Online Occlusion Culling

Studienarbeit

Florian Schoppmann

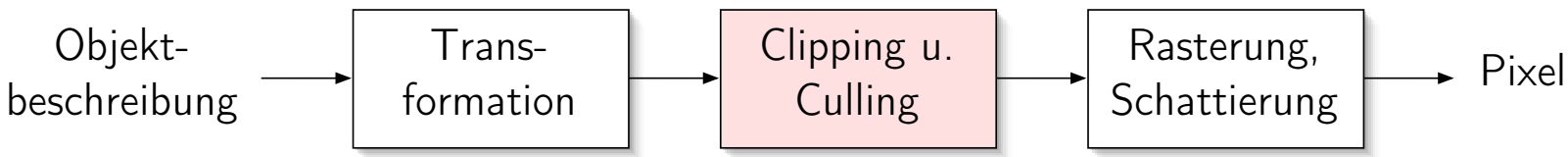
Betreuer: Dr. Christian Sohler

Fachgruppe Algorithmen und Komplexität
Fakultät für Elektrotechnik, Mathematik und Informatik
Universität Paderborn

19. Januar 2005

Motivation

- Computergrafik: Effizientes Darstellen dreidimensionaler Szenen
- Objekte zusammengesetzt aus Primitiven
- Occlusion Culling: Aussortieren nicht sichtbarer Objekte
  $\longrightarrow$ Entlastung der Rendering-Pipeline



Annahme für Culling-Verfahren:

- Ohne Einfluss auf Rendering
- Kein direkter Zugriff auf Datenstrukturen der Szene
  $\longrightarrow$ Zwischenspeicher beschränkter Größe
- Aussortierung bei Verdeckung durch Objekt im Zwischenspeicher

Problemstellung

- Culling-Algorithmus erhält Objekte der Reihe nach
- Bei Ankunft eines Objektes:
  1. Verdeckung (durch Zwischenspeicher) $\rightarrow$ Verwerfen
  2. Keine Verdeckung $\rightarrow$ Zeichnen
     optional: in Zwischenspeicher, wenn voll: Verdrängung eines anderen Objektes

Offline-Problem:

- Culling-Algorithmus kennt Eingabe
- Verdrängungsstrategie: Stets das am wenigsten nützliche Objekt

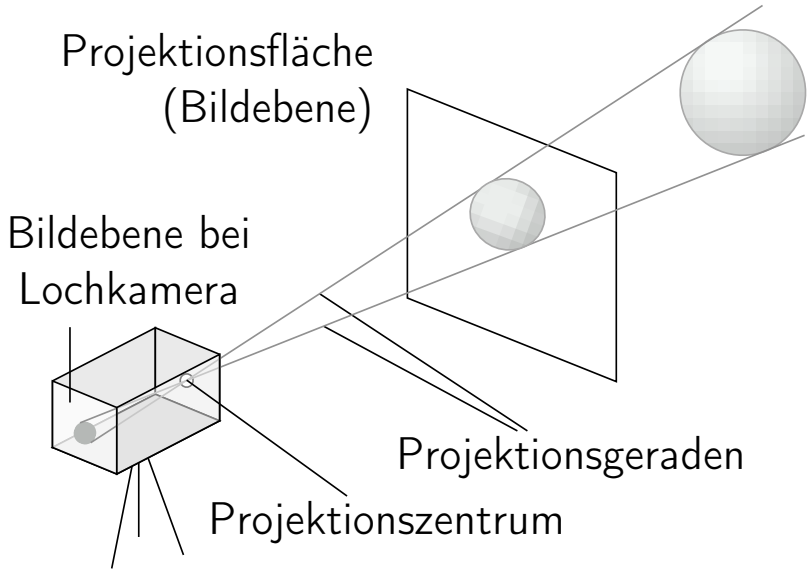
Online-Problem:

- Culling-Algorithmus ohne Kenntnis noch folgender Objekte
- Welche Objekte im Zwischenspeicher bringen den meisten Nutzen?

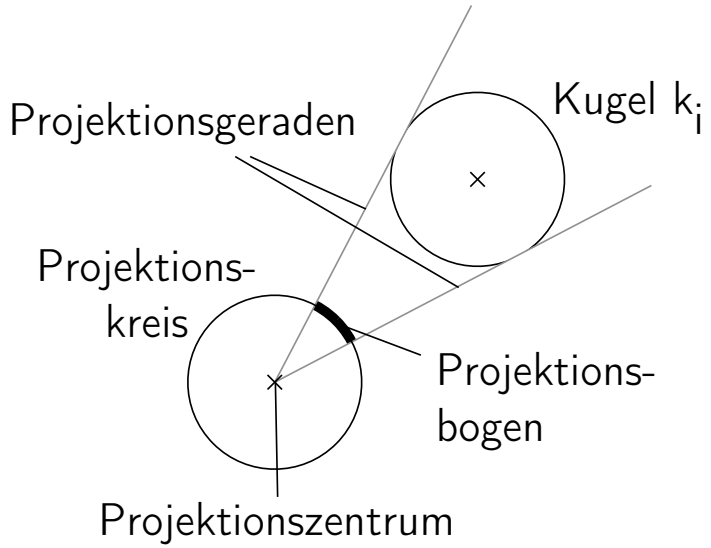
Gliederung

Das Modell

Üblich: An die Fotografie angelehntes Modell



In dieser Arbeit:
Zweidimensionale Modellierung



Szene (informelle Definition):

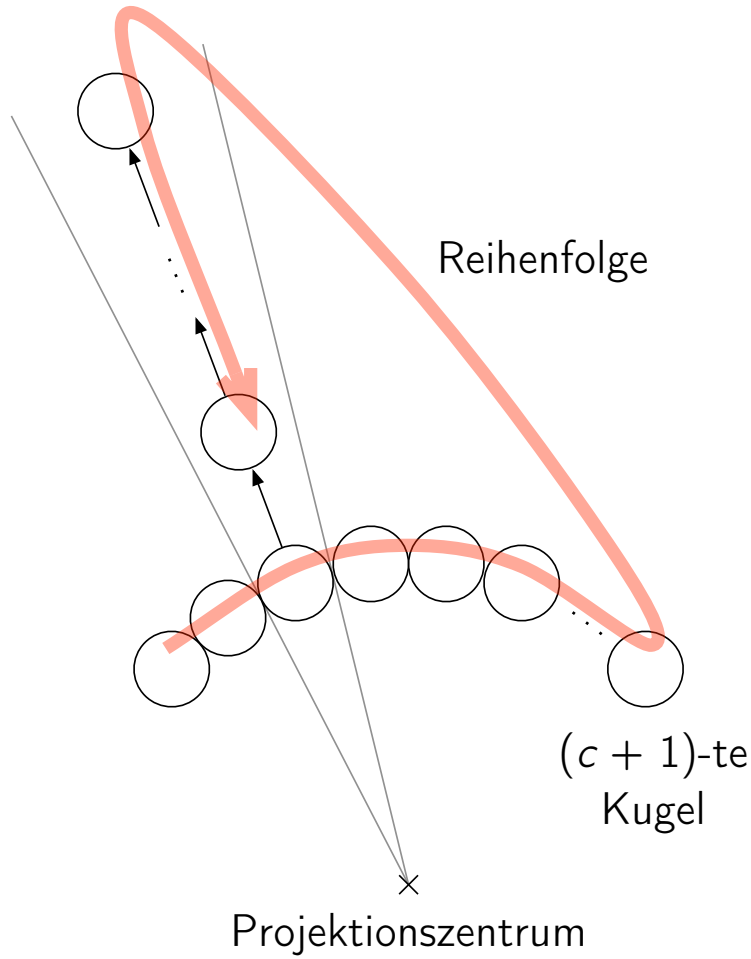
- Menge von Kugeln $\{k_1, \ldots, k_n\}$ im $\mathbb{R}^2$, wobei k_i Mittelpunkte
- Eine Kugel k_i *verdeckt* k_j, wenn Projektionsbogen von k_i den von k_j echt „überlappt“. Symbolisch: $k_i \gg k_j$
- Abstand einer Kugel k_i vom Projektionszentrum: $\operatorname{rad} k_i$

Bewertung von Online-Algorithmen

Competitive Ratio: $\inf\{\gamma \mid \text{ALG}[I] \leq \gamma \cdot \text{OPT}[I] + \alpha \quad \text{für alle } I \in \mathcal{I}\}$

Problem:

- Online Culling-Algorithmen mit konstant großem Zwischenspeicher nie $o(n)$-competitive
  Eingabe:
  - Zwischenspeicher-Größe c
  - Kugeln und Reihenfolge wie rechts
- OBdA wird nach $(c+1)$-ter Kugel die dritte aus Zwischenspeicher verdrängt
- Offline-Algorithmus hätte andere Kugeln verdrängt



Erweiterung der Competitive Analysis

Ergebnis:

- Kostenverhältnis von Online- zu Offline-Algorithmus bei einzelner beliebig gewählter Eingabe unbeschränkt
- Sogar wenn Offline-Algorithmus nur mit Zwischenspeicher der Größe 1
- Auch bei Randomisierung / unterschiedlichen Ressourcen

Koutsoupias & Papadimitriou (2000): Alternative, in der Praxis realistischere Bewertung von Online-Algorithmen:

$$\inf \left\{ \gamma \mid \mathbf{E}_D[\mathrm{ALG}[I]] \leq \gamma \cdot \mathbf{E}_D[\mathrm{OPT}[I]] + \alpha \quad \begin{array}{l} \text{für alle } D \in \Delta, \\ I \text{ ZG mit } I \sim D \end{array} \right\}$$

Δ: Klasse von erlaubten Eingabeverteilungen

In dieser Arbeit: Δ enthält nur Gleichverteilung

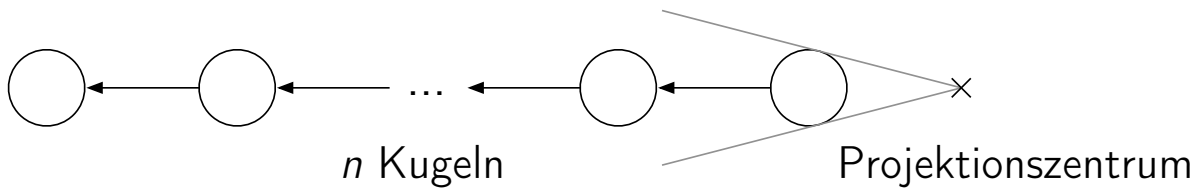
Gliederung

Unbeschränkter Zwischenspeicher

Optimal: Aufnahme jedes Objektes in den Zwischenspeicher

Beispiele:
K_i jeweils Indikatorzufallsgröße, dass k_i gezeichnet werden muss

- Feste Anordnung, zufällige Reihenfolge:
  $\mathbf{Pr}[K_i = 1] = \frac{1}{j+1}$, wobei j Anzahl der Kugeln, die k_i verdecken
- n Kugeln auf einer Projektionslinie, zufällige Reihenfolge:



Erwartete Anzahl gezeichneter Objekte insgesamt:

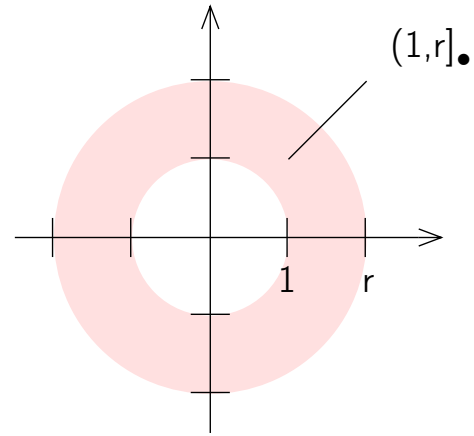
$$\sum_{i=1}^{n} \mathbf{E}[K_i] = \sum_{i=1}^{n} \mathbf{Pr}[K_i = 1] = \sum_{i=1}^{n} \frac{1}{i} = H_n = \Theta(\ln n)$$

Zufällige Anordnungen

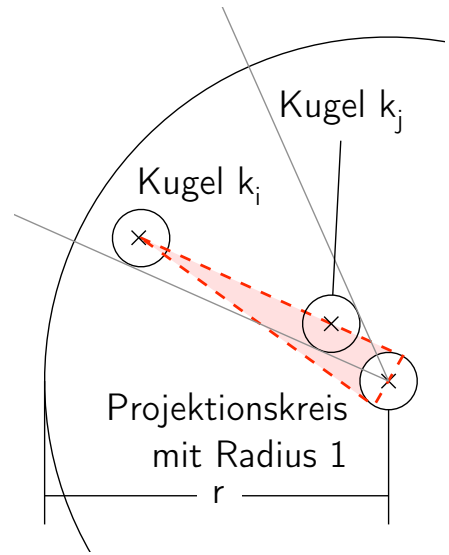
Kugeln zufällig gleichverteilt in $(1, r]_\bullet$, Reihenfolge zufällig gleichverteilt

R Zufallsgröße für Abstand einer beliebigen Kugel vom Projektionszentrum. $F(x) := F_R(x)$:

$$F(x) = \begin{cases} 0 & \text{für } x \leq 1 \\ \frac{\pi(x^2-1)}{\pi(r^2-1)} = \frac{x^2-1}{r^2-1} & \text{für } 1 < x < r \\ 1 & \text{für } x \geq r \end{cases}$$



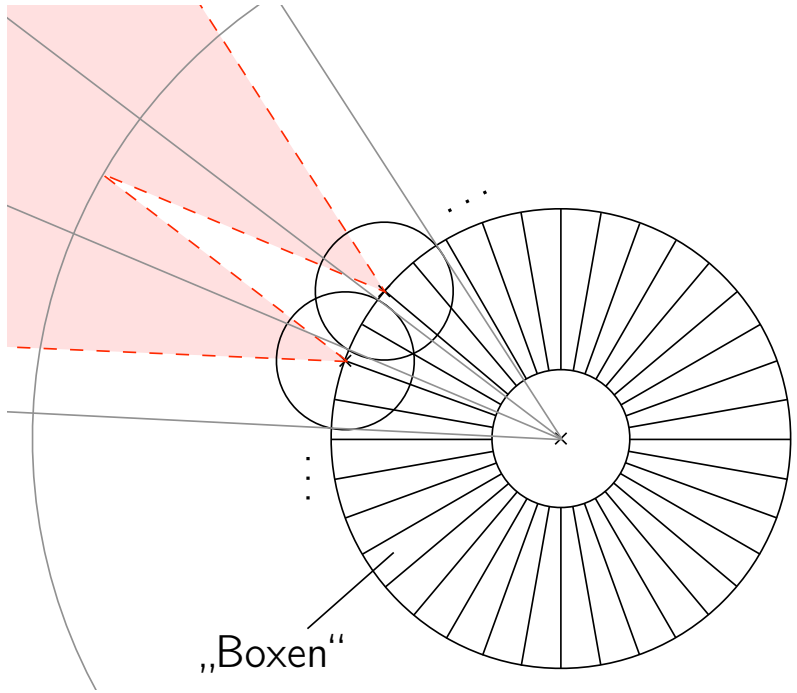
- Exakte Berechnung prinzipiell möglich, aber kaum handhabbar
- Wahrscheinlichkeit für $k_i \gg k_j$ von ungünstiger Gestalt (Zähler rot markiert, Nenner wäre $\pi r^2 - 1$)

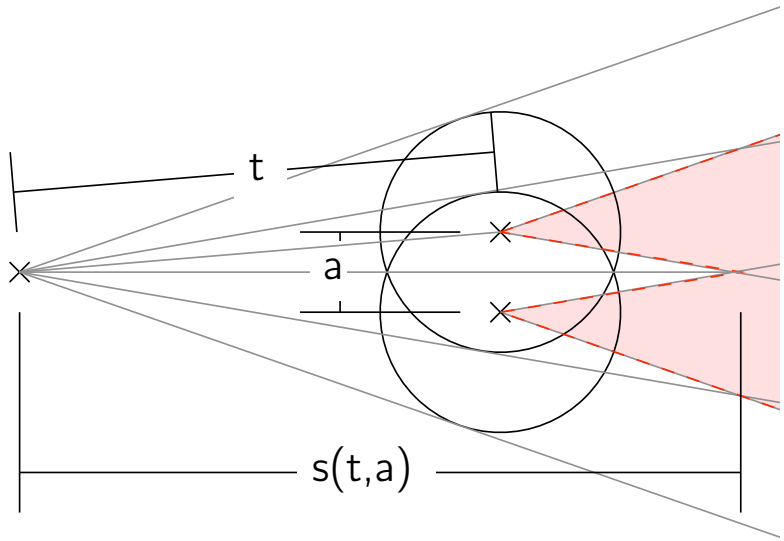


Abschätzung

Idee:

- Einteilung eines Kranzes um den Projektionskreis in „Boxen“
- Dabei: In jeder Box eine Kugel $\Rightarrow \exists s$: jede Kugel k_i mit rad $k_i \geq s$ wird verdeckt





Problem: Wie Parameter t und a wählen, so dass sich eine gute Schranke ergibt?

Das Balls into Bins-Modell

Definition (Unendliches *Balls into Bins*-Modell)

Gegeben m Boxen und eine unbegrenzte Anzahl von Kugeln. Es wird der Reihe nach auf die m Boxen geworfen. Jeder Wurf trifft zufällig unabhängig gleichverteilt genau eine Box.

Im Erwartungswert $m \cdot H_m$ Kugeln, bis keine Box mehr leer

Culling-Algorithmus: Einteilung von Eingabe in zwei Phasen:

- Erste Phase: Es gibt Boxen ohne Kugel
  $\longrightarrow$ Obere Abschätzung: Zeichnung jeder Kugel
  Länge der ersten Phase: Balls into Bins-Modell!
- Zweite Phase: Jede Box enthält Kugel, jede ankommende Kugel k_i mit $\operatorname{rad} k_i \geq s$ verdeckt
  $\longrightarrow$ Obere Abschätzung: Zeichnung jeder Kugel mit Wahrscheinlichkeit $F(s)$

Obere Schranke für Anzahl zu zeichnender Kugeln

M ZG für Anzahl gezeichneter Kugeln insgesamt
N Zufallsgröße für Anzahl Kugeln, bis keine Box mehr leer

$$\mathbf{E}[M] < \mathbf{E}[N] + \mathbf{E}[(n-N)\cdot F(s)] < \frac{1-F(s)}{F(t)} \cdot m \cdot H_m + n \cdot F(s)$$

Startwerte für t und m gegeben, asymptotische Abschätzung:

- Anzahl Kugeln ver-16-facht
- „Dicke“ des Kranzes halbiert
- Anzahl Boxen verdoppelt

$\Rightarrow$ Phase 1 ca. 8-mal länger, in Phase 2 Zeichnung max. 8-mal so vieler Kugeln

$\Rightarrow$ Anzahl Kugeln: n, gezeichnete Kugeln ca. $\mathcal{O}(8^{\log_{16} n}) = \mathcal{O}(n^{3/4})$

Exaktes Ergebnis: Erwartete Anzahl gezeichneter Kugeln:

$$\mathcal{O}(n^{3/4} \cdot \log_{16} n)$$

Gliederung

Schranken im Modell mit beschränktem Zwischenspeicher

Heuristiken:

- Maximal überdeckte Fläche
- Einteilung eines Kranzes in Boxen (wie zuvor)
  Problem: Bei konstant großem Zwischenspeicher nur konstant viele Boxen!

Zwischenspeicher der Größe n^c mit $c \in (0, 1)$:

- Wende vorherige Überlegungen an
- $\exists$ Algorithmus mit Schranke $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$, wenn $c \geq \frac{1}{4}$

Sei ALG[I] solch ein Online-Algorithmus:

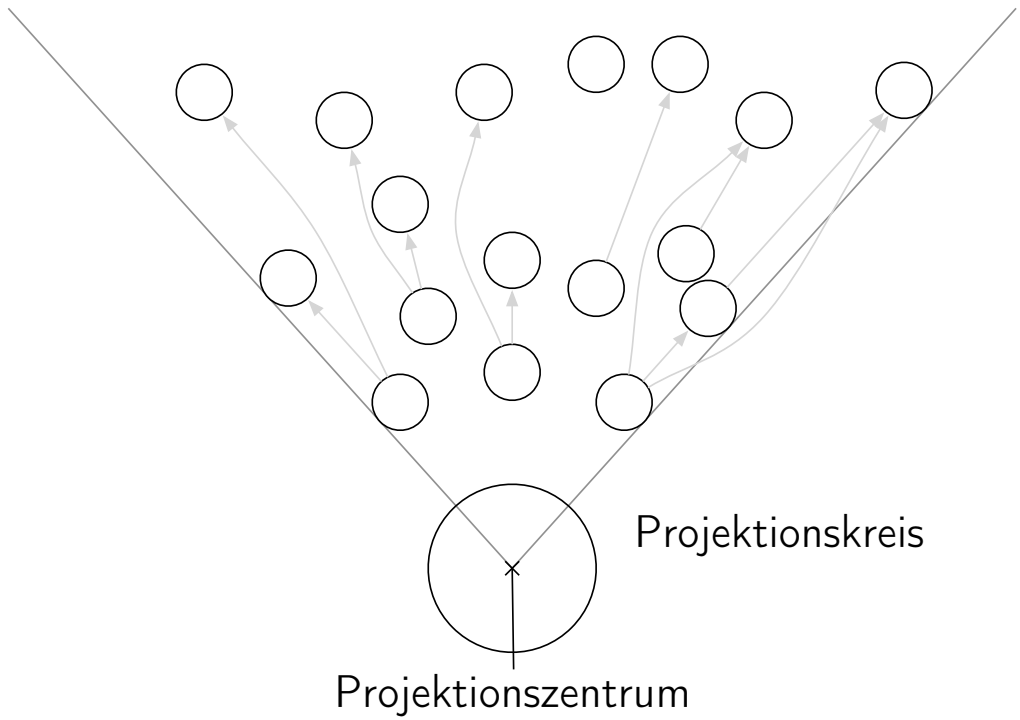
$$\frac{\mathbf{E}[\text{ALG}[I]]}{\mathbf{E}[\text{OPT}[I]]} = \mathcal{O}\left(\frac{n^{3/4} \cdot \log_{16} n}{\ln n}\right) = \mathcal{O}(n^{3/4})$$

Gliederung

Verdeckungsgraphen

Definition

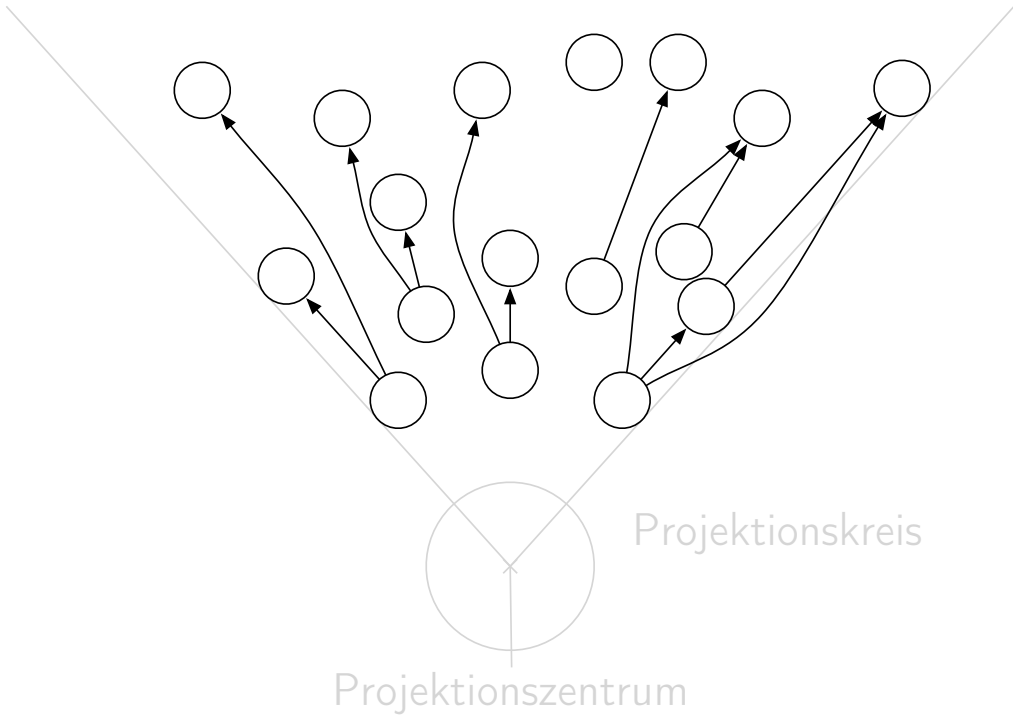
Sei $V = \{k_1, \ldots, k_n\}$ Menge von Kugeln. Ferner sei $E := \{(k_i, k_j) \in V^2 \mid k_i \gg k_j\}$. Dann heißt der gerichtete Graph $G = (V, E)$ *Verdeckungsgraph* zu V.



Verdeckungsgraphen

Definition

Sei $V = \{k_1, \ldots, k_n\}$ Menge von Kugeln. Ferner sei $E := \{(k_i, k_j) \in V^2 \mid k_i \gg k_j\}$. Dann heißt der gerichtete Graph $G = (V, E)$ *Verdeckungsgraph* zu V.



Das Optimierungsproblem

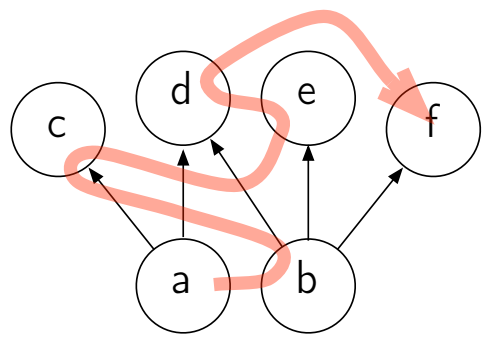
Occlusion Culling als (verallgemeinertes) Graphen-Problem *Knotenaussortierung*:

Eingabe: DAG $G = (V, E)$, $n := |V|$, Reihenfolge der Knoten, Zwischenspeicher-Größe c.

Finde: Verdrängungssequenz aus Zwischenspeicher, die Gesamtanzahl aussortierter Knoten maximiert.
Aussortierung eins Knotens v, wenn $\exists v' \in V$ mit:

- v' vor v in der Reihenfolge
- $\exists$ Pfad von v' zu v
- v' zum Zeitpunkt der Betrachtung von v noch im Zwischenspeicher

Beispiel wie rechts:
Optimale Verdrängungssequenz:
$(\varepsilon, a, c, e, d, f)$
Aussortierte Knoten: 3



Reihenfolge:
a, b, c, e, d, f

Zwischenspeicher:
Größe 1

Entscheidungsproblem zum Culling-Problem

Frage: Gibt es effizienten (Offline) Algorithmus für das Culling-Problem?

Zunächst: Problem Knotenaussortierung und sein Entscheidungsproblem

Definition (Knotenaussortierung$_{Ent}$)

Gibt es zu einem Verdeckungsgraphen $G = (V, E)$, einer Knotenreihenfolge, und einem Zwischenspeicher der Größe c eine Verdrängungssequenz, so dass $\geq m$ Knoten aussortiert werden?

Frage: Gibt es ein $\mathcal{NP}$-vollständiges Problem, das sich in polynomieller Zeit auf Knotenaussortierung$_{Ent}$ reduzieren lässt?

Wenn ja, würde folgen:

- Knotenaussortierung$_{Ent}$ $\mathcal{NP}$-schwer
- Sofern $\mathcal{P} \neq \mathcal{NP}$, Knotenaussortierung nicht effizient lösbar

Entscheidungsproblem zum Culling-Problem

Frage: Gibt es effizienten (Offline) Algorithmus für das Culling-Problem?

Zunächst: Problem Knotenaussortierung und sein Entscheidungsproblem

Definition (Knotenaussortierung$_{Ent}$)

Gibt es zu einem Verdeckungsgraphen $G = (V, E)$, einer Knotenreihenfolge, und einem Zwischenspeicher der Größe c eine Verdrängungssequenz, so dass $\geq m$ Knoten aussortiert werden?

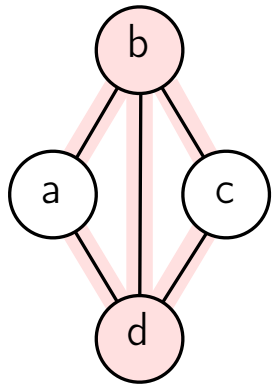
Frage: Gibt es ein $\mathcal{NP}$-vollständiges Problem, das sich in polynomieller Zeit auf Knotenaussortierung$_{Ent}$ reduzieren lässt?

Wenn ja, würde folgen:

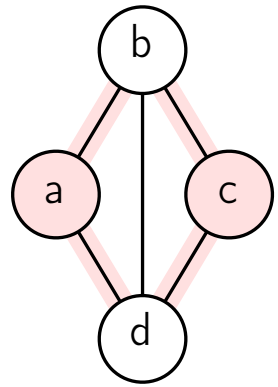
- Knotenaussortierung$_{Ent}$ $\mathcal{NP}$-schwer
- Sofern $\mathcal{P} \neq \mathcal{NP}$, Knotenaussortierung nicht effizient lösbar

Knotenüberdeckung und Knotenaussortierung

Knotenüberdeckung$_{Ent}$: Gibt es zu einem ungerichteten Graphen $G = (V, E)$ und einem $k \in \mathbb{N}_0$ eine Teilmenge $V' \subseteq V$, so dass $|V'| \leq k$ und für jede Kante $\{e, d\} \in E$ wenigstens e oder d in V' enthalten ist?



$\{b,d\}$ 2-Knoten-überdeckung



$\{a,c\}$ *keine* 2-Knoten-überdeckung

Bekannt:

- Knotenüberdeckung$_{Ent}$ ist $\mathcal{NP}$-vollständig

Satz

Knotenüberdeckung$_{Ent}$ *ist polynomiell reduzierbar auf Knotenaussortierung*$_{Ent}$.

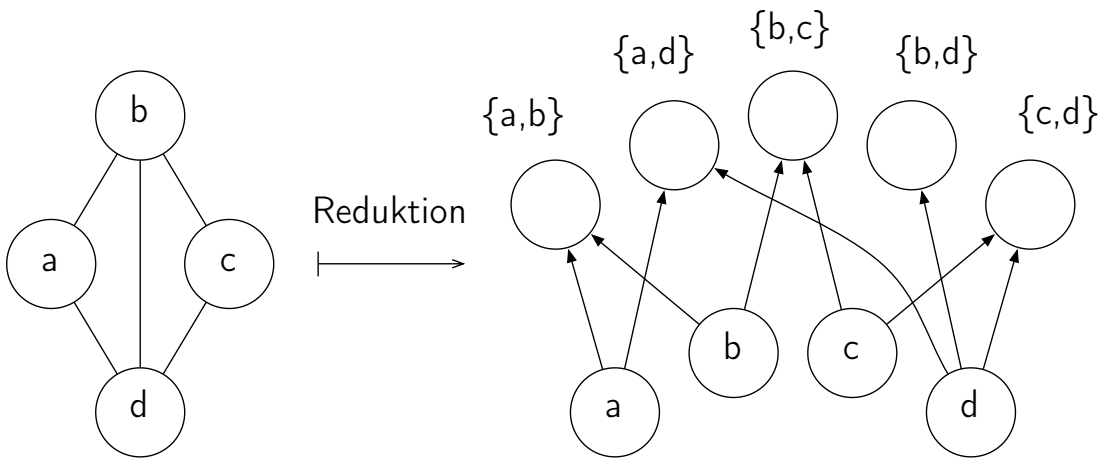
Reduktion von Knotenüberdeckung

Beweisskizze: Verwende folgende Reduktion von Knotenüberdeckung$_{Ent}$ auf Knotenaussortierung$_{Ent}$ (hier ohne Korrektheitsbeweis):

Eingabe: $G = (V, E)$, $k \in \mathbb{N}_0$. Dabei $V := \{v_1, \ldots, v_n\}$, $E := \{e_1, \ldots, e_s\}$ mit $e_i = \{v_j, v_k\}$ für $j, k \in \mathbb{N}_{\leq n}$, $j \neq k$ und $i \in \mathbb{N}_{\leq s}$.

Ausgabe: DAG $G' = (V', E')$ mit $V' = \{v_1, \ldots, v_n, e_1, \ldots, e_s\}$ und $E' = \{(v, e) \mid e \in E \text{ und } v \in e\}$. Reihenfolge der Kugeln genauso. $c = k$. $m = |E|$

Beispiel:



Zusammenfassung

Motivation: Modellierung von Occlusion Culling
Problemstellung: Gute Ausnutzung des Zwischenspeichers

- Konstant großer Zwischenspeicher: Kein Online Culling-Algorithmus $o(n)$-competitive
- Bei zufälliger Eingabe und unbegrenztem Zwischenspeicher: Mindestens $\omega(\ln n)$ und maximal $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$ gezeichnete Kugeln
- Auf n^c, $c \geq \frac{1}{4}$, begrenzter Zwischenspeicher: Es gibt Online-Algorithmus, der maximal $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$ Kugeln zeichnet.
- Das dem Culling-Problem ähnliche Optimierungsproblem Knotenaussortierung ist nicht effizient lösbar, wenn $\mathcal{P} \neq \mathcal{NP}$.

Zum Weiterlesen...


Elias Koutsoupias und Christos H. Papadimitriou (2000):
Beyond Competitive Analysis.
SIAM Journal on Computing 30, Nr. 1, 300–317.
http://epubs.siam.org/sam-bin/dbq/article/29954


Martin Raab und Angelika Steger (1998):
"Balls into Bins" - A Simple and Tight Analysis.
RANDOM '98: Proce. of the Second Intern. Workshop on Randomization and Approximation Techniques in Computer Science, Springer-Verlag.
http://wwwmayr.informatik.tu-muenchen.de/personen/raab/publ/balls.pdf


Alan Watt (2002):
3D-Computergrafik.
3. Auflage.
Pearson Studium, München.