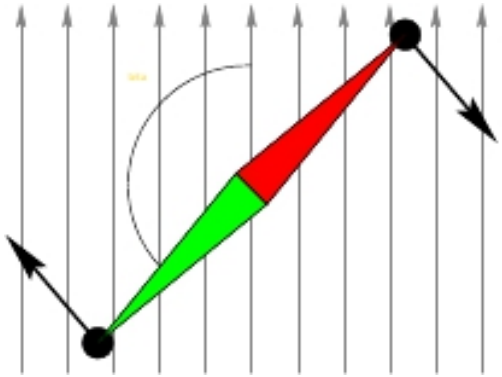


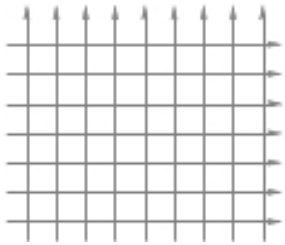
Fig. 6. The same Moebius strips as in Fig. 5 but drawn in 4D and projected in 3D. Note the smooth twists.

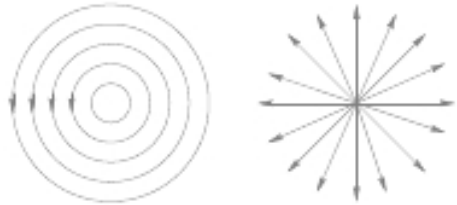
Clustered Graphs



Gerichtete Graphen

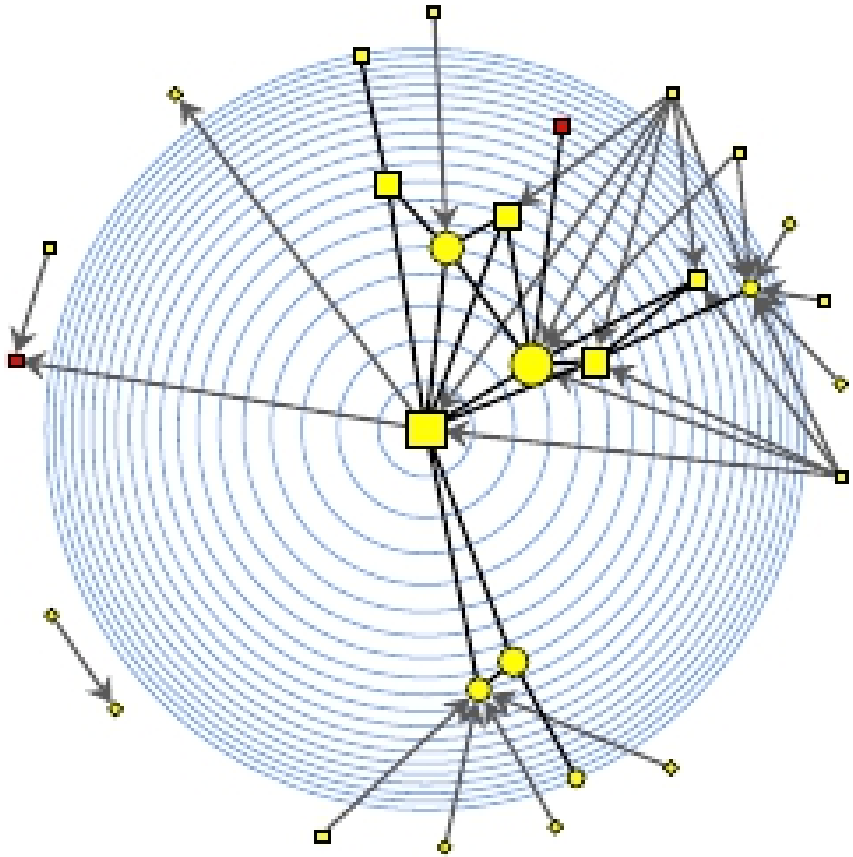






[Sugiyama & Misue, ’95]

Constraints

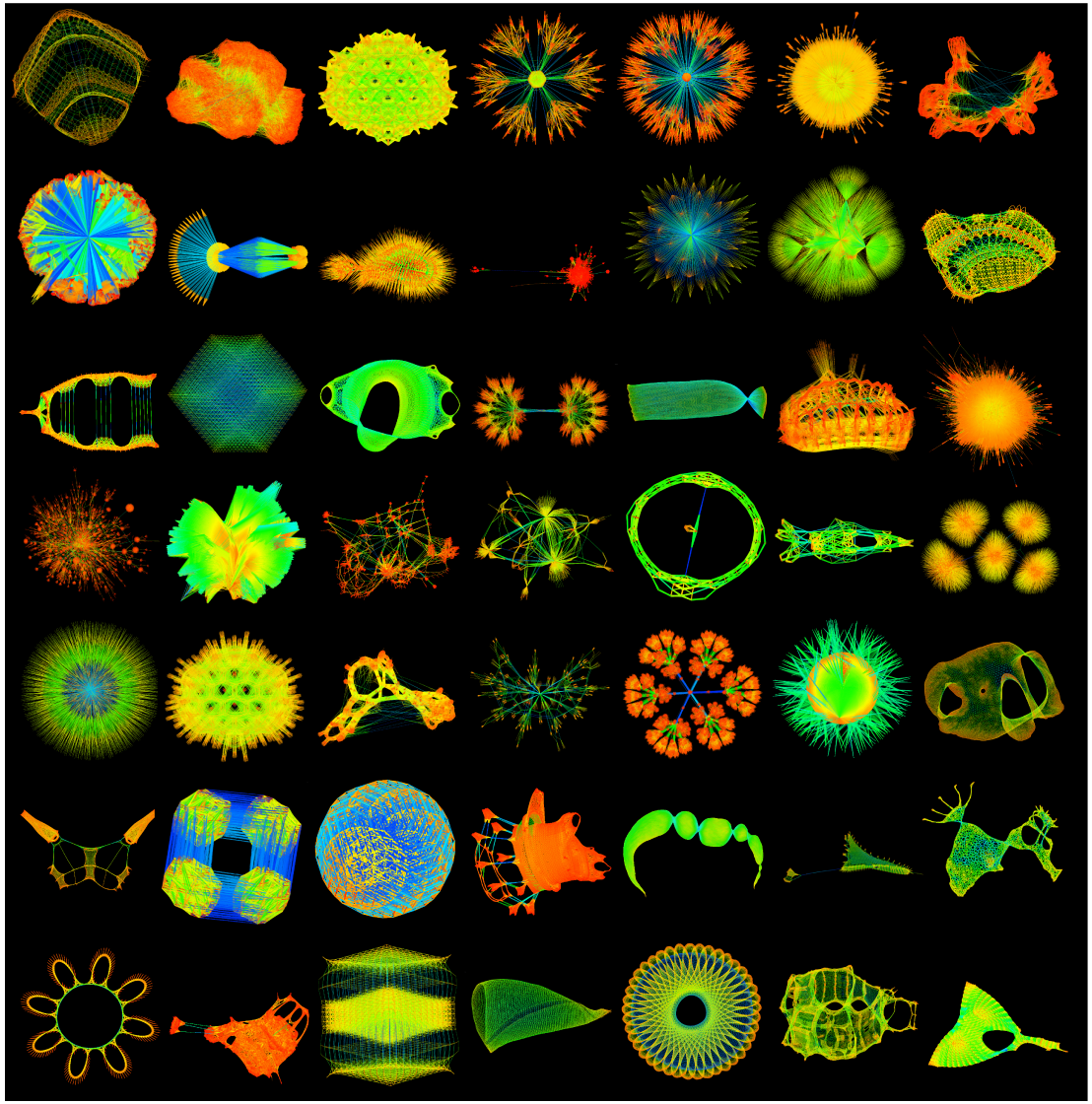


[Brandes, ’99]

Zusammenfassung

Was ist wichtig?

- Motivation / Ziele
- Kräftemodelle
- Beschleunigungsideen
- Algorithmen
- Einschätzung



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