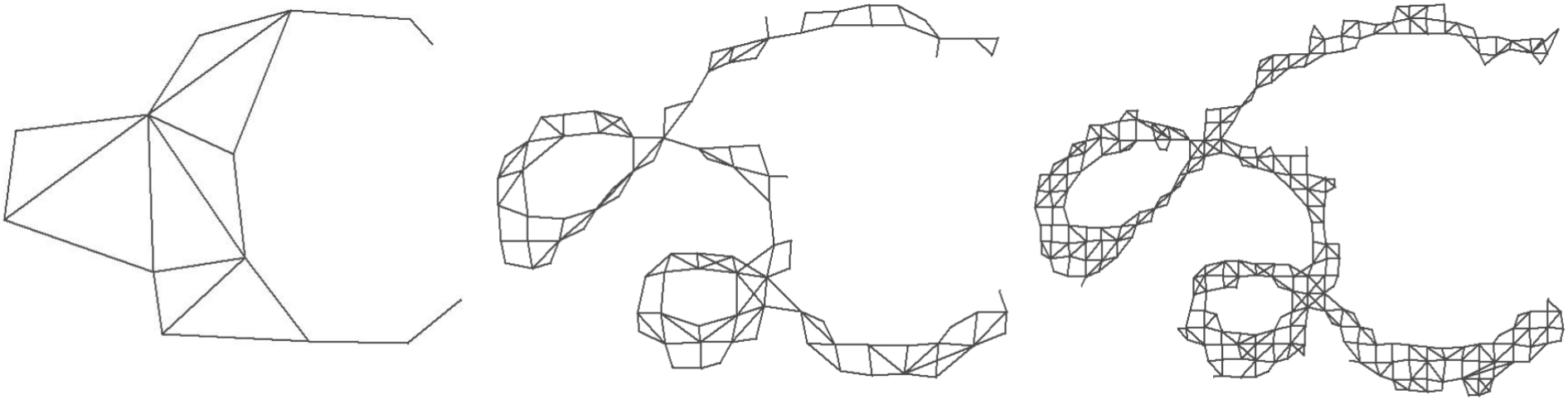
Karlsruher Institut für Technologie

FADE: Graph Drawning, Clustering and visual Abstraction

Seminar-Vortrag von Michael Vollmer

Institut für Theoretische Informatik

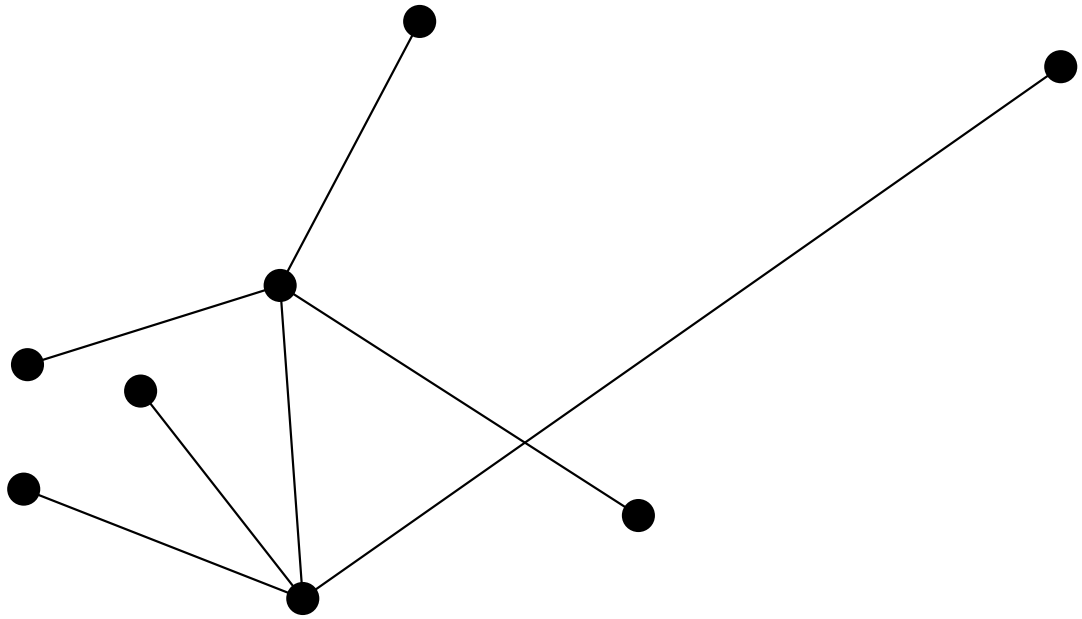


Inhalt

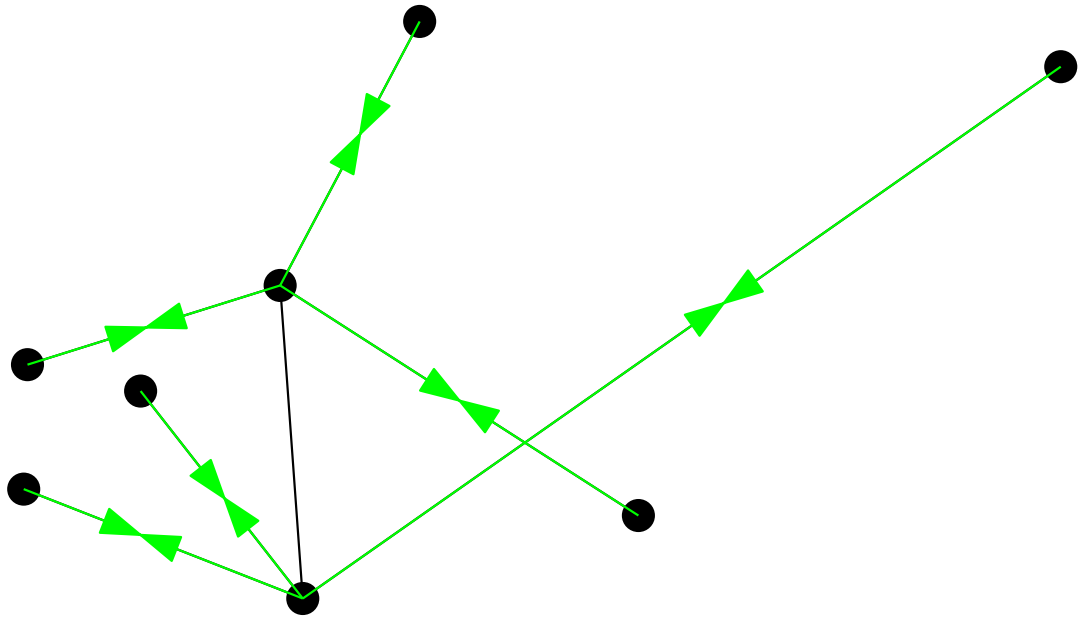
Force Directed Drawing

- Algorithmen zum zeichnen theoretischer Graphen (punktförmige Knoten)
- Physikalisches Modell
  - **Edge Forces** Knoten die verbunden sind ziehen sich an, als seien sie durch eine mechanische Feder verbunden
  - **Non-Edge Forces** Knoten stoßen sich gegenseitig ab, als wären sie gleich geladene Teilchen
- Ablauf:
  1. Beginne mit beliebiger Einbettung
  2. Repeat
     - 2.1 Berechne Kräfte
     - 2.2 Wende Kräfte an
  3. Until: Konvergenz

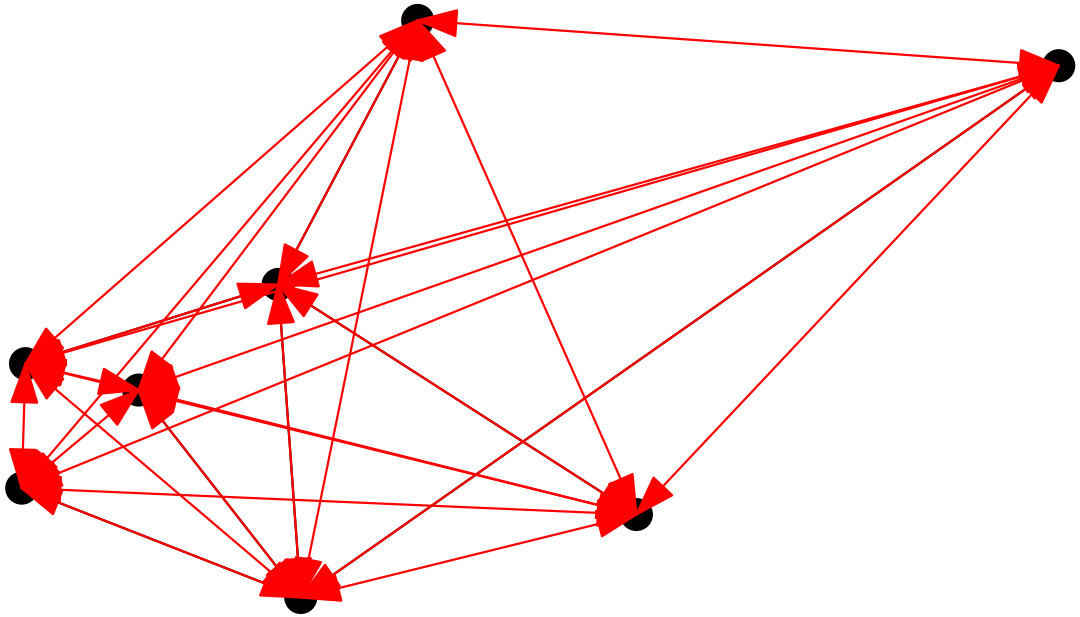
Force Directed Drawing - Beispiel



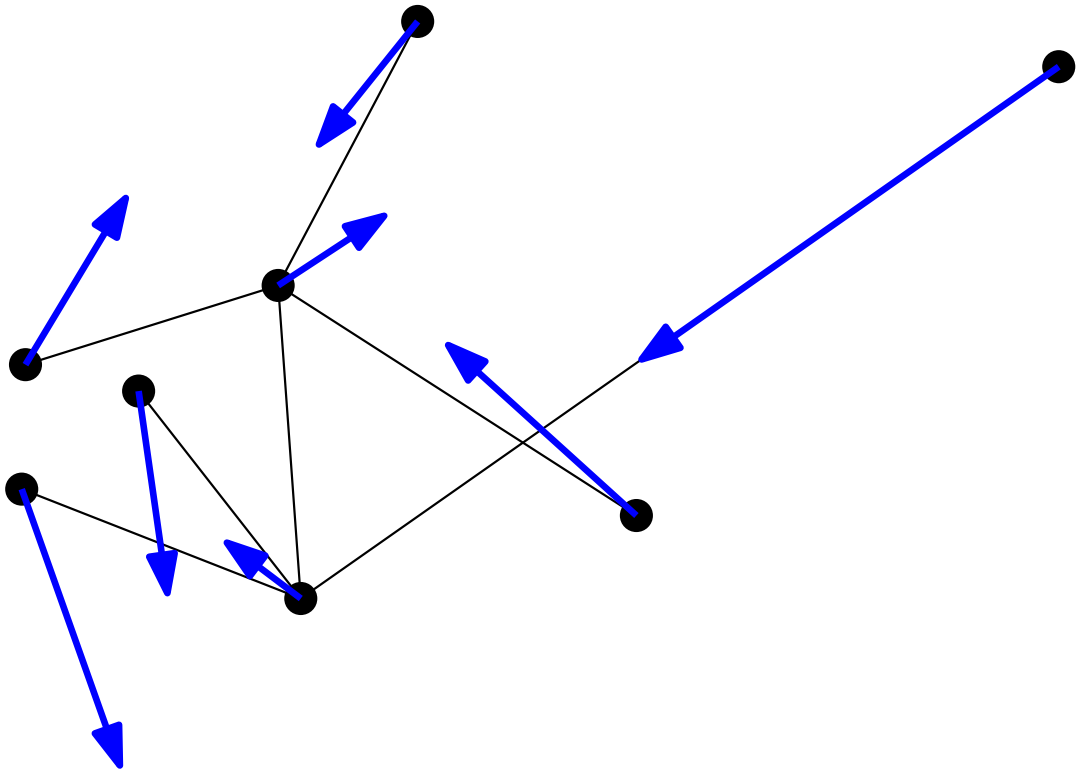
Force Directed Drawing - Beispiel



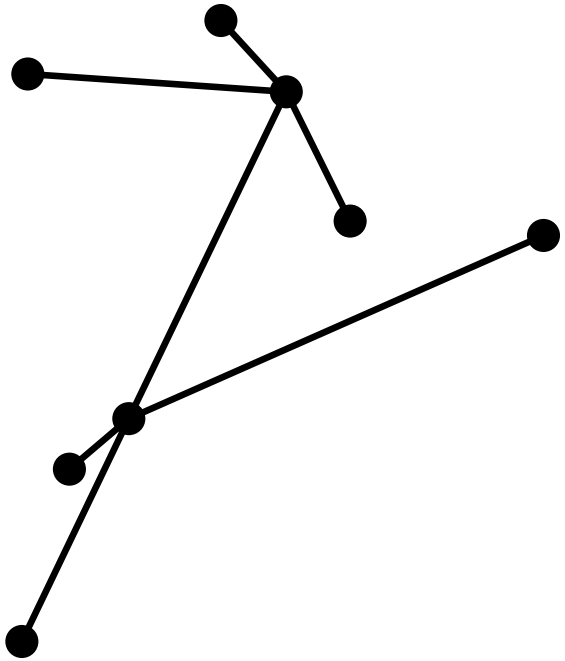
Force Directed Drawing - Beispiel



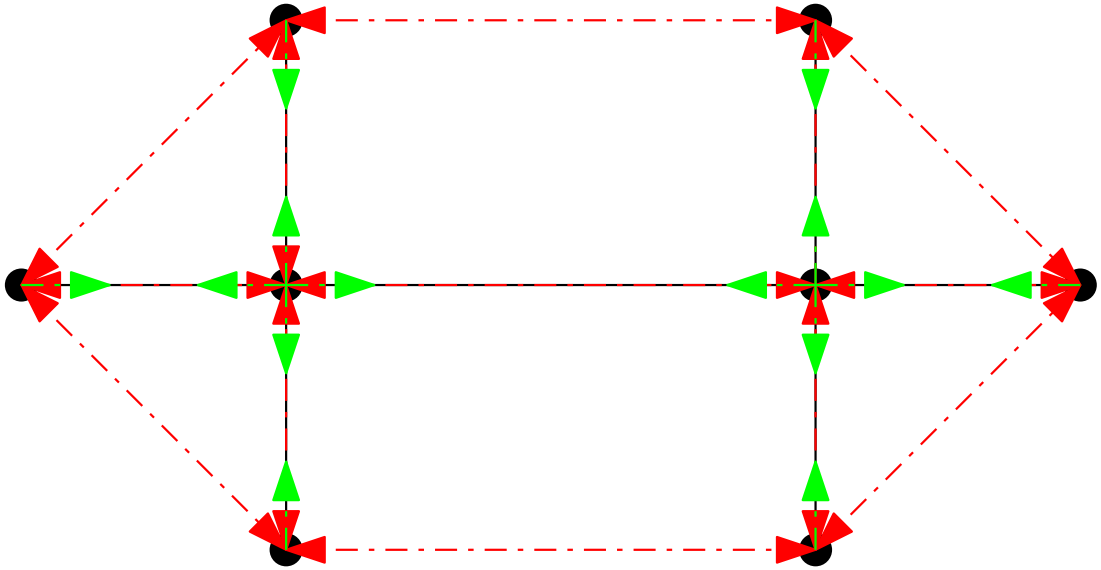
Force Directed Drawing - Beispiel



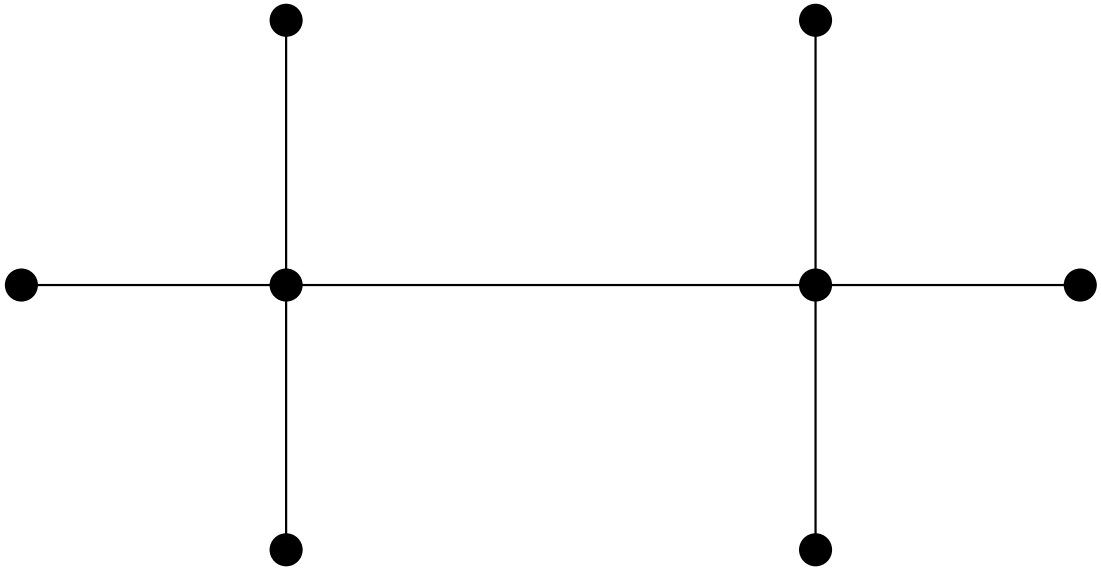
Force Directed Drawing - Beispiel



Force Directed Drawing - Beispiel



Force Directed Drawing - Beispiel

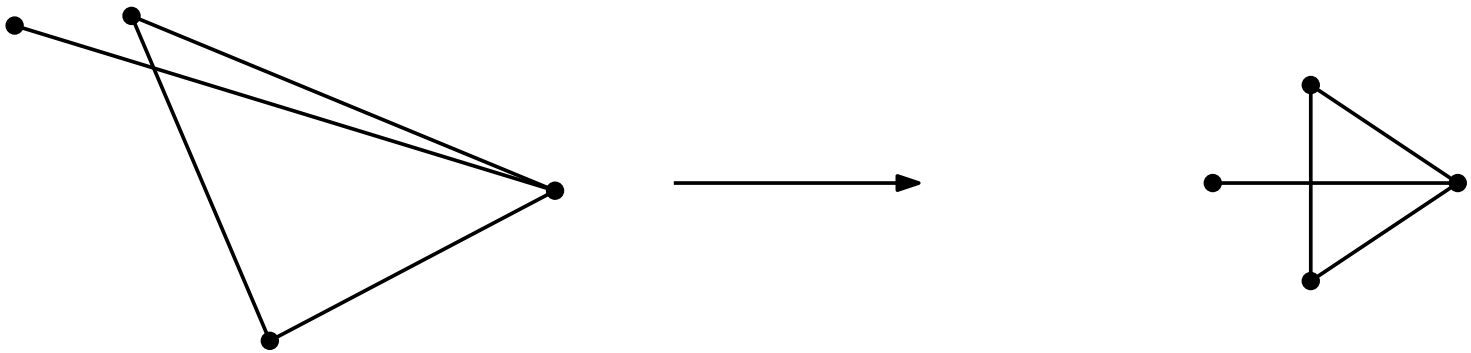


Force Directed Drawing

- Vorteile
  - Erzielt meist gutes Clustering, d.h. stark vernetze Knoten sind nah bei einander und bilden entsprechende „Grüppchen“
  - Geringe Komplexität
- Nachteile
  - Hohe Laufzeit, da Non-Edge-Forces zwischen allen Knoten ($\mathcal{O}(n^2)$) berechnet werden müssen.
  - Das erste gefundene Kräfte-Gleichgewicht kann wesentlich schlechter sein als das beste Ergebnis

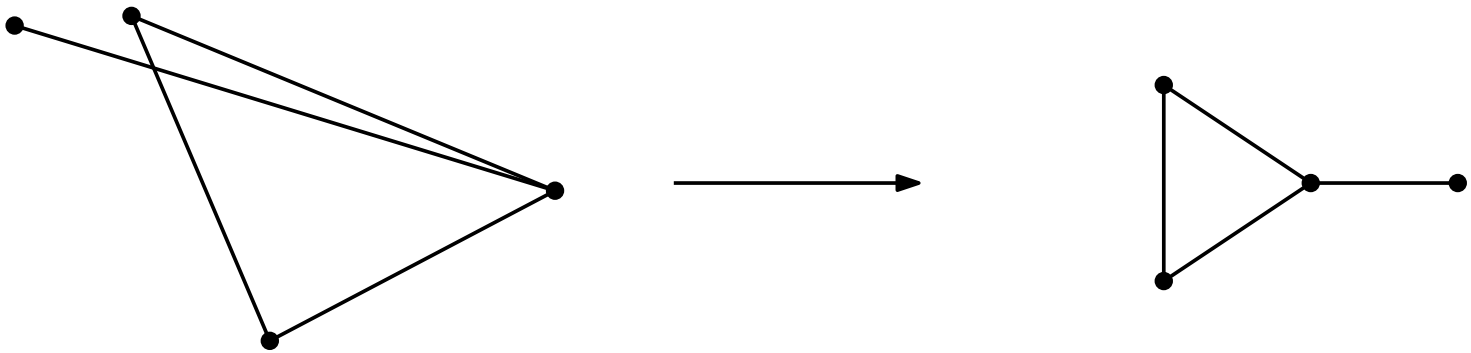
Force Directed Drawing

- Vorteile
  - Erzielt meist gutes Clustering, d.h. stark vernetze Knoten sind nah bei einander und bilden entsprechende „Grüppchen“
  - Geringe Komplexität
- Nachteile
  - Hohe Laufzeit, da Non-Edge-Forces zwischen allen Knoten ($\mathcal{O}(n^2)$) berechnet werden müssen.
  - Das erste gefundene Kräfte-Gleichgewicht kann wesentlich schlechter sein als das beste Ergebnis



Force Directed Drawing

- Vorteile
  - Erzielt meist gutes Clustering, d.h. stark vernetze Knoten sind nah bei einander und bilden entsprechende „Grüppchen“
  - Geringe Komplexität
- Nachteile
  - Hohe Laufzeit, da Non-Edge-Forces zwischen allen Knoten ($\mathcal{O}(n^2)$) berechnet werden müssen.
  - Das erste gefundene Kräfte-Gleichgewicht kann wesentlich schlechter sein als das beste Ergebnis



FADE

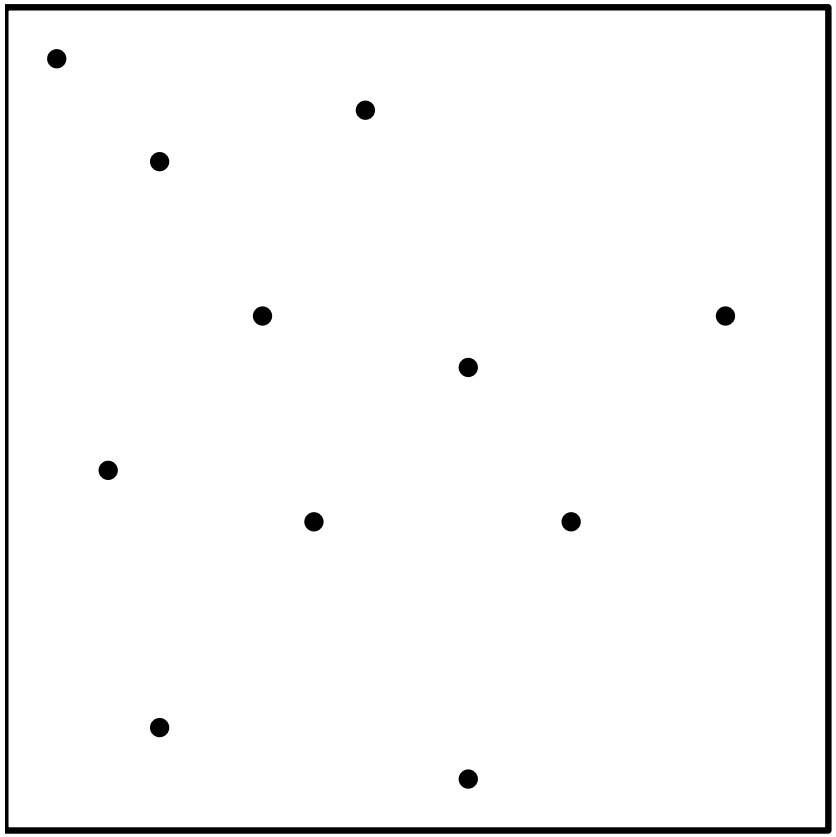
- FADE beschleunigt diese Algorithmen, indem es Knoten zur Berechnung der Non-Edge-Forces bei weit entfernten Knoten gruppiert
  - $n \cdot log(n)$ statt n^2 Berechnungen

- Hierfür werden sogenannte Quad-Trees erstellt

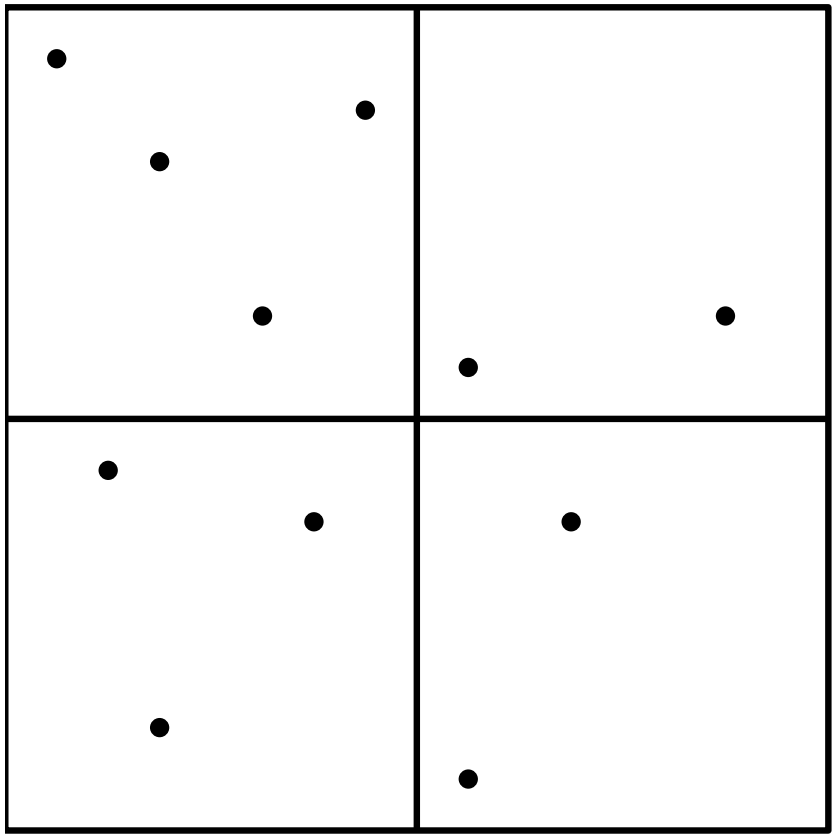
Quad-Trees

- Teilt den Raum des Graph in 4 Teile, falls ein Teil mehr als einen Knoten enthält wird dieser wieder in 4 Teile aufgeteilt usw.
- Hieraus ergibt sich ein Baum bei dem jeder Nicht-Blatt-Knoten genau 4 Kinder besitzt und jedes Blatt entweder einen oder keinen Knoten enthält
- Zusätzlich speichert man sich in jedem Knoten wieviele Knoten er repräsentiert und die mittleren Position dieser Punkte

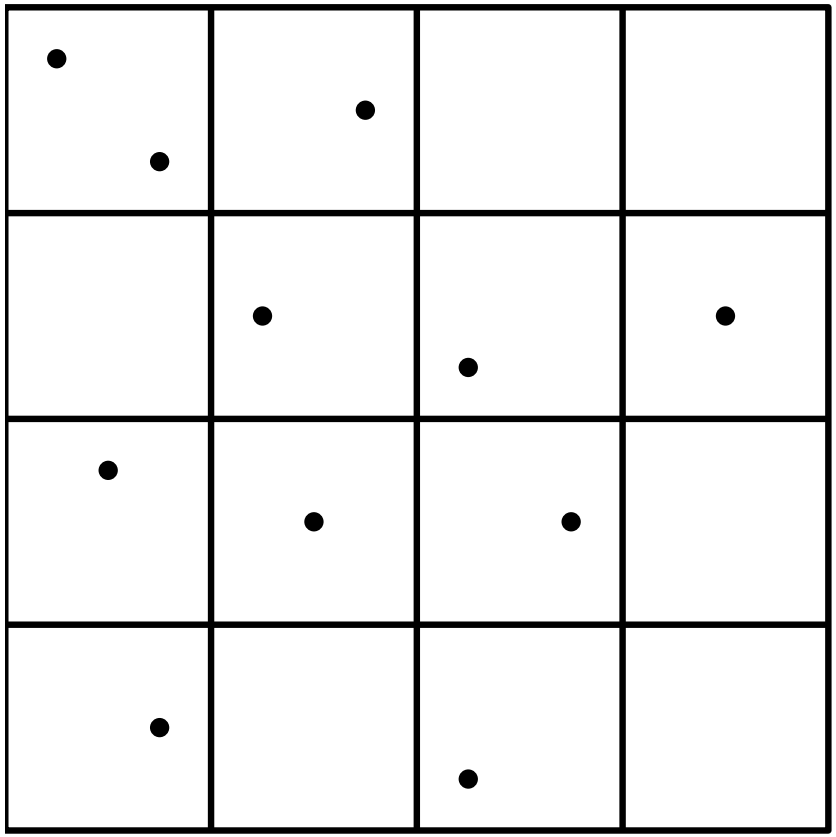
Quad-Trees - Beispiel



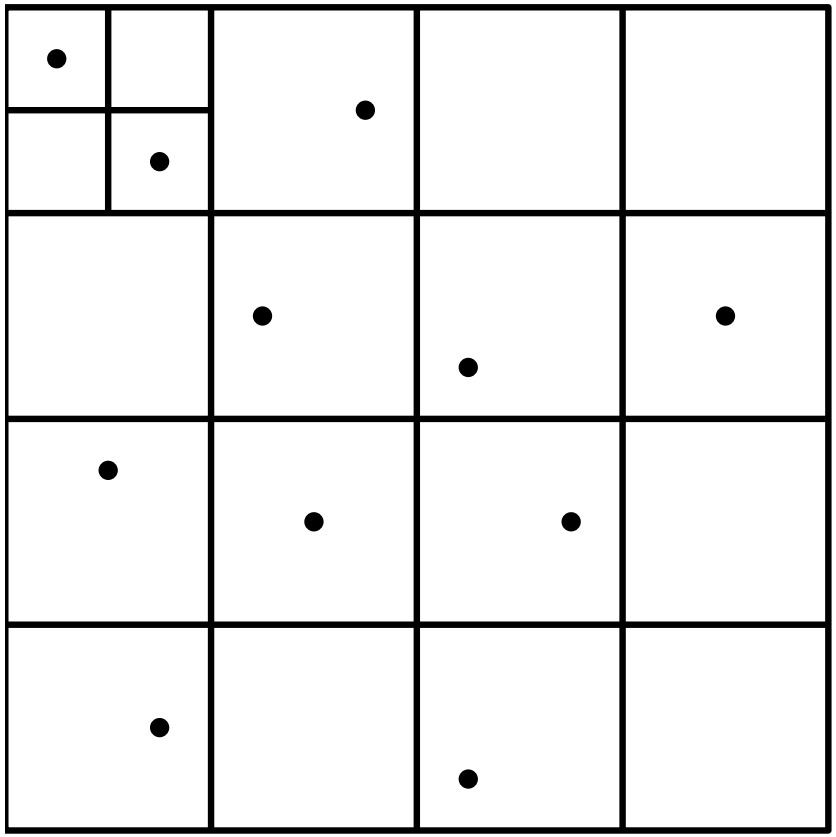
Quad-Trees - Beispiel



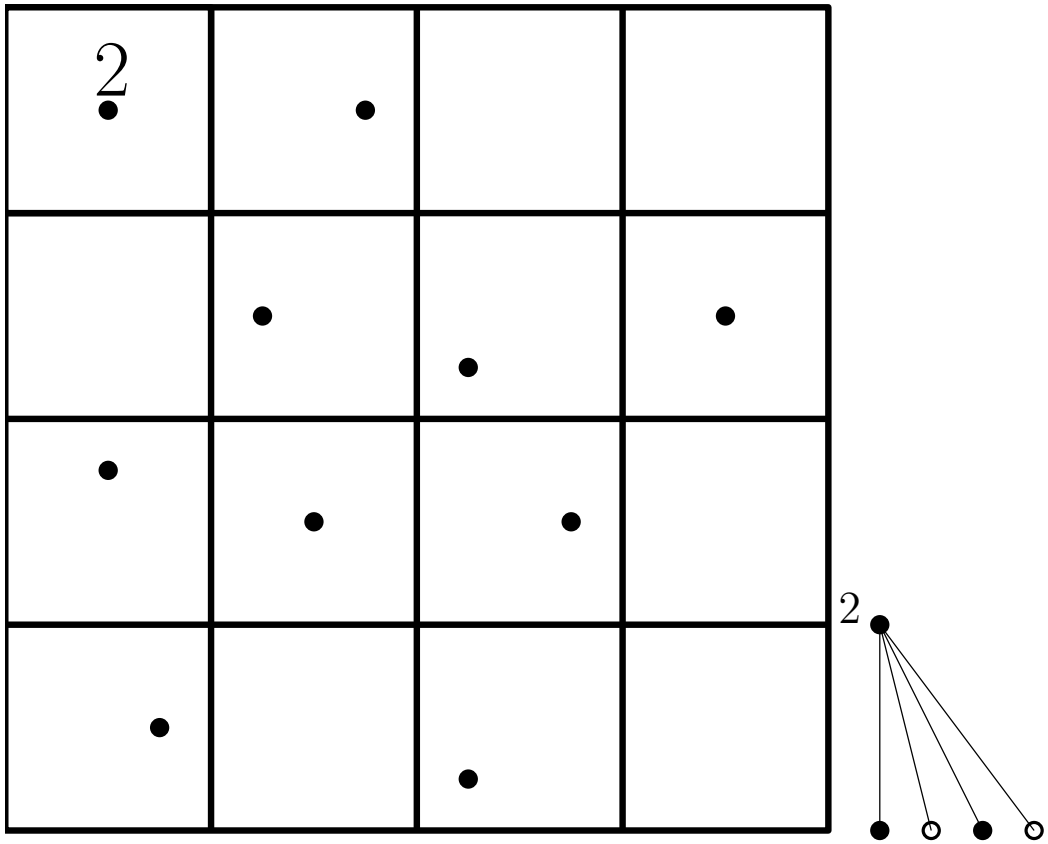
Quad-Trees - Beispiel



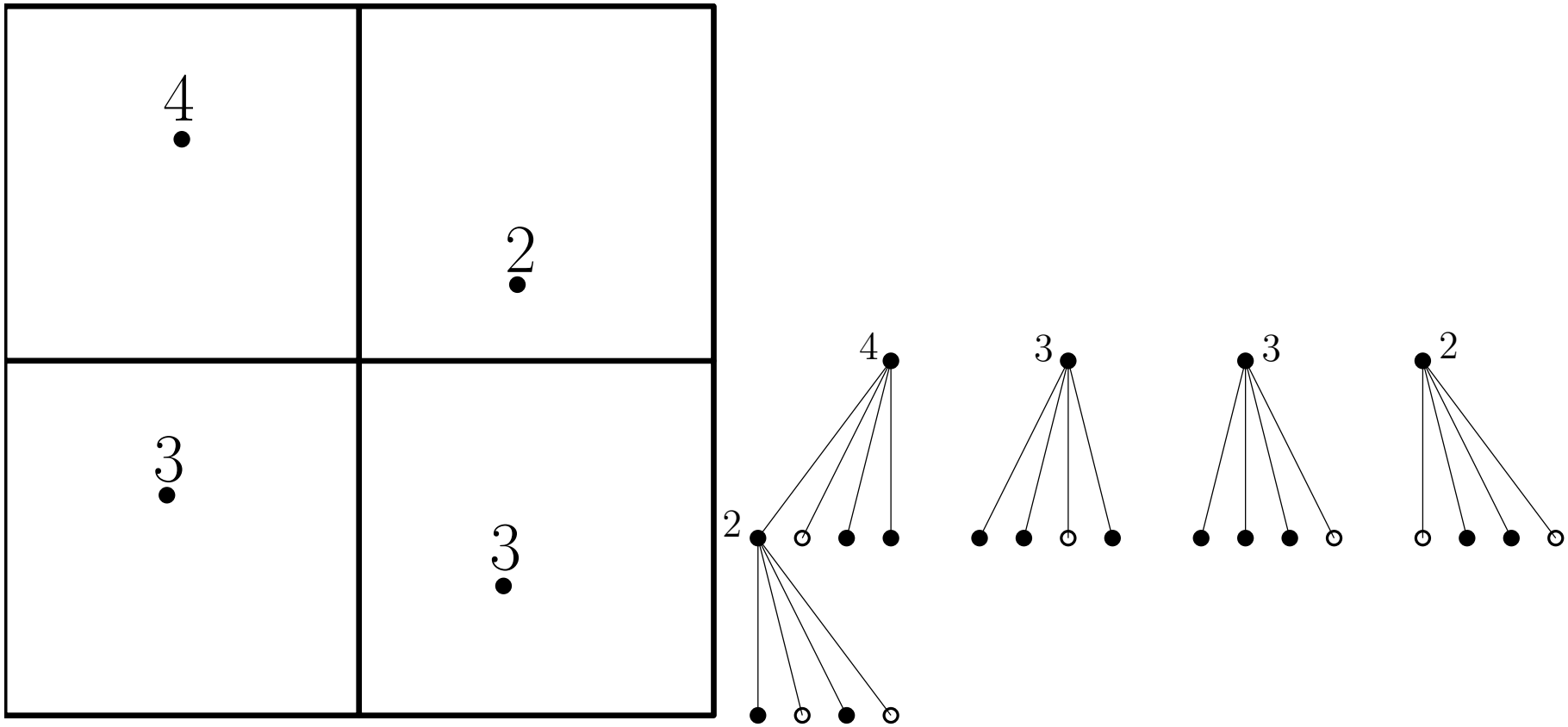
Quad-Trees - Beispiel



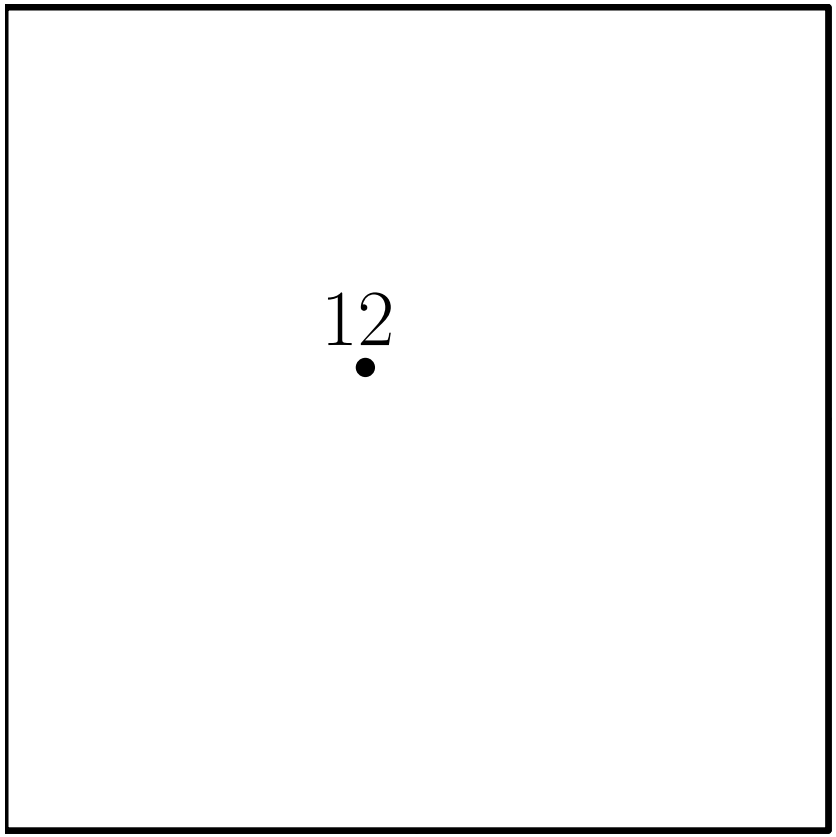
Quad-Trees - Beispiel

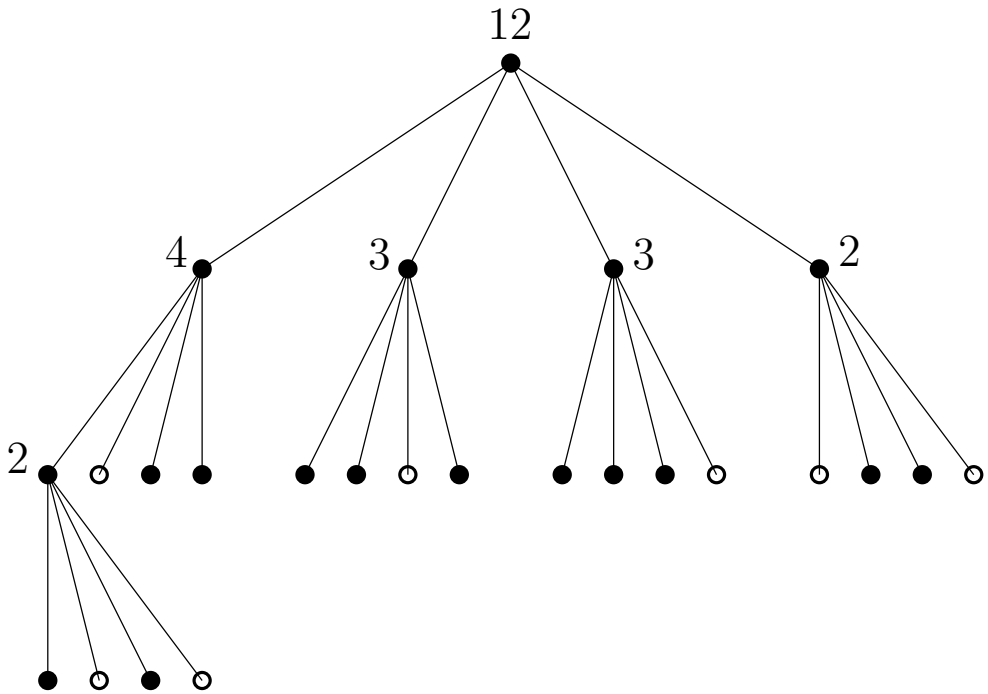


Quad-Trees - Beispiel



Quad-Trees - Beispiel





FADE Ablauf

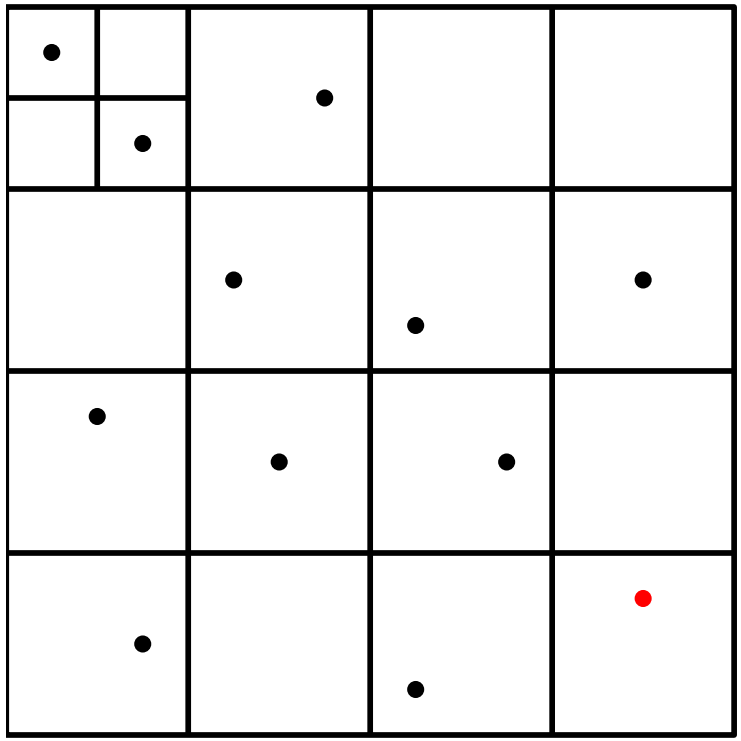
1. Beginne mit initialer Einbettung
2. Repeat
   - 2.1 Berechne Edge-Forces
   - 2.2 Erzeuge Quad-Tree des Graphen
   - 2.3 Berechne Non-Edge-Forces für alle Knoten
     - Berechne zunächst zwischen dem Knoten und den Baumknoten, falls (einer) dieser Baumknoten *nahe* ist, teile diesen Knoten in seine 4 Kinder auf und wiederhole entsprechend für alle 4 Kinder
   - 2.4 Wende Kräfte an
3. Until: Konvergenz

FADE Non-Edge-Forces

- Nahe bedeutet $\frac{s}{d} > \theta$

s = Zellenbreite des Knotens ($\frac{Graphbreite}{4^{Baumlevel}}$)
d = Distanz zwischen Knoten und Baumknoten
θ = Toleranzparameter (bspw. 1)

- $\frac{s}{d} > 1 \Leftrightarrow s > d$

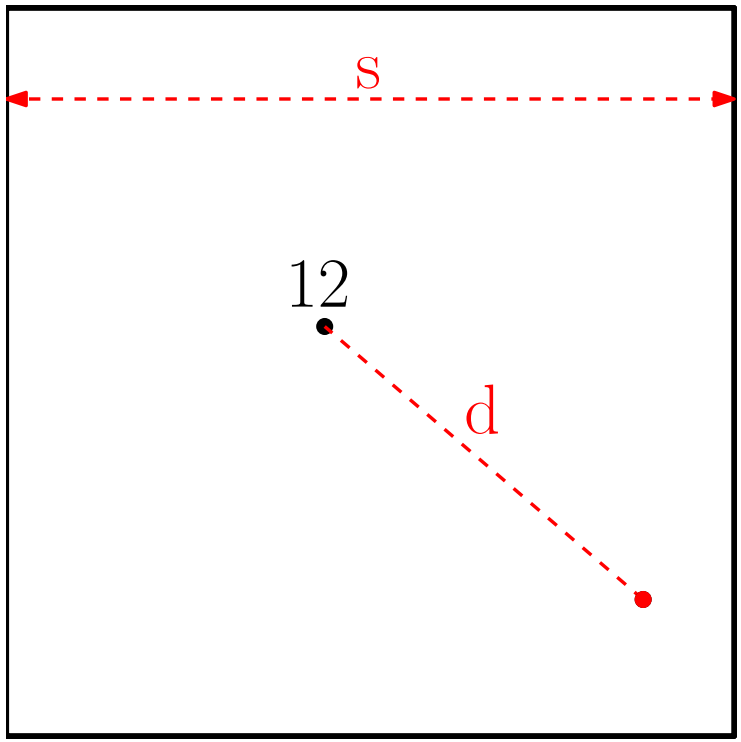


FADE Non-Edge-Forces

- Nahe bedeutet $\frac{s}{d} > \theta$

s = Zellenbreite des Knotens ($\frac{Graphbreite}{4^{Baumlevel}}$)
d = Distanz zwischen Knoten und Baumknoten
θ = Toleranzparameter (bspw. 1)

- $\frac{s}{d} > 1 \Leftrightarrow s > d$

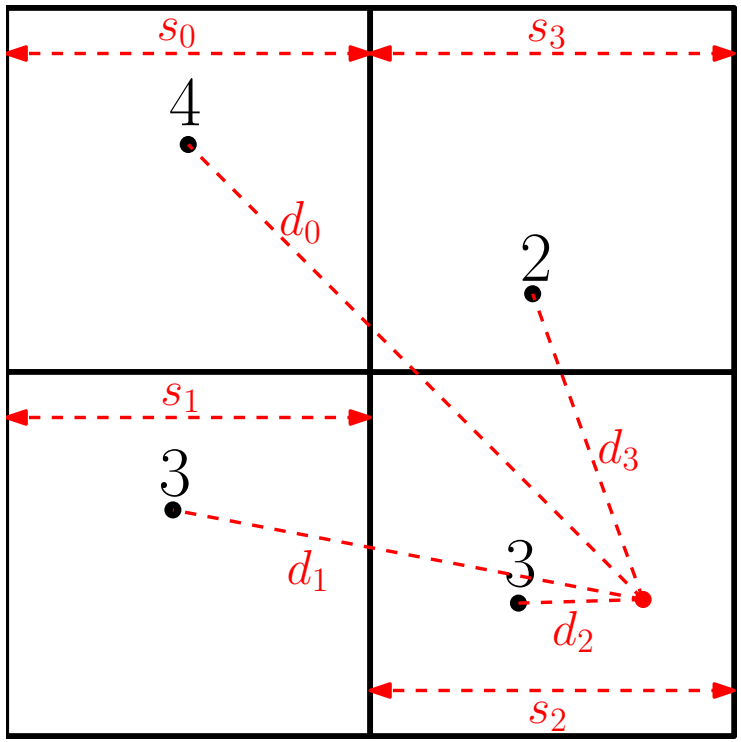


FADE Non-Edge-Forces

- Nahe bedeutet $\frac{s}{d} > \theta$

s = Zellenbreite des Knotens ($\frac{Graphbreite}{4^{Baumlevel}}$)
d = Distanz zwischen Knoten und Baumknoten
θ = Toleranzparameter (bspw. 1)

- $\frac{s}{d} > 1 \Leftrightarrow s > d$

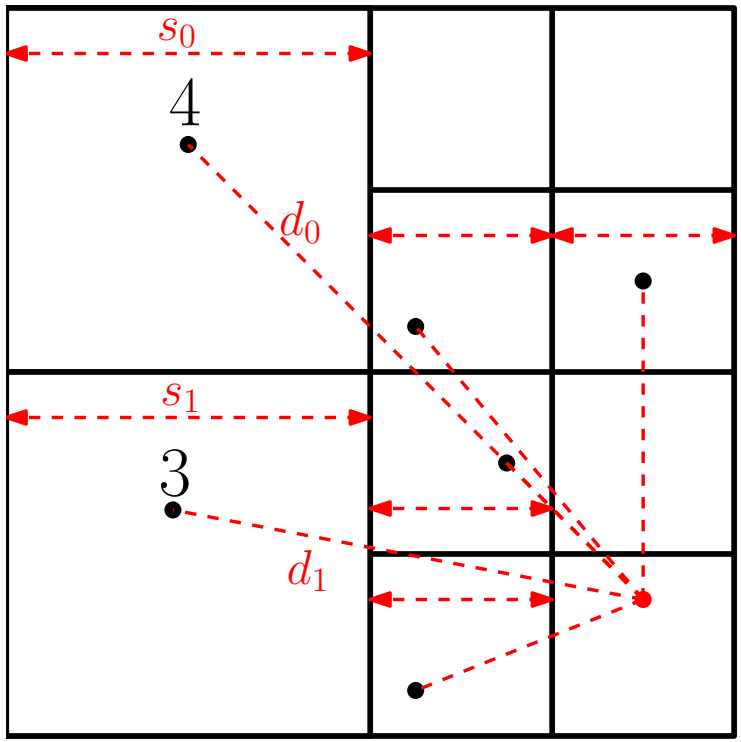


FADE Non-Edge-Forces

- Nahe bedeutet $\frac{s}{d} > \theta$

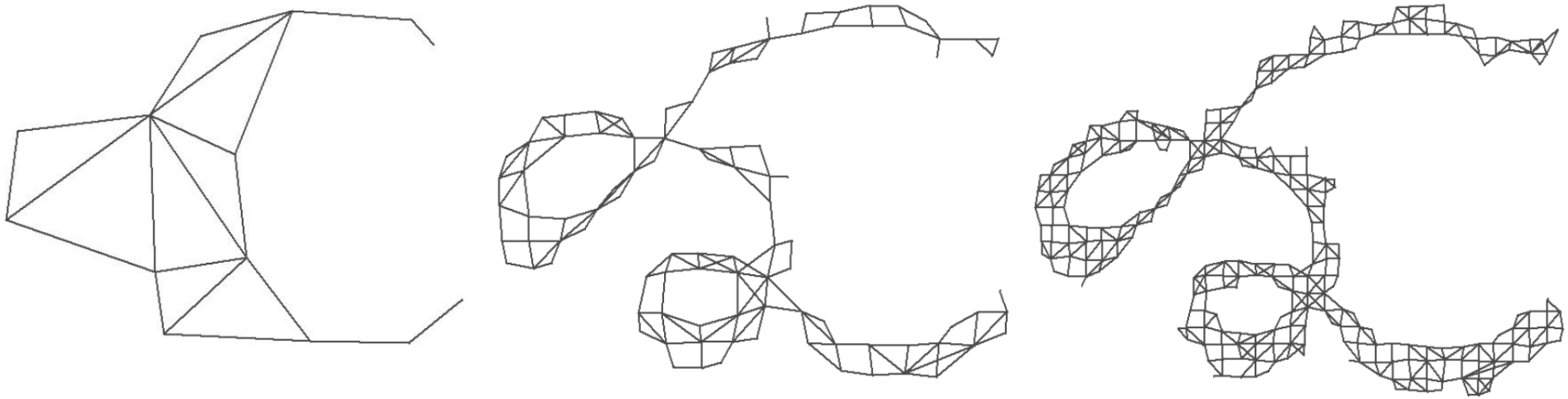
s = Zellenbreite des Knotens ($\frac{Graphbreite}{4^{Baumlevel}}$)
d = Distanz zwischen Knoten und Baumknoten
θ = Toleranzparameter (bspw. 1)

- $\frac{s}{d} > 1 \Leftrightarrow s > d$



Auswertung

- Beschleunigt die Berechnung der Non-Edge-Forces in Force-Directed-Drawing Algorithmen wesentlich
- Erzeugt Quadtrees, die man zur Abstraktion des Graphen verwenden kann
  - Bedingungen für Kanten zwischen Pseudoknoten bei FADE nicht spezifiziert



Verwendung für Argumentkarten

- Der Algorithmus geht von punktförmigen Knoten aus
  - Man kann auch einheitliche, größere Knoten zeichnen, indem man die abstoßenden Kräfte anpasst um einen Mindestabstand zu erzwingen
- Algorithmus erzeugt automatisches Clustering, das über Quad-Tree abstrahiert werden kann
  - Allerdings ohne manuelle Kategorisierung
  - Bedingungen für Kanten zwischen Pseudoknoten nicht spezifiziert
- Kleine Änderungen werden i.d.R. die bisherige Struktur nicht stark verändern

Lizenzen