UNIVERSITÄT PADERBORN
Fakultät für Elektrotechnik, Informatik und Mathematik

Studienarbeit

Online Occlusion Culling

Florian Schoppmann
Dörener Weg 61
33100 Paderborn
fschopp@upb.de

Matrikel-Nummer: -------

6. Januar 2005

Betreuer: Dr. Christian Sohler
Fachgruppe Algorithmen und Komplexität

Zusammenfassung

Ein grundlegendes Problem in der Computergrafik ist das effiziente Darstellen (Rendern) von dreidimensionalen Szenen. Solche Szenen bestehen gewöhnlich aus vielen einzelnen Objekten, die wiederum aus vielen geometrischen Primitiven zusammengesetzt sind. Bei der Darstellung einer solchen Szene sind im Allgemeinen nur wenige Objekte vom Betrachter aus sichtbar. Daher versucht man Objekte, die nicht sichtbar sind, zu erkennen und diese gar nicht erst zur Grafikhardware zu schicken. Diesen Prozess bezeichnet man als Occlusion Culling.

In dieser Arbeit wird folgende einfache Methode für das Occlusion Culling modelliert und analysiert, die insbesondere Anwendungen in sich schnell verändernden Szenen (Bewegung, Animation etc.) hat: Bevor ein Objekt zur Grafikhardware geschickt wird, prüft der Culling-Algorithmus, ob es durch eines oder mehrere bereits gezeichnete Objekte aus einem Puffer verdeckt wird. Ist dies nicht der Fall, so wird das Objekt zur Grafikhardware geschickt und gezeichnet. Dann darf das Objekt optional eines der bereits im Puffer gespeicherten Objekte ersetzen. Das zugehörige Online-Problem wird als *Online Occlusion Culling* bezeichnet. In dieser Arbeit wird die Performance von Algorithmen für dieses Problem untersucht und gezeigt, dass kein Online-Algorithmus $o(n)$-competitive sein kann.

Des Weiteren wird eine average-case Analyse vorgenommen. Dazu werden Objekte durch Einheitskreise in der Ebene modelliert. Es wird gezeigt: Gibt es n zufällig verteilte Objekte in einem begrenzten Radius und ist der Zwischenspeicher nicht beschränkt, so existiert für Algorithmen, die jede gezeichnete Kugel in den Zwischenspeicher aufnehmen, im Erwartungswert eine untere Schranke von $\mathcal{O}(\log n)$ und eine obere Schranke von $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$ gezeichneten Kugeln.

Schließlich folgen Überlegungen zur Komplexität des der Modellierung zugehörigen Optimierungsproblems und es wird die $\mathcal{NP}$-Vollständigkeit eines verwandten Graphen-Problems gezeigt.

Inhaltsverzeichnis

1 Einleitung

Interaktive Darstellung und Durchlaufen von umfangreichen geometrischen Modellen stellt trotz der großen in den vergangenen Jahren erzielten Fortschritte im Hardwarebereich immer noch eine Herausforderung für die Computergrafik dar. Parallel dazu wird eine immer größere Detailtreue möglich, die jedoch ihrerseits wieder Optimierungen bei der Grafikverarbeitung erforderlich macht.

Der übliche Ablauf vom Modell im Computer – also den entsprechenden Datenstrukturen, die die Szenerie repräsentieren – bis zum gezeichneten Pixel-Bild ist in Abbildung 1 schematisch dargestellt; vergleiche zum Beispiel Watt (2002, Seite 165). Typisch dabei ist der Einsatz einer *Rendering-Pipeline*, die von jedem Objekt durchlaufen wird und bei der die Arbeitsschritte vorgenommen werden, ohne dass die gesamte Objektmenge als vollständige Eingabe bereits vorliegt. Die Verarbeitung kommt daher der Vorstellung eines Fließband-Ablaufes recht nahe. Dieses Vorgehen wird insbesondere im Bereich der interaktiven Computergrafik eingesetzt, bei der ein hoher Durchsatz erforderlich ist.

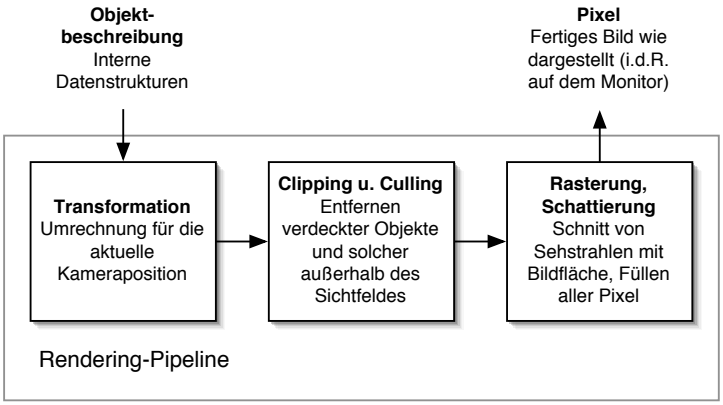


Abbildung 1: Schematische Darstellung der Verarbeitungsschritte bei Computergrafik: Von der Repräsentation bis zum fertigen Bild

Intention hinter Occlusion Culling ist stets eine Entlastung der Rendering-Pipeline (englisch: *to cull* – to separate, select, pick out [Webster's Revised Unabridged Dictionary]). Dabei werden die Verdeckungsbeziehungen zwischen Objekten untersucht und nicht sichtbare Objekte aussortiert.

Ein grundlegendes Prinzip in der Computergrafik zur einfachen und schnellen Verarbeitung ist die Darstellung komplexer Objekte durch viele „primitive“ geometrische Strukturen – üblicherweise ausschließlich Polygone. Als Konsequenz ist effektives Culling in großen Szenerien nahezu unerlässlich, da andernfalls Millionen von Polygonen zur Grafikhardware geschickt werden müssten, obwohl im Allgemeinen die meisten von ihnen verdeckt werden.

Aufgrund der extremen Datenmengen ist andererseits aber weder genügend Zeit noch Speicher vorhanden, um die Eingabe vollständig abzuwarten und erst dann komplexe Offline-Algorithmen anzuwenden. Die Vorstellung, Occlusion Culling stets optimal vor-

nehmen zu können, ist dementsprechend unrealistisch. Daher ist die Frage nach Online-Algorithmen interessant, die die Objektmenge sequentiell erhalten und Entscheidungen treffen, ohne den Rest der Eingabe zu kennen.

Im Rahmen dieser Studienarbeit wird eine Modellierung dieser Problematik im zweidimensionalen Fall vorgenommen. Ferner werden für Online- und Offline-Algorithmen Schranken aufgezeigt sowie Überlegungen zur Komplexität des Offline-Problems angestellt.

Bemerkung 1.1: Ohne Einschränkung wird in dieser Arbeit davon ausgegangen, dass Culling das Ergebnis des Renderings nicht beeinflusst. Dies ist insbesondere bei interaktiver Computergrafik, bei der ein hoher Durchsatz entscheidend ist, zumeist richtig. Dennoch gibt es auch Rendering-Verfahren, bei denen Culling das Ergebnis gravierend beeinflussen kann: Etwa bei der *Radiosity*-Methode wird der Energieaustausch zwischen Objekten berechnet, auf den natürlich auch nicht sichtbare Objekte Einfluss haben. Dies ist jedoch nicht Gegenstand dieser Arbeit. Eine Beschreibung von Radiosity gibt beispielsweise Watt (2002, Seite 342 ff.).

2 Mathematische Werkzeuge zur Algorithmen-Analyse

Bevor in den folgenden Abschnitten das formale Modell beschrieben und untersucht wird, werden an dieser Stelle kurz einige wiederkehrende mathematische Grundlagen gegeben sowie die in dieser Arbeit verwendete Notation beschrieben.

Als Konvention wird eingeführt: $\mathbb{N}_0$ bezeichnet die natürlichen Zahlen (mit 0), $\mathbb{N}$ entsprechend ohne. Ferner sei $\mathbb{N}_{\leq n}$ abkürzende Schreibeweise für $\{1, 2, \ldots, n\}$. $\ln(\cdot)$ bezeichnet den natürlichen Logarithmus, $\log(\cdot)$ den Zweierlogarithmus und $\log_k(\cdot)$, wie üblich, den Logarithmus zur Basis k.

2.1 Stochastische Grundlagen

Wie bekannt, lassen sich Wahrscheinlichkeiten eines *Versuches* mathematisch mit *Wahrscheinlichkeitsräumen* modellieren und beschreiben. Ein Wahrscheinlichkeitsraum ist definiert als 3-Tupel $(\Omega, \mathfrak{F}, P)$, wobei Ω Menge von Elementarereignissen, $\mathfrak{F}$ ein Mengensystem auf Ω (die Menge aller Ereignisse) und $P : \mathfrak{F} \to \mathbb{R}$ ein Wahrscheinlichkeitsmaß mit $P(\Omega) = 1$ ist. $(\Omega, \mathfrak{F})$ ist dabei ein messbarer Raum.

Die in dieser Arbeit betrachteten Zufallsgrößen sind (messbare) Abbildungen $\Omega \to \mathbb{R}$. In vielen Fällen interessiert ihr durchschnittlicher Wert, der *Erwartungswert* $\mathbf{E}[\cdot]$. Die wichtigste Eigenschaft des Erwartungswertes ist seine Linearität: Seien X, Y Zufallsgrößen. Dann gilt, selbst wenn X und Y abhängig sind: $\mathbf{E}[a + bX + cY] = a + b\mathbf{E}[X] + c\mathbf{E}[Y]$.

Wie allgemein üblich, wird als Kurzschreibweise für eine Zufallsgröße X eingeführt: $\mathbf{Pr}[X \leq x] := P(X^{-1}(\{z \mid z \leq x\}))$; analog für „<“, „=“, etc.

Ferner werden bedingte Wahrscheinlichkeiten verwendet. $\mathbf{Pr}[A|B]$ ist dabei die Wahrscheinlichkeit des Ereignisses A unter der Voraussetzung, dass das Eintreten von B bereits bekannt ist. Eine anwendungsorientierte Einführung in die Stochastik gibt beispielsweise Hübner (2003).

2.2 Bernoulli-Versuche

Sei $p \in [0,1]$. Eine Zufallsgröße X heißt *Bernoulli(p)-verteilt* (auch *Indikatorzufallsgröße* genannt), wenn $X : \Omega \to \{0,1\}$ und die Wahrscheinlichkeit für einen *Erfolg* $\mathbf{Pr}[X = 1] = p$ ist. Dann folgt offenbar $\mathbf{E}[X] = p$.

Es wird später folgendes Lemma benötigt:

Lemma 2.1: *Gegeben ein Versuch, der sich unter identischen Bedingungen beliebig oft wiederholen lässt, und eine Bernoulli(p)-verteilte Zufallsgröße X mit $p > 0$. Dann gilt, wenn man die Anzahl der Durchführungen bis zum ersten Erfolg als einen neuen Versuch auffasst: Im Erwartungswert sind $\frac{1}{p}$ Durchführungen bis (inklusive) zum ersten Erfolg erforderlich.*

Beweis: Es werden als Zufallsgrößen eingeführt ($i = 1, 2, 3, \ldots$):

$$X_i = \begin{cases} 0 & \text{falls erster Erfolg vor } i\text{-tem Versuch} \\ 1 & \text{falls erster Erfolg bei oder nach } i\text{-tem Versuch} \end{cases}$$

Offensichtlich ist dann $\sum_{i=1}^{\infty} X_i$ die Nummer des Versuchs, bei dem zum ersten Mal ein Erfolg auftritt.

Mit der Linearität des Erwartungswertes erhält man:

$$\mathbf{E}[\sum_{i=1}^{\infty} X_i] = \sum_{i=1}^{\infty} \mathbf{E}[X_i] = \sum_{i=1}^{\infty} (1-p)^{i-1} = \sum_{i=0}^{\infty} (1-p)^i = \frac{1}{1-(1-p)} = \frac{1}{p} \qquad \square$$

Das Lemma ergibt sich ebenfalls, wenn man die Zusammenfassung der Einzelversuche als neuen Versuch mit geometrischer Verteilung auffasst; vergleiche beispielsweise Hübner (2003, Seite 92).

2.3 Totale Wahrscheinlichkeit

Wie im diskreten Fall gibt es auch für stetige Wahrscheinlichkeitsverteilungen eine Formel für die totale Wahrscheinlichkeit. Das uneigentliche Integral sei hier wie üblich durch Grenzwertbildung erklärt: Sind X, Y stetige Zufallsgrößen mit den Dichten f_X und f_Y, dann gilt:

$$f_X(x) = \int_{\mathbb{R}} f_X(x \mid Y = y) f_Y(y) dy \tag{2.2}$$

Einen Beweis gibt beispielsweise Papoulis (1984, Kapitel 7.3).

Satz 2.3: *Seien X, Y stetige Zufallsgrößen mit den Dichten f_X und f_Y. Dann gilt:*

$$F_X(x) = \int_{\mathbb{R}} F_X(x \mid Y = y) f_Y(y) dy$$

Beweis: Nachrechnen ergibt:

$$\begin{aligned} F_X(x) &= \int_{-\infty}^{x} f_X(z)dz \\ &= \int_{-\infty}^{x} \int_{\mathbb{R}} f_X(z \mid Y = y) f_Y(y) dy\, dz \quad \text{nach (2.2)} \\ &= \lim_{a\to\infty} \int_{-a}^{x} \int_{-a}^{a} f_X(z \mid Y = y) f_Y(y) dy\, dz \\ &= \lim_{a\to\infty} \int_{-a}^{a} \int_{-a}^{x} f_X(z \mid Y = y) f_Y(y) dz\, dy \qquad (2.4) \\ &= \int_{\mathbb{R}} \int_{-\infty}^{x} f_X(z \mid Y = y) f_Y(y) dz\, dy = \int_{\mathbb{R}} \int_{-\infty}^{x} f_X(z \mid Y = y) dz\, f_Y(y) dy \\ &= \int_{\mathbb{R}} F_X(x \mid Y = y) f_Y(y) dy \end{aligned}$$

Dabei ist (2.4) nach Satz von Fubini erlaubt: $g(z, y) := f_X(z \mid Y = y) f_Y(y)$ ist als Produkt integrierbarer Funktionen wieder integrierbar. Ferner existieren $\int_{-a}^{x} g(z, y) dz$ für jedes $y \in [-a, a]$ und $\int_{-a}^{a} g(z, y) dy$ für jedes $z \in [-a, x]$. Damit ist die Reihenfolge der Integration über ein kompaktes mehrdimensionales Intervall vertauschbar. Einen Beweis des Satzes von Fubini gibt beispielsweise Heuser (1991, Seite 448 ff.). □

Satz 2.5: *Sei X diskrete Zufallsgröße mit Werten aus $\mathbb{Z}$, Y stetige Zufallsgröße. Dann gilt für $x \in \mathbb{Z}$:*

$$F_X(x) = \int_{\mathbb{R}} F_X(x \mid Y = y) f_Y(y) dy$$

Beweis: Es wird eine neue stetige Zufallsgröße X' wie folgt definiert:

$$f_{X'}(x) := \mathbf{Pr}\left[X = \lceil x \rceil\right] \quad (x \in \mathbb{R})$$

Den Wahrscheinlichkeitsraum stelle man sich so erweitert vor, dass $X = \lceil X' \rceil$. Dann gilt sicherlich: $F_{X'}(x) = F_X(x)$ sowie $F_{X'}(x \mid Y = z) = F_X(x \mid Y = z)$ für $x \in \mathbb{Z}, z \in \mathbb{R}$. Folglich lässt sich Satz 2.3 anwenden und es folgt die Behauptung. □

2.4 Analytische Grundlagen

Lemma 2.6 (Harmonische Zahl): *Die n-te* Harmonische Zahl $H_n := \sum_{i=1}^{n} \frac{1}{i}$ *wird eingegrenzt durch:*

$$\ln n \leq \sum_{i=1}^{n} \frac{1}{i} = H_n \leq \ln n + 1$$

Beweis: Es gilt:

$$\ln n \leq \ln(n+1) = \int_1^{n+1} \frac{1}{x} dx \leq H_n \leq 1 + \int_1^n \frac{1}{x} dx = 1 + \ln n$$

Offensichtlich ist $H_n = \sum_{i=1}^{n} \frac{1}{x}$ Obersumme für das linke Integral, aber $H_n - 1 = \sum_{i=2}^{n} \frac{1}{x}$ ist Untersumme für $\int_1^n \frac{1}{x} dx$. □

Lemma 2.7 (Bernoulli-Ungleichung): *Sei $x \geq -1$ und $n \in \mathbb{N}_0$. Dann: $(1+x)^n \geq 1 + nx$.*

Beweis: Dies ist induktiv mit folgendem Induktionsschritt einzusehen:

$$\begin{aligned}(1+x)^{n+1} &= (1+x)^n \cdot (1+x) \geq (1+nx) \cdot (1+x) = 1 + (n+1)x + nx^2 \\ &\geq 1 + (n+1)x\end{aligned}$$ □

3 Das formale Modell

3.1 Grundbegriffe

Das fertige Pixel-Bild auf dem Monitor, das am Ende der Rendering-Pipeline steht, ist üblicherweise nur eine zweidimensionale Projektion einer dreidimensionalen Szenerie. Bei der Projektion wird in der Computergrafik für gewöhnlich ein an die Fotografie angelehntes Modell verwendet: In einem *Projektionszentrum* laufen alle Sehstrahlen (*Projektionslinien* genannt) zusammen. Projektionslinien sind imaginäre Verbindungsgeraden, die durch einen Eckpunkt eines Objektes und das Projektionszentrum verlaufen. Schließlich existiert eine Bildfläche (zumeist *Bildebene* genannt), deren Schnittpunkte mit den Projektionslinien offenbar die Projektionen von Eckpunkten der in der Szenerie vorhandenen Objekte sind. Bei der Rasterung des Bildes wird ein Ausschnitt der Bildebene zum Bereich des fertigen Pixel-Bilds. Jedes Pixel entspricht also einem kleinen Quadrat auf der Bildebene. Die Schnittpunkte werden nun jeweils dem sie überdeckenden Pixel zugeordnet. Pixel, die nicht von Projektionslinien durchlaufen aber von einem Objekt überdeckt werden, werden per Interpolation gefüllt. Falls ein Pixel durch kein Objekt überdeckt wird, so wird es mit der Hintergrundfarbe gefüllt.

Abbildung 2 zeigt diese Zusammenhänge und die Analogie zur Fotografie: Man kann sich das Projektionszentrum intuitiv als unendlich kleines Loch einer Lochkamera (camera obscura) vorstellen – in der Computergrafik befindet sich die Bildebene jedoch nicht am Ende der Kamera (das heißt hinter dem Projektionszentrum), sondern *davor*. Anders als bei der Lochkamera ist aus diesem Grund die Projektion nicht spiegelverkehrt.

Im Folgenden werden nun die begrifflichen Grundlagen gegeben, die anschließend für die formale Problembeschreibung benötigt werden.

Im Zweidimensionalen wird aus der Projektionsebene eine Projektionslinie. In der Modellierung sei die Projektionslinie dabei der gesamte Einheitskreis, während Projektionszentrum der Ursprung $(0, 0)$ ist. Die zu zeichnenden Objekte seien als eine endliche Folge

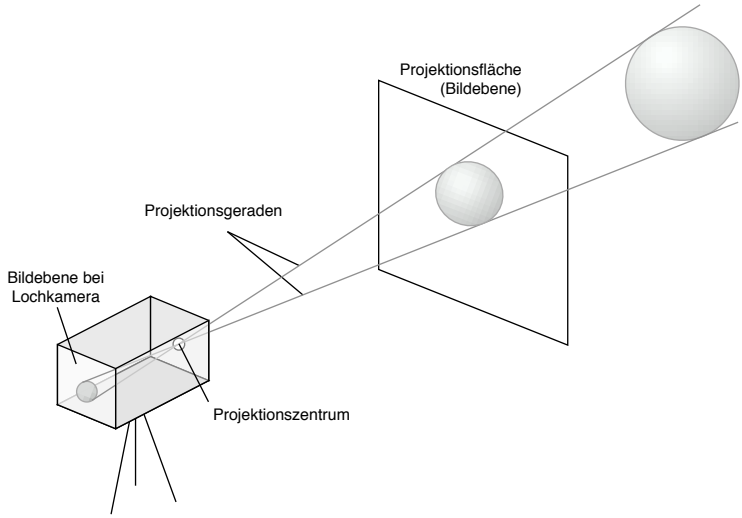


Abbildung 2: Projektionszentrum und -ebene in der Computergrafik mit Analogie zur Fotografie

$V = (k_1, \ldots, k_n)$ von Kugeln k_i mit Radius 1 im $\mathbb{R}^2$ gegeben, wobei die $k_i = (x_i, y_i)$ die Mittelpunkte der Kugeln beschreiben.

Zum einfacheren Rechnen mit Winkeln wird die Menge $\mathfrak{T} := \{\{r + 2\pi\mathbb{Z}\} \mid r \in \mathbb{R}\}$ eingeführt. (Mit Notation wie in der Algebra üblich, könnte man diese Menge als $\mathbb{R}/2\pi\mathbb{Z}$ bezeichnen, wobei die Restklassenbildung bezüglich der Addition zu verstehen ist.)

Oftmals ist das Rechnen mit Polarkoordinaten einfacher:

Definition 3.1: *Distanz-* (rad), *normierte Winkel-* (ang_0), *Winkel-* (ang) und *Projektionswinkel-Abbildung* (pang) werden definiert als:

$$\text{rad} : \mathbb{R}^2 \to \mathbb{R},\ \text{rad}((x,y)) := \|(x,y)\|_2 \quad \text{(euklidische Norm)}$$

$$\text{ang}_0 : \mathbb{R}^2 \to (-\pi, \pi],\ \text{ang}_0((x,y)) := \begin{cases} 0 & \text{wenn } \text{rad}(x,y) = 0 \\ \arccos \frac{x}{\text{rad}(x,y)} & \text{sonst, wenn } y \geq 0 \\ -\arccos \frac{x}{\text{rad}(x,y)} & \text{sonst} \end{cases}$$

$$\text{ang} : \mathbb{R}^2 \to \mathfrak{T},\ \text{ang}((x,y)) := \text{ang}_0((x,y)) + 2\pi\mathbb{Z}$$

$$\text{pang} : \mathbb{R} \to (0, 2\pi],\ \text{pang}((x,y)) := 2 \arcsin \frac{1}{\text{rad}((x,y))}$$

Abbildung 3 illustriert die Definitionen grafisch.

Definition 3.2: Sei $V = (k_1, \ldots, k_n)$ endliche Folge von Kugeln. Ferner seien $i, j \in \mathbb{N}_{\leq n}$ mit $i \neq j$. Dann gilt: k_i *verdeckt indirekt* k_j ($\Leftrightarrow$: $k_i \gg k_j$), wenn:

$$\text{ang}(k_i) + [-\frac{\text{pang}(k_i)}{2}, \frac{\text{pang}(k_i)}{2}] \supsetneq \text{ang}(k_j) + [-\frac{\text{pang}(k_j)}{2}, \frac{\text{pang}(k_j)}{2}]$$

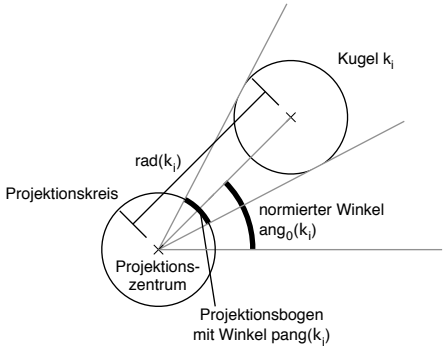


Abbildung 3: Projektionszentrum, -kreis und -winkel im zweidimensionalen Modell

Die Addition ist hier, wie üblich, elementweise zu verstehen. Gibt es zusätzlich *kein* $l \in \mathbb{N}_{\leq n}$, so dass $k_i \gg k_l$ und $k_l \gg k_j$, dann *verdeckt* k_i *das Objekt* k_j *direkt* ($\Leftrightarrow$: $k_i > k_j$).

Bemerkung 3.3: Die Definition entspricht einer „intuitiven“ Definition von Verdecken: Eine Kugel verdeckt eine andere, wenn vom Projektionszentrum aus die andere Kugel nicht sichtbar ist. Ferner existieren folgende Zusammenhänge für die Relationen:

i) Verdeckt Kugel k_i Kugel k_j indirekt, also $k_i \gg k_j$, so folgt daraus offenbar $\text{pang}(k_i) > \text{pang}(k_j)$. Der Projektionswinkel einer Kugel fällt aber streng monoton mit der Entfernung zum Projektionszentrum. Daher folgt aus $k_i \gg k_j$ auch $\text{rad}(k_i) < \text{rad}(k_j)$.

ii) Die Relation „verdeckt indirekt“ ist natürlich die transitive Hülle der Relation „verdeckt direkt“.

Abbildung 4 zeigt grafisch die Verdeckungszusammenhänge aus Definition 3.2 für eine endliche Folge von Kugeln. Eine gerichtete Kante in dem dargestellten Graphen repräsentiert jeweils eine indirekte Verdeckung. Man beachte, dass es sich bei dem Graphen nicht zwangsläufig um einen Baum handeln muss. In Abbildung 4 ist dies zum Beispiel an den Kugeln k_3, k_9 und k_{11} zu sehen. Nach Bemerkung 3.3 muss der Graph aber azyklisch, also kreisfrei sein.

Offensichtlich induziert jede endliche Folge V von Kugeln zusammen mit der Relation „verdeckt indirekt“ einen azyklischen Graphen:

Definition 3.4: Sei $V = (k_1, \ldots, k_n)$ eine endliche Folge von Kugeln im $\mathbb{R}^2$. 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 der endlichen Folge von Kugeln V.

Für den Culling-Algorithmus sei nun angenommen, dass er nur Objekte samt ihrer Verdeckungsbeziehungen betrachten kann, die er bereits als Eingabe erhalten hat. Er sieht also bis wenigstens zum letzten Objekt nur einen beschränkten Ausschnitt aus dem Verdeckungsgraphen. Insbesondere heißt dies, dass neu ankommende Kugeln nur anhand von

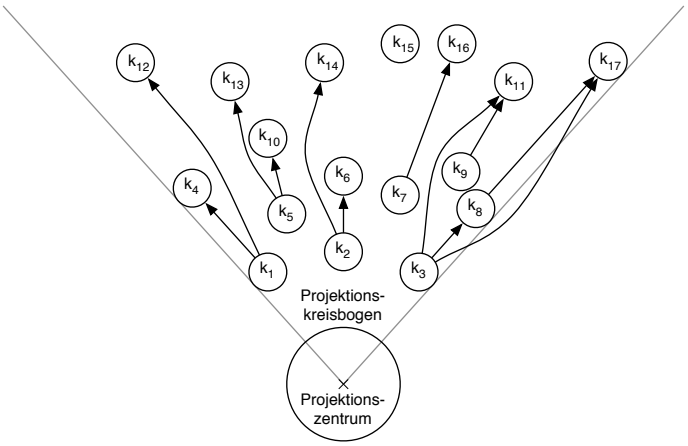


Abbildung 4: Objekte und indirekte Verdeckungen

Objekten im Zwischenspeicher auf Sichtbarkeit untersucht werden können. Kann eine Verdeckung nicht anhand von Objekten im Zwischenspeicher festgestellt werden, so muss die neue Kugel in jedem Fall gezeichnet werden.

Für Culling-Algorithmen soll folgendes Kostenmaß gelten: Jedes Objekt k_i, das an die Grafikhardware gesendet wird, verursache Kosten 1. Das Aufnehmen eines Objektes in den Puffer und Vergleiche von Objekten im Puffer sind kostenlos. Wie im Dreidimensionalen haben verdeckte Objekte keinen Einfluss auf das Ergebnis des Rendering. Es stellt sich also die Frage nach einer geschickten Ausnutzung des Zwischenspeichers, so dass die Anzahl der gezeichneten Kugeln minimiert wird.

3.2 Das Optimierungsproblem

Mit den bisherigen Erklärungen lässt sich das Modell des in dieser Arbeit betrachteten Culling-Problems nun formalisieren:

Definition 3.5: Gegeben folgende Eingabe:

- Endliche Folge von Kugeln $(k_1, \ldots, k_n)$ aus dem $\mathbb{R}^2$.
- Rendering-Reihenfolge: Permutation $\sigma : \mathbb{N}_{\leq n} \to \mathbb{N}_{\leq n}$.
- Größe des Zwischenspeichers c.

Das (Offline) *Culling-Problem* besteht darin, für die Sequenz von Kugeln $k_{\sigma(1)}, \ldots, k_{\sigma(n)}$ eine Verdrängungsreihenfolge aus dem Zwischenspeicher anzugeben, so dass die Gesamtanzahl gezeichneter Kugeln minimiert wird.

Beim zugehörigen Online-Problem kennt der Algorithmus vorab nur die Anzahl der Kugeln n. Die Kugelfolge erhält er sequentiell erst während seine Ablaufs.

Bemerkung 3.6: Das Culling-Problem kann auf den ersten Blick an das Paging-Problem erinnern, bei dem ebenfalls ein Zwischenspeicher verwendet wird. Im nächsten Abschnitt werden sich jedoch große Unterschiede herausstellen, die dazu führen, dass bekannte Ergebnisse zu Online-Algorithmen für das Paging-Problem nicht auf das Culling-Problem übertragen werden können.

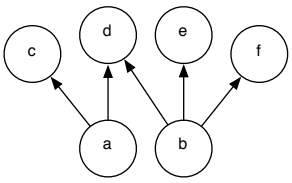
Dieses Problem lässt sich etwas weiter abstrahieren, so dass es sich als reines Graphen-Problem formulieren lässt.

Definition 3.7: Gegeben folgende Eingabe:

- Gerichteter azyklischer Graph $G = (V, E)$, $n := |V|$
- Reihenfolge der Knoten, dargestellt als bijektive Abbildung $\rho : V \to \mathbb{N}_{\leq n}$, die für jeden Knoten seine Position in der Reihenfolge angibt.
- Größe des Zwischenspeichers c.

Das (Offline) Problem *Knotenaussortierung* besteht darin, eine Verdrängungsreihenfolge aus dem Zwischenspeicher anzugeben, so dass die Gesamtanzahl aussortierter Knoten maximiert wird. Ein Knoten v kann dabei aussortiert werden, wenn es einen Knoten v' gibt, so dass $\rho(v') < \rho(v)$, es einen Pfad von v' zu v gibt und v' zum Zeitpunkt der Betrachtung von v noch im Zwischenspeicher enthalten ist.

Beispiel 3.8: Betrachte Abbildung 5.



Reihenfolge:
a, b, c, e, d, f
Zwischenspeicher:
Größe 1

Abbildung 5: Beispiel für eine Instanz des Problems Kontenaussortierung

Es lassen sich nur Knoten aussortieren, die eine eingehende Kante haben. Umgekehrt helfen nur die Knoten bei der Aussortierung weiter, die eine ausgehende Kante haben. Man sieht, dass b drei Knoten verdeckt, a aber nur zwei. Eine optimale Verdrängungssequenz ist also $(\varepsilon, a, c, e, d, f)$. Hierbei bezeichnet ε das Überschreiben einer nicht initialisierten Stelle des Zwischenspeichers – in diesem Fall mit a. Anschließend wird a durch b mit der gegebenen Begründung verdrängt. Die restlichen Knoten c, e, d und f werden gar nicht erst in den Zwischenspeicher aufgenommen. Insgesamt können mit Zwischenspeicher der Größe 1 also drei Knoten aussortiert werden. ◊

Bemerkung 3.9: Die beiden Probleme sehen zwar sehr ähnlich aus, allerdings ist Knotenaussortierung mächtiger: Es lassen sich gerichtete azyklische Graphen konstruieren, zu denen es keine entsprechende Anordnung von Kugeln gibt, so dass der Graph Verdeckungsgraph dieser Anordnung wäre.

3.3 Bewertung von Online-Algorithmen

Eine recht nahe liegende Herangehensweise zur Bewertung eines Online-Algorithmus ist der Vergleich zu einem optimalen Offline-Algorithmus, wobei beide Algorithmen die gleiche Eingabe erhalten. Insbesondere interessiert, ob es bei beliebiger Eingabe eine obere Schranke für dieses Verhältnis gibt. Erstmals vorgeschlagen und an klassischen Problemen wie dem Listenzugriff und Paging durchgeführt wurde dieses Vorgehen von Sleator u. Tarjan (1985). Später prägten Karlin u. a. (1988, Seite 81) den Begriff der *Competitive Analysis* und es wurde folgende Definition üblich.

Ein deterministischer Online-Algorithmus ALG wird oft nach seinem *Competitive Ratio* bewertet, das wie folgt definiert ist:

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

Dabei bezeichnet $\mathcal{I}$ die Menge aller gültigen Eingaben, ALG$[I]$ die von ALG bei Eingabe I verursachten Kosten und OPT$[I]$ die Kosten, die ein optimaler Algorithmus erzeugt.

Oft interessiert nur eine obere Schranke c für das Competitive Ratio. Kann ein solches c angegeben werden, so heißt der Online-Algorithmus *c-competitive*.

Bemerkung 3.10: Die Competitive Analysis wird oft mit einem *(grausamen) Gegenspieler* erklärt: Er darf zu einem Online-Algorithmus – bei voller Kenntnis seiner Funktionsweise – eine beliebige zulässige Eingabe auswählen, so dass der Online-Algorithmus bei eben dieser Eingabe möglichst hohe und der Offline-Algorithmus möglichst niedrige Kosten hat.

Man stellt fest, dass dieses Bewertungsmaß bei der vorgeschlagenen Modellierung des Cullings wenig sinnvoll ist, da für jeden Online-Algorithmus das Competitive Ratio $\Theta(n)$ sein würde (wobei n die maximale Anzahl von Kugeln bezeichnet). Folgendes kleine Beispiel, in Abbildung 6 dargestellt, soll dies demonstrieren:

Sei ALG ein beliebiger deterministischer Online-Algorithmus für das Culling-Problem. Ferner gebe es einen Zwischenspeicher der Größe c, der für das Culling verwendet werden kann. Werden nun die Kugeln $k_1, k_2, \ldots, k_{c+1}$ an ALG geschickt, wie in der Abbildung zu sehen, so kann spätestens nach k_{c+1} wenigstens eine der Kugeln $k_1, \ldots, k_{c+1}$ nicht mehr im Zwischenspeicher vorhanden sein. Ohne Beschränkung der Allgemeinheit sei dies Kugel k_3. Würden nun die restlichen Kugeln $k_{c+2}, \ldots, k_n$ in eben dieser Reihenfolge an den Culling-Algorithmus geschickt, so müsste jede Kugel gezeichnet werden.

Ein optimaler Offline-Algorithmus hätte natürlich eine andere als Kugel k_3 aus dem Zwischenspeicher entfernt und so nur insgesamt $c + 1$ Kugeln an die Hardware geschickt.

Die beschriebene Eingabesequenz sei mit I_n bezeichnet. Wegen

$$\frac{\text{ALG}[I_n]}{\text{OPT}[I_n]} = \frac{n}{c+1} = \Theta(n)$$

folgt dann die obige Behauptung, dass ALG nur $\Theta(n)$-competitive beziehungsweise das Competitive Ratio $\Theta(n)$ ist. (Mit anderer Sprechweise ist ALG *nicht competitive*, da das Verhältnis mit wachsendem n beliebig schlecht werden kann.)

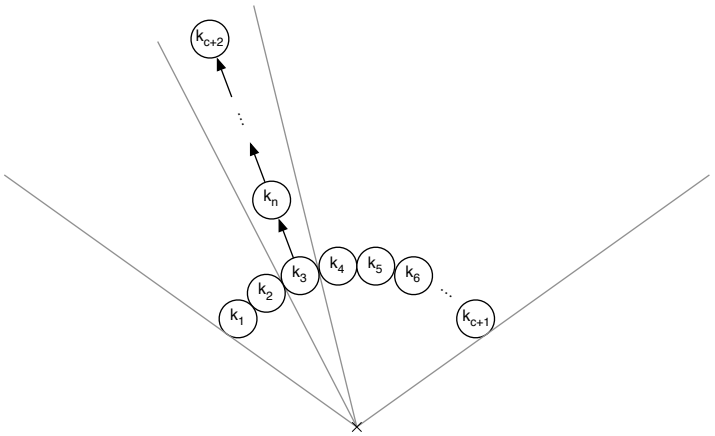


Abbildung 6: Direkte Verdeckungen einer worst-case Eingabe für einen Online-Algorithmus

3.3.1 Randomisierung

Wie gesehen, ist jeder deterministische Online-Algorithmus beliebig schlecht bezüglich der Competitive Analysis. Zu klären bleibt jedoch noch die Frage, ob diese Aussage auch auf randomisierte Online-Algorithmen zutrifft. Aufgrund der zufälligen Entscheidungen randomisierter Algorithmen sind die verursachten Kosten bei einer Eingabe eine Zufallsgröße. Das Competitive Ratio eines randomisierten Online-Algorithmus ALG definiert man daher analog als:

$$\inf\{c \mid \mathbf{E}[\text{ALG}[I]] \leq c \cdot \text{OPT}(I) + \alpha \quad \text{für alle } I \in \mathcal{I}\} \tag{3.11}$$

Eine Einführung in randomisierte Online-Algorithmen geben beispielsweise Borodin u. El-Yaniv (1998), Kapitel 2.

Man betrachte wieder eine Anordnung wie in Abbildung 6 dargestellt. Nun gilt sicherlich nach dem Verarbeiten von Kugel k_{c+1} durch den Culling-Algorithmus:

$$\sum_{i=1}^{c+1} \mathbf{Pr}[\text{Kugel } k_i \text{ nicht im Zwischenspeicher}] = 1,$$

da die Größe des Zwischenspeichers nur c beträgt und der Erwartungswert für die Anzahl sich nicht im Zwischenspeicher befindlicher Kugeln wenigstens 1 ist. Folglich muss es ein $i \in \mathbb{N}_{\leq c+1}$ geben mit $p := \mathbf{Pr}[\text{Kugel } k_i \text{ nicht im Zwischenspeicher}] \geq \frac{1}{c+1}$. In der Abbildung sei ohne Beschränkung der Allgemeinheit $i = 3$. Somit folgt für jeden randomisierten Online-Algorithmus ALG:

$$\frac{\mathbf{E}[\text{ALG}[I_n]]}{\text{OPT}[I_n]} = \frac{c + 1 + p(n - c - 1)}{c + 1} = \Theta(n)$$

ALG ist also nur $\Theta(n)$-competitive.

3.3.2 Ressourcen-Vergrößerung

Ein weiteres Konzept zur Untersuchung von Online-Algorithmen besteht darin, bei der Competitive Analysis Online- und Offline-Algorithmus mit unterschiedlichen Ressourcen auszustatten. Im Falle des deterministischen Paging etwa zeigten Sleator u. Tarjan (1985), dass die Online-Algorithmen LRU oder FIFO mit Speicher der Größe n maximal das c-fache an Kosten eines optimalen Offline-Algorithmus mit Speicher $(1 - \frac{1}{c})n$ verursachen ($c \geq 1$). Je nach Vergrößerung der Ressourcen werden LRU oder FIFO also beliebig competitive, während bei gleichen Ressourcen das Competitive Ratio die Größe des Zwischenspeichers war.

Angewandt auf das Culling-Problem kann jedoch auch mit Ressourcen-Vergrößerung keine Verbesserung bezüglich der Competitive Analysis erzielt werden: Als Beispiel dient wieder Abbildung 6. Auch hier existiert für beliebig große c eine Kugel k_i, $i \in \mathbb{N}_{c+1}$, die nach dem Verarbeiten von Kugel k_{c+1} mit einer Wahrscheinlichkeit von $\geq \frac{1}{c+1}$ (falls der Online-Algorithmus deterministisch ist, sogar mit Wahrscheinlichkeit 1) nicht im Zwischenspeicher enthalten ist. Folglich gilt für die beschriebenen Eingabesequenzen immer: $\mathbf{E}[\text{ALG}[I_n]] = \Theta(n)$.

In einem Lemma zusammengefasst:

Lemma 3.12: *Beim Betrachten einzelner Eingabesequenzen kann das Kostenverhältnis von Online-Algorithmus zu Offline-Algorithmus beliebig schlecht werden – selbst dann, wenn der Offline-Algorithmus nur einen konstanten Zwischenspeicher der Größe 1 hat.*

Beweis: Wie zuvor beschrieben, genügt dazu das Betrachten einer Eingabesequenz wie folgt: Hinter einer Kugel k_i, die mit Wahrscheinlichkeit $c > 0$ aus dem Zwischenspeicher entfernt wird, folgen nur noch (in nach absteigender Entfernung vom Projektionszentrum sortierter Reihenfolge) n viele Kugeln, die von Kugel k_i indirekt verdeckt werden. Dabei ist c eine von n unabhängige Konstante. □

3.3.3 Erweiterung der Competitive Analysis

In den vorangegangenen Abschnitten wurde gezeigt, dass die ausschließliche Betrachtung des worst-case Verhältnisses für eine einzelne Eingabe, wie bei der Competitive Analysis gemacht, für das Culling-Problem unbefriedigend ist: Einerseits ist intuitiv leicht einsehbar, dass es in der Praxis gut funktionierende Heuristiken gibt, andererseits lässt sich – nach Lemma 3.12 – die Qualität von Online-Algorithmen nicht mit der Competitive Analysis messen.

In der Tat ist dieses Problem ein oft angeführter Kritikpunkt der Competitive Analysis, das auch bei der Analyse anderer bekannter Probleme zu übermäßig pessimistischen Ergebnissen führt. Koutsoupias u. Papadimitriou (2000, Seite 300 f.) illustrieren diese Kritik anhand des bekannten Paging-Problems, bei dem bezüglich der Competitive Analysis der in der Praxis bewährte LRU-Algorithmus nicht über den empirisch nur mäßig gut funktionierenden FIFO-Algorithmus hinauskommt.

Insbesondere betrachtet die Competitive Analysis nur das worst-case Verhältnis zwischen Online- und Offline-Algorithmus und ignoriert dabei, dass üblicherweise über die Verteilung

der Eingaben durchaus einiges bekannt ist. Koutsoupias u. Papadimitriou (2000) schlagen daher vor, für deterministische Online-Algorithmen die Definition des Competitive Ratio wie folgt zu erweitern:

$$\inf\{c \mid \mathbf{E}_D[\text{ALG}[I]] \leq c \cdot \mathbf{E}_D[\text{OPT}(I)] + \alpha \quad \text{für alle } D \in \Delta\} \tag{3.13}$$

Dabei ist Δ eine vorgegebene Klasse möglicher Verteilungen der Eingaben. Der Gegenspieler kann nur noch *eine Verteilung* D aus Δ wählen, so dass die *erwarteten Kosten* von Online- und Offline-Algorithmus – bei gemäß D zufälliger Wahl einer Eingabe – möglichst weit auseinander gehen. I ist als Zufallsgröße mit Verteilung D aufzufassen.

Offensichtlich ist diese Definition eine echte Erweiterung: (3.13) und (3.11) stimmen überein, wenn Δ nur jene Verteilungen umfasst, bei der mit Wahrscheinlichkeit 1 eine worst-case Eingabe ausgewählt wird.

In den folgenden Abschnitten wird aus den erwähnten Gründen diese Modifikation angewandt, wobei Δ nur die Gleichverteilung aller Eingaben enthält. Ferner wird sich in dieser Arbeit bei der Analyse auf den Quotienten der Kosten von Online- zu optimalem Offline-Algorithmus beschränkt. Anders formuliert, es wird vorgeschlagen, statt des Competitive Ratio (3.11) eines Online-Algorithmus ALG den Quotienten

$$\frac{\mathbf{E}_D[\text{ALG}[I]]}{\mathbf{E}_D[\text{OPT}[I]]} \tag{3.14}$$

zu betrachten, wobei D die Gleichverteilung über alle Anordnungen von Kugeln (in einem begrenzten Radius um das Projektionszentrum) und zufälligen Reihenfolgen ist. Dieser Quotient ist asymptotisch natürlich eine obere Schranke für das erwartete Competitive Ratio (3.13).

Aufgrund der Eindeutigkeit wird im Folgenden nur noch $\mathbf{E}[\cdot]$ statt $\mathbf{E}_D[\cdot]$ geschrieben.

4 Schranken im Modell mit unbeschränktem Zwischenspeicher

Es wird zunächst die Frage geklärt, mit welcher Wahrscheinlichkeit eine beliebige Kugel k_i einer gegebenen und festen Anordnung gezeichnet werden muss. Vorausgesetzt:

- Der Culling-Algorithmus erhält die Objekte in zufälliger Reihenfolge und
- die Puffergröße ist unbegrenzt.

Es ist zu bemerken, dass in diesem Fall der Offline-Algorithmus keinen Vorteil gegenüber einem Online-Algorithmus hat, da aufgrund des unbegrenzten Zwischenspeichers überhaupt keine Entscheidungen getroffen werden müssen. Trivialerweise ist sogar jeder Algorithmus optimal, der jede gezeichnete Kugel im Zwischenspeicher hält.

Es wird für eine Kugel k_i aus $V = (k_1, \ldots, k_n)$ die (Indikator-)Zufallsgröße K_i eingeführt. Es soll gelten $K_i = 1$, falls die Kugel k_i gezeichnet werden muss und $K_i = 0$, wenn nicht. Da K_i Indikatorzufallsgröße ist, gilt natürlich

$$\mathbf{E}[K_i] = \mathbf{Pr}[K_i = 1].$$

Aufgrund der Linearität des Erwartungswertes ist dann

$$\mathbf{E}[\sum_{i=1}^{n} K_i] = \sum_{i=1}^{n} \mathbf{E}[K_i]$$

gleich dem Erwartungswert für die Anzahl gezeichneter Kugeln insgesamt.

Für K_i gilt: $K_i = 1$, falls noch keine der j vor k_i liegenden Kugeln gezeichnet worden ist, wenn k_i selbst vom Culling-Algorithmus verarbeitet wird – andernfalls $K_i = 0$. Die Reihenfolge der Kugeln sei durch die Permutation $\sigma : \mathbb{N}_{\leq n} \to \mathbb{N}_{\leq n}$ beschrieben.

Die Menge der Indizes der j vor k_i liegenden Kugeln sei $\{i_1, \ldots, i_j\}$. Man stellt fest, dass $K_i = 1$ genau dann, wenn $\sigma(i) = \min \sigma(\{i_1, \ldots, i_j, i\})$. Nur in diesem Fall wird k_i vor seinen räumlichen Vorgängern an den Rendering-Algorithmus geschickt. Folglich, da σ zufällig gleichverteilt aus der Menge der Permutation stammt:

$$\mathbf{Pr}[K_i = 1] = \frac{1}{j+1}$$

In einem Lemma zusammengefasst:

Lemma 4.1: *Sei k_i eine Kugel, die von genau j anderen Kugeln indirekt verdeckt wird. Dann zeichnet ein optimaler Culling-Algorithmus mit unbegrenztem Zwischenspeicher die Kugel k_i mit Wahrscheinlichkeit genau $\frac{1}{j+1}$.*

4.1 Spezielle Anordnungen

Beispiel 4.2: Es gebe n Objekte $k_1, \ldots, k_n$, die allesamt auf einer Projektionslinie liegen: Je ein Objekt verdeckt direkt genau ein weiteres.

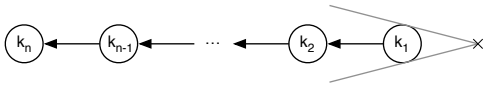


Abbildung 7: Anordnung von Objekten auf einer Projektionslinie

Lemma 4.1 ergibt:

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{E}[K_i] &= \sum_{i=1}^{n} \frac{1}{i} = H_n \\ &= \Theta(\log n) \qquad \text{nach Lemma 2.6}\end{aligned}$$ ◊

Beispiel 4.3: Es gebe $n := 2^m - 1$ Objekte, die so angeordnet sind, dass ein Objekt jeweils zwei andere direkt verdeckt. Dabei gebe es ein einziges Objekt, das alle anderen indirekt verdeckt und 2^{m-1} Objekte, die kein anderes Objekt verdecken. Folglich ergibt

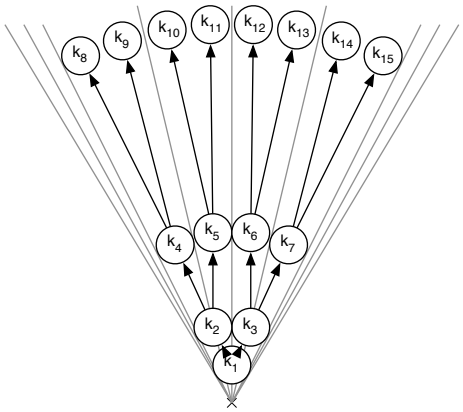


Abbildung 8: Anordnung mit zur Entfernung exponentiell zunehmender Anzahl verdeckter Objekte

sich eine Anordnung als vollständiger binärer Baum, wie in Abbildung 8 (dort für $m = 4$) gezeigt.

Man erhält wieder nach Lemma 4.1:

$$\sum_{i=1}^{n} \mathbf{E}[K_i] = \sum_{i=1}^{n} \frac{1}{\lfloor \log i \rfloor + 1} = \sum_{i=0}^{m-1} \frac{2^i}{i+1}$$
$$= \Omega(\frac{1}{m} \sum_{i=0}^{m-1} 2^i) = \Omega(\frac{2^m - 1}{m}) = \Omega(\frac{n}{\log n}) \qquad \Diamond$$

4.2 Der allgemeine Fall

Korollar 4.4: *Sei $V = (k_1, \ldots, k_n)$ eine beliebige Anordnung von Kugeln und $\mu_i := 1 + |\{k_j \mid k_j \gg k_i\}|$. Dann hat ein optimaler Culling-Algorithmus mit unbeschränktem Zwischenspeicher erwartete Kosten von genau*

$$\sum_{i=1}^{n} \frac{1}{\mu_i}$$

Beweis: Dies folgt sofort aus Lemma 4.1. □

4.3 Der zufällig gleichverteilte Fall

Im Folgenden wird nun überlegt, welche Aussage sich für die erwartete Anzahl zu zeichnender Kugeln treffen lässt, wenn

- für die Kugeln $k_1, \ldots, k_n$ gilt: $\mathrm{rad}(k_i) \in (1, r]$,

- alle Kugeln zufällig gleichverteilt angeordnet sind und
- die Reihenfolge der n Kugeln zufällig gleichverteilt ist.

Die Bedingung $\text{rad}(k_i) > 1$ existiert, da andernfalls eine Kugel auf dem Projektionszentrum liegen kann.

Es wird folgende Definition benötigt:

Definition 4.5: Seien $a, b > 0$. Dann definiere:

$$[a,b]_\bullet := \{x \in \mathbb{R}^2 \mid \|x\|_2 \in [a,b]\}$$

Analog dazu seien $(a,b]_\bullet$, $[a,b)_\bullet$ und $(a,b)_\bullet$ definiert.

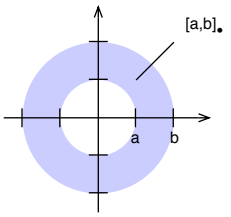


Abbildung 9: Anschauliche Darstellung von Definition 4.5

Ein „Intervall“ $[a,b]_\bullet$ repräsentiert eine Fläche, deren Punkte einen Abstand vom Projektionszentrum im Intervall $[a,b]$ haben, wie in Abbildung 9 dargestellt. Es ist offensichtlich, dass für die Größe dieser Fläche gilt: $\int_{\mathbb{R}} \int_{\mathbb{R}} \mathbb{1}_{[a,b]_\bullet}(x,y)dx\,dy = \pi(b^2 - a^2)$.

Daraus folgt: Ist R eine Zufallsgröße für den Abstand einer beliebigen Kugel vom Projektionszentrum, so ist R stetig und die Verteilungsfunktion F_R von R hat die Form:

$$F_R(x) = \begin{cases} 0 & \text{für } x \leq 1 \\ \frac{\pi(x^2-1)}{\pi(r^2-1)} = \frac{x^2-1}{r^2-1} & \text{für } 1 < x < r \\ 1 & \text{für } x \geq r \end{cases} \tag{4.6}$$

Dementsprechend folgt für die Dichte f_R als Ableitung von F_R:

$$f_R(x) = \begin{cases} 0 & \text{für } x \notin (1,r] \\ \frac{2x}{r^2-1} & \text{für } x \in (1,r] \end{cases}$$

Im Folgenden sei stets $F := F_R$ für eine beliebige, aber nicht näher bestimmte Zufallsgröße R, wie oben beschrieben.

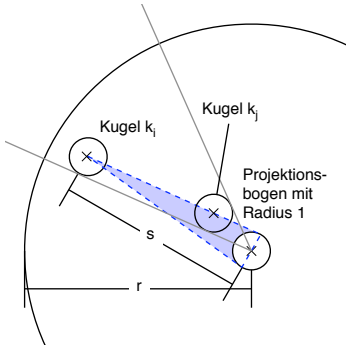


Abbildung 10: Bereich, in dem eine Kugel k_j liegen muss, um k_i zu verdecken

4.3.1 Exakte Formel

Es wird die Zufallsgröße $T_{i,j} \in \{0,1\}$ eingeführt, für die gelten soll: $T_{i,j} = 1 :\Leftrightarrow k_j \gg k_i$.

In Abbildung 10 ist farbig markiert, in welchem Bereich eine Kugel k_j liegen muss, damit sie die Kugel k_i indirekt verdeckt. Man stellt leicht fest, dass für eine Kugel k_i mit $s := \text{rad}(k_i)$ gilt: Diese Fläche hat die Größe

$$\frac{\pi}{2} - 2\arctan(s) - \sin(2 \cdot \arctan(s)) < 2s.$$

Eine Abschätzung nach oben wäre etwa durch das gestrichelte Dreieck mit Höhe s und Grundseite 2 möglich.

Die Wahrscheinlichkeit für $T_{i,j} = 1$ ergibt sich als Quotient aus der Größe der farbig markierten Fläche durch die gesamte Kreisfläche (abzüglich Projektionskreis):

$$(0,1) \ni \mathbf{Pr}[T_{i,j} = 1 \mid R = s] < \frac{2s}{\pi(r^2 - 1)} \tag{4.7}$$

Da in (4.7), abgesehen von zuvor definierten Konstanten, einzig eine Abhängigkeit von s vorliegt, wird zur Vereinfachung im Folgenden die Festlegung $p(s) := \mathbf{Pr}[T_{i,j} = 1 \mid R = s]$ verwendet.

Es wird eine neue Zufallsgröße J_i eingeführt, die die Anzahl der Kugeln angibt, welche k_i indirekt verdecken. Es folgt dann mit der Formel der totalen Wahrscheinlichkeit:

$$\begin{aligned}
\mathbf{Pr}[K_i = 1 \mid R = s] &= \sum_{j=0}^{n-1} \mathbf{Pr}[(K_i = 1 \mid J_i = j) \mid R = s] \cdot \mathbf{Pr}[J_i = j \mid R = s] \\
&= \sum_{j=0}^{n-1} \frac{1}{1+j} \cdot p(s)^j \cdot (1 - p(s))^{n-1-j} \cdot \binom{n-1}{j} \\
&= \frac{1}{p(s)} \sum_{j=0}^{n-1} \underbrace{\frac{1}{1+j} \cdot p(s)^{j+1}}_{= \int_0^{p(s)} t^j \, dt} \cdot (1 - p(s))^{n-1-j} \cdot \binom{n-1}{j}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{p(s)} \int_0^{p(s)} \sum_{j=0}^{n-1} t^j \cdot (1-p(s))^{n-1-j} \cdot \binom{n-1}{j} dt \\
&= \frac{1}{p(s)} \int_0^{p(s)} (t + (1-p(s)))^{n-1} dt \\
&= \frac{1}{p(s)} \Big[\frac{(t + (1-p(s)))^n}{n} \Big]_0^{p(s)} = \frac{1-(1-p(s))^n}{p(s)n} \in (0,1] \qquad (4.8)
\end{aligned}
$$

Bemerkung 4.9: Die Intervalleingrenzung aus (4.8) (die bei korrekter Herleitung natürlich zwangsläufig erfüllt sein muss), folgt auch rechnerisch aus der Bernoulli-Ungleichung (die untere Schranke gilt offensichtlich bereits wegen $0 < p(s) < 1$):

$$\frac{1-(1-p(s))^n}{p(s)n} > 1 \Leftrightarrow 1 + (-p(s))n > (1 + (-p(s)))^n \quad \lightning$$

Unter Ausnutzung von Satz 2.5 und $\mathbf{Pr}[K_i = 1] = F_{K_i}(1) - F_{K_i}(0)$ erhält man für den Erwartungswert von K_i:

$$
\begin{aligned}
\mathbf{E}[K_i] &= \mathbf{Pr}[K_i = 1] = \int_1^r \mathbf{Pr}[K_i = 1 \mid R = s] \cdot f_R(s) ds \qquad \text{nach Satz 2.5} \\
&= \int_1^r \frac{2s}{r^2-1} \cdot \frac{1-(1-p(s))^n}{p(s)n} ds \\
&= \frac{2}{(r^2-1)n} \cdot \int_1^r \frac{s}{p(s)} \big(1-(1-p(s))^n\big) ds \qquad (4.10)
\end{aligned}
$$

Durch Multiplikation mit n ist also eine exakte Formel für die erwartete Anzahl zu zeichnender Kugeln gegeben. Die Formel ist allerdings nicht gut für Abschätzungen geeignet. Aufgrund des Potenzierens von $(1-p(s))$ mit n führt ein Abschätzen der „komplizierten“ Funktion $p(s)$ zu großen Ungenauigkeiten: Sowohl Einsetzen von (4.7) zur Abschätzung nach unten:

$$\mathbf{E}[K_i] \geq \frac{2}{(r^2-1)n} \cdot \int_1^r \frac{s}{\frac{2s}{\pi(r^2-1)}} \big(1-(1-p(s))^n\big) ds = \frac{\pi}{n}(r - 1 - \int_1^r (1-p(s))^n ds)$$

als auch Vereinfachungen zur Abschätzung nach oben:

$$\mathbf{E}[K_i] \leq \frac{2}{(r^2-1)n} \cdot \int_1^r sn \; ds = \frac{2}{r^2-1} \cdot \frac{r^2-1}{2} = 1 \qquad \text{nach Bernoulli: Lemma 2.7}$$

liefern nur triviale Schranken.

4.4 Approximation für den zufällig gleichverteilten Fall

Im Folgenden wird durch einen anderen Ansatz versucht, bessere Schranken herzuleiten. Dazu stelle man sich die Fläche innerhalb eines (noch zu bestimmenden) Kranzes um den Projektionsradius in m kongruente Bereiche zerlegt vor. Ist nun im Zwischenspeicher eine Kugel aus jedem dieser Bereiche enthalten, so ist dies hinreichend dafür, dass außerhalb eines gewissen Radius keine Kugel mehr gezeichnet werden muss. Dies ist in Abbildung 11 illustriert.

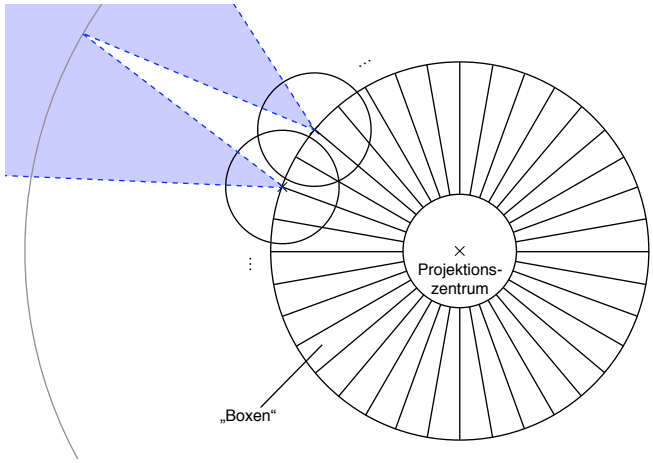


Abbildung 11: Mindestabstand vom Projektionszentrum, ab dem jede Kugel verdeckt wird

4.4.1 Das Balls into Bins-Modell

Die kongruenten Teile des Kranzes seien fortan nur als Boxen bezeichnet. Es ist klar, dass für jede Kugel die Wahrscheinlichkeit, in einer bestimmte Box zu liegen, für alle Boxen gleich groß ist. Es wird daher zunächst folgendes Szenario betrachtet:

Definition 4.11: Es gebe m Boxen und eine (potentiell) unbegrenzte Anzahl von Kugeln, die der Reihe nach auf die m Boxen geworfen werden. Mit jedem Wurf wird genau eine Box getroffen, und zwar jeweils unabhängig, zufällig und gleichverteilt. Dieses Szenario werde als (unendliches) *Balls into Bins*-Modell bezeichnet. Beim endlichen Modell gibt es natürlich nur endlich viele Kugeln.

Bemerkung 4.12: Dieses Modell findet in der Informatik eine vielfältige Anwendung. Raab u. Steger (1998) geben für das endliche Modell einige Beispiele an: So lässt sich etwa das Hashing-Problem auf diese Art modellieren, ebenso wie Online Load Balancing, bei dem ohne zentrale Instanz Jobs an eine jeweils zufällig ausgewählte Maschine eines Maschinen-Pools vergeben werden. Haben alle Anfragen die gleiche Größe, so entspricht die maximale Last einer Maschine genau der Anzahl Kugeln in einer Box.

Hier interessiert beim Balls into Bins-Modell, wie viele Würfe benötigt werden, damit in jeder Box eine Kugel enthalten ist.

Satz 4.13: *Beim (unendlichen) Balls into Bins-Modell mit m Boxen sind im Erwartungswert $mH_m = \Theta(m \ln m)$ Kugeln zu werfen, bevor alle Boxen wenigstens eine Kugel enthalten.*

Beweis: Es sei X_i $(i = 0, 1, 2, \dots)$ die Zufallsgröße, die angibt, wie viele Kugeln geworfen wurden, nachdem genau i Boxen gefüllt waren, aber bevor die $(i+1)$-te Box getroffen wurde. Dann enthalten nach $\sum_{i=0}^{m-1} X_i$ Würfen offenbar alle Boxen wenigstens eine Kugel.

Wenn genau i Boxen mindestens eine Kugel enthalten, dann gilt: Die Wahrscheinlichkeit, dass eine Kugel eine Box trifft, in der bisher noch keine Kugel lag, ist: $\frac{m-i}{m}$. Mit Lemma 2.1 folgt:

$$\mathbf{E}[X_i] = \frac{m}{m-i} \tag{4.14}$$

Denn solange noch keine neue Box getroffen wurde, handelt es sich bei jedem Wurf sicherlich um einen Bernoulli-Versuch.

Aufgrund der Linearität des Erwartungswertes ergibt sich mit Einsetzen von (4.14):

$$\begin{aligned}\mathbf{E}[\sum_{i=0}^{m-1} X_i] &= \sum_{i=0}^{m-1} \mathbf{E}[X_i] = \sum_{i=0}^{m-1} \frac{m}{m-i} \\ &= m \sum_{i=1}^{m} \frac{1}{m} = mH_m \leq m \ln m + m \qquad \text{nach Lemma 2.6}\end{aligned}$$ □

4.4.2 Obere Schranke

Wie zuvor gebe es als Eingabe n Kugeln, die maximal Abstand r vom Projektionszentrum haben. Es gebe eine Fläche $(1,t]_\bullet$, die in m Boxen aufgeteilt ist.

Die Zufallsgröße N bezeichne die Anzahl der Kugeln, die an den Culling-Algorithmus gesandt werden, bevor im Zwischenspeicher aus jeder Box eine Kugel enthalten ist. Die Zufallsgröße L gebe die Anzahl gezeichneter Objekte danach an. Die Boxen seien so gewählt, dass es ein $s \in (t,r]$ gebe, so dass anschließend keine Kugel mit Abstand vom Projektionszentrum $\geq s$ mehr gezeichnet werden muss.

Die Wahrscheinlichkeit, dass eine beliebige Kugel Abstand vom Projektionszentrum $< s$ hat, ist offenbar $F(s)$. Bezeichnet also M die Zufallsgröße, die die Anzahl gezeichneter Kugeln insgesamt angibt, so ergibt sich, dass $M \leq N + L$. Ist L' eine Zufallsgröße für die Anzahl der $(n-N)$ letzten erhaltenen Kugeln, die Abstand vom Projektionszentrum $< s$ haben, so ist L' verteilt wie $(n-N) \cdot F(s)$. Natürlich gilt: $L \leq L'$. Dementsprechend folgt für den Erwartungswert:

$$\begin{aligned}\mathbf{E}[M] &\leq \mathbf{E}[N] + \mathbf{E}[(n-N) \cdot F(s)] = \mathbf{E}[N] + nF(s) - \mathbf{E}[N] \cdot F(s) \\ &= (1 - F(s)) \cdot \mathbf{E}[N] + nF(s)\end{aligned}$$

Offenbar können nur solche Kugeln als Teil des Balls into Bins-Modells aufgefasst werden, die in $(1,t]_\bullet$ liegen. Ferner ist das Modell endlich, da nur n Kugeln existieren. Es bezeichne also N' eine Zufallsgröße, welche die (potentiell) unendliche Anzahl von Kugeln angibt, die der Culling-Algorithmus erhält, bis alle Boxen wenigstens eine Kugel enthalten. Dann gilt: $N = \min(n, N')$ und $\mathbf{E}[N] < \mathbf{E}[N'] = \frac{mH_m}{F(t)}$ und

$$\mathbf{E}[M] < (1 - F(s)) \cdot \mathbf{E}[N'] + nF(s) = \frac{1 - F(s)}{F(t)} \cdot mH_m + nF(s) \tag{4.15}$$

4.4.3 Geometrische Zusammenhänge

Wie eingangs des Abschnitts 4.4 skizziert, stelle man sich einen Kranz um das Projektionszentrum mit äußerem Radius t – also die Fläche $(1,t]_\bullet$ – in hinreichend viele Boxen aufgeteilt vor, wobei ferner im Zwischenspeicher aus jeder Box eine Kugeln enthalten ist. Dann sieht man leicht, dass es ein s gibt, so dass alle Kugeln mit Abstand $> s$ vom Projektionszentrum verdeckt werden. Im Folgenden wird eine Formel für dieses s hergeleitet. Dabei wird s in Abhängigkeit von t und dem maximalen Abstand a zweier Kugeln k, k' in benachbarten Boxen angegeben, für die gilt: $\operatorname{rad} k = \operatorname{rad} k' = t$. Abbildung 12 illustriert die geometrischen Zusammenhänge.

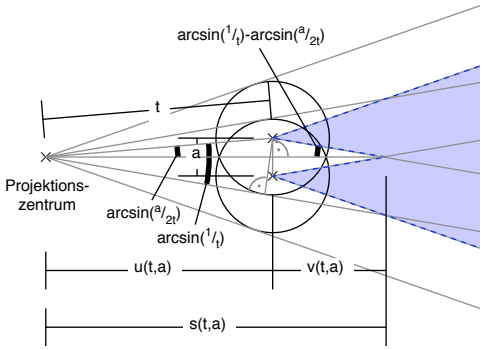


Abbildung 12: Verdeckter Bereich zweier Kugeln k, k' mit gegenseitigem Abstand a und $\operatorname{rad} k = \operatorname{rad} k' = t$

Lemma 4.16: *Seien $t \in (1,\infty)$, $a \in (0,1]$. Die Fläche $(1,t]_\bullet$ sei wie in Abbildung 11 in kongruente Boxen aufgeteilt, so dass zwei in benachbarten Boxen liegende Kugeln k, k' mit $\operatorname{rad} k = \operatorname{rad} k' = t$ maximal Abstand a haben können. Dann gilt: Ist in jeder der Boxen (mindestens) eine Kugel enthalten, so wird jede Kugel k'' mit $\operatorname{rad} k'' \geq (2a+1)t$ indirekt verdeckt.*

Beweis: Geometrisch ergibt sich für $t \in (0,\infty)$ und $a \in (0,2)$:

$$u(t,a) = \sqrt{t^2 - \frac{a^2}{4}} < t \tag{4.17}$$

$$v(t,a) = \frac{a}{2\tan(\arcsin\frac{1}{t} - \arcsin\frac{a}{2t})} \tag{4.18}$$

mit offensichtlich $s(t,a) = u(t,a) + v(t,a)$.

Es gilt:

$$\begin{aligned} s(t,1) &= u(t,1) + \frac{1}{2\tan(\arcsin\frac{1}{t} - \arcsin\frac{1}{2t})} && \text{nach (4.17), (4.18)} \\ &< u(t,1) + \frac{1}{2\tan(\arcsin\frac{1}{2t})} && (4.19) \\ &< u(t,1) + \frac{1}{2\tan(\frac{1}{2t})} && (4.20) \\ &< u(t,1) + t < 2t && (4.21) \end{aligned}$$

Dabei gilt (4.19), da arcsin auf $(0,1)$ konvex ist, also insbesondere $\arcsin\frac{x}{2} \leq \frac{1}{2}\arcsin x$. Die Konvexität ergibt sich daraus, dass $\frac{d^2}{dx^2}\arcsin x = \frac{x}{(1-x^2)^{3/2}} > 0$ für $x \in (0,1)$. (4.20) und (4.21) sind einzusehen, da die Ableitungen von arcsin und tan auf $(0,1)$ jeweils ≥ 1 sind. Folglich lässt sich für $a \in (0,1]$ folgende Abschätzung angeben:

$$\begin{aligned} s(t,a) &\leq t + \frac{a}{2\tan(\arcsin\frac{1}{t} - \arcsin\frac{a}{2t})} && \text{nach (4.17), (4.18)} \\ &\leq t + a \cdot \frac{1}{2\tan(\arcsin\frac{1}{t} - \arcsin\frac{1}{2t})} \\ &= t + a \cdot (s(t,1) - u(t,1)) \\ &< t + 2at = (2a+1)t && \text{nach (4.21)} \end{aligned}$$

□

Lemma 4.22: *Seien $t \in (1,\infty)$, $a \in (0,1]$ wie im vorigen Lemma. Dann muss die Fläche $(1,t]_\bullet$ in maximal $\lceil\frac{4t\pi}{m}\rceil$ kongruente Boxen aufgeteilt werden, so dass bei Vorhandensein einer Kugel in allen Boxen jede Kugel k mit $\operatorname{rad} k \geq s(t,a)$ indirekt verdeckt wird.*

Beweis: Ist die Fläche $(1,t]_\bullet$ in m kongruente Boxen aufgeteilt, so überdeckt eine Box den Winkel $\frac{2\pi}{m}$. Der Kreisbogen über Winkel $\frac{4\pi}{m}$ mit Radius t, also der Kreisbogen zweier benachbarter Boxen, ist obere Schranke für a und lässt sich durch $\frac{4t\pi}{m}$ angeben.

Für die maximale Anzahl benötigter Boxen $m(t,a)$ in Abhängigkeit von t und a ergibt sich durch Umstellung und dadurch, dass $m(t,a)$ eine natürliche Zahl repräsentiert:

$$m(t,a) \leq \lceil\frac{4t\pi}{a}\rceil < \frac{4t\pi}{a} + 1$$

□

4.4.4 Approximation mit mehrfacher Verdeckung

Erlaubt man im Modell, anders als zuvor, dass eine Kugel auch erst durch mehrere Kugeln zusammen verdeckt wird, so bleiben die bisherigen Erkenntnisse gültig. In Abbildung 13 ist der verdeckte Bereich farbig gekennzeichnet.

Analog zum Fall wie im letzten Unterabschnitt beschrieben, stellt man fest:

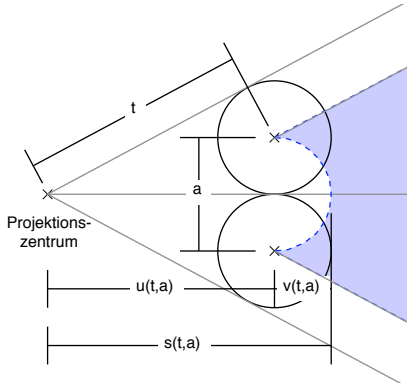


Abbildung 13: Verdeckter Bereich zweier Kugeln k, k' mit gegenseitigem Abstand a und $\operatorname{rad} k = \operatorname{rad} k' = t$, wenn mehrfache Verdeckung erlaubt ist

Lemma 4.23: *Seien $t \in (1, \infty)$, $a \in (0, 1]$. Die Fläche $(1, t]_\bullet$ sei in kongruente Boxen aufgeteilt, so dass zwei in benachbarten Boxen liegende Kugeln k, k' mit $\operatorname{rad} k = \operatorname{rad} k' = t$ maximal Abstand a haben können. Dann gilt: Ist in jeder der Boxen (mindestens) eine Kugel enthalten, so wird jede Kugel k'' mit $\operatorname{rad} k'' \geq t + \frac{a}{2}$ indirekt verdeckt.*

Beweis: Mit geometrischen Mitteln stellt man fest:

$$\begin{aligned} u(t,a) &= \sqrt{t^2 - \frac{a^2}{4}} < t \quad \text{und} \quad v(t,a) = 1 - \sqrt{1 - \frac{a^2}{4}} \\ s(t,a) &= u(t,a) + v(t,a) < t + 1 - \sqrt{1 - \frac{a^2}{4}} \\ &= t + 1 - \frac{\sqrt{(2-a)(2+a)}}{2} \leq t + 1 - \frac{\sqrt{(2-a)^2}}{2} \\ &= t + 1 - \frac{2-a}{2} = t + \frac{a}{2} \end{aligned}$$

□

Auch in diesem Fall strebt $s(t, a)$ nur für $t \to 1$ *und* gleichzeitig $a \to 0$ gegen 1. Im Folgenden wird daher ohne Einschränkung nur einfache Verdeckung wie im letzten Abschnitt betrachtet.

4.4.5 Verbesserung der Schranke

Die in (4.15) gegebene obere Schranke für die Anzahl zu zeichnender Kugeln,

$$\mathbf{E}[M] < \frac{1 - F(s(t,a))}{F(t)} m(t,a) H_{m(t,a)} + nF(s(t,a)),$$

liefert für unterschiedliche Parameter t und a unterschiedliche Abschätzungen, wobei die maximale Anzahl benötigter Boxen m sich nach Lemma 4.22 ergibt. Man sieht leicht, dass

durch feste Wahl von t und a nur die triviale Schranke von $\mathcal{O}(n)$ zu zeichnender Kugeln gezeigt werden kann.

Zu klären bleibt daher, ob sich die Parameter so in Abhängigkeit von n wählen lassen, dass sich eine bessere Schranke zeigen lässt.

Man betrachte dazu die Folgen $(a_i)_{i\in\mathbb{N}_0}$ und $(t_i)_{i\in\mathbb{N}_0}$, die wie folgt definiert seien:

$$a_0 := 1,\ a_i := \frac{a_{i-1}}{2} = a_0 \cdot 2^{-i} \qquad\qquad t_0 := r,\ t_i := \frac{t_{i-1}+1}{2},$$

jeweils für $i = 1, 2, 3, \ldots$.

Für die Verteilungsfunktion $F(x)$, die die Verteilung des Abstandes vom Projektionszentrum einer zufällig ausgewählten Kugel angibt (für die Definition von F siehe (4.6)), lassen sich folgende Ungleichungen angeben. Es sei $t > 1$:

$$F\left(\frac{t+1}{2}\right) = \frac{(t^2+2t+1)-4}{4(r^2-1)} = \frac{t^2-1+2t-2}{4(r^2-1)} > \frac{t^2-1}{4(r^2-1)} = \frac{1}{4}\cdot F(t), \tag{4.24}$$

da $2t > 2$, und

$$\begin{aligned} F\left(\frac{t+1}{2}\right) &= \left((t^2-1)+2(t-1)\right)\cdot\frac{1}{4(r^2-1)} = \left(1+\frac{2}{t+1}\right)\cdot\underbrace{\frac{t^2-1}{4(r^2-1)}}_{=\frac{1}{4}\cdot F(t)} \\ &= \frac{t+3}{4t+4}\cdot F(t) < \frac{1}{2}\cdot F(t) \end{aligned} \tag{4.25}$$

Ferner seien Folgen: $(m_i)_{i\in\mathbb{N}_0}$ und $(s_i)_{i\in\mathbb{N}_0}$ mit

$$m_i := \frac{4r\pi+1}{a_i} \qquad\qquad s_i := (2a_i+1)t_i$$

für $i = 0, 1, 2, \ldots$.

Nach Lemma 4.22 gilt für die maximale Anzahl benötigter Boxen $m(t_i, a_i)$:

$$m(t_i,a_i) < \frac{4t_i\pi + a_i}{a_i} \le \frac{4r\pi+1}{a_i} = m_i \tag{4.26}$$

und

$$m_i = \frac{4r\pi+1}{a_0\cdot 2^{-i}} = 2^i m_0 \tag{4.27}$$

Ferner gilt nach Lemma 4.16, dass $s(t_i, a_i) < (2a_i+1)t_i = s_i$. Außerdem:

$$\begin{aligned} s_{i+1} = (2a_{i+1}+1)t_{i+1} &= \frac{(a_i+1)(t_i+1)}{2} = \frac{a_it_i + a_i + t_i + 1}{2} \\ &< \frac{a_it_i + a_it_i + t_i + 1}{2} && \text{da } t_i > 1 \\ &= \frac{2a_it_i + t_i + 1}{2} = \frac{(2a_i+1)t_i+1}{2} = \frac{s_i+1}{2} \end{aligned}$$

Mit (4.24) ergibt sich $F(t_{i+1}) > \frac{1}{4} \cdot F(t_i)$ und mit (4.25) ergibt sich $F(s_{i+1}) < \frac{1}{2} \cdot F(s_i)$. Induktiv folgt also:

$$F(t_i) > 4^{-i} F(t_0) \tag{4.28}$$
$$F(s_i) < 2^{-i} F(s_0) \tag{4.29}$$

Schließlich sei $(n_i)_{i \in \mathbb{N}_0}$ mit $n_i := k^i$ für ein $k \in \mathbb{N}$. Dann:

$$\begin{aligned} \mathbf{E}[M] &< \underbrace{\frac{1 - F(s(t_i, a_i))}{F(t_i)}}_{< \frac{1}{F(t_0)} < \frac{4^i}{F(t_0)}} \cdot \underbrace{m_i}_{=2^i m_0} \cdot \underbrace{H_{m_i}}_{\leq i \cdot c} + \underbrace{n_i}_{=k^i} \cdot \underbrace{F(s_i)}_{< 2^{-i} F(s_0)} \\ &< 8^i \cdot i \cdot c' + \frac{k^i}{2^i} \cdot c'', \end{aligned} \tag{4.30}$$

wobei c, c', c'' nicht von i abhängig, hier also Konstanten, sind. Die Abschätzungen folgen direkt aus (4.26), (4.27), (4.28) und (4.29).

Wegen $i = \log_k n$ ergibt sich:

$$\begin{aligned} \mathbf{E}[M] &< 8^{\log_k n} \cdot (\log_k n) \cdot c' + \frac{n}{2^{\log_k n}} \cdot c'' \\ &= n^{3 \log_k 2} \cdot (\log_k n) \cdot c' + n^{1 - \log_k 2} \cdot c'' \end{aligned} \tag{4.31}$$

Offensichtlich wird der Term $\max\{3 \log_k 2, 1 - \log_k 2\}$ durch $\log_k 2 = \frac{1}{4}$, also $k = 16$, minimiert. Folglich erhält man als obere Schranke:

$$\mathbf{E}[M] < n^{3/4} \cdot (\log_{16} n) \cdot c' + n^{3/4} \cdot c'' = \mathcal{O}(n^{3/4} \cdot \log_{16} n) \tag{4.32}$$

4.4.6 Untere Schranke

Eine untere Schranke für zufällig gleichverteilte Anordnungen kann gegeben werden durch die Anzahl der Kugeln, die jeweils bei Erhalt durch den Culling-Algorithmus den geringsten Abstand vom Projektionszentrum aller bislang erhaltenen Kugeln haben.

Ohne Einschränkung sei die Reihenfolge der Kugeln $k_1, \ldots, k_n$. Somit gilt, wenn K_i Indikator-Zufallsgröße dafür ist, ob Kugel k_i gezeichnet werden muss:

$$\mathbf{Pr}[K_i = 1] \geq \frac{1}{i} \tag{4.33}$$

Denn die i-te Kugel ist mit Wahrscheinlichkeit $\geq \frac{1}{i}$ diejenige mit dem bislang kleinsten Abstand: Offensichtlich muss für i unabhängige und gleichverteilte Zufallsgrößen $R_1, \ldots, R_i$ (die den Abstand der Kugel k_i vom Projektionszentrum angeben) gelten, dass $\mathbf{Pr}[R_i = \min\{R_1, \ldots, R_i\}] \geq \frac{1}{i}$. Man beachte, das *nicht* vorausgesetzt wird, dass R_i echt kleiner als alle anderen R_j sein muss.

Bemerkung 4.34: Da die R_i stetige Zufallsgrößen sind, folgt eigentlich sogar Gleichheit in (4.33). Denn: Aufgrund der Stetigkeit gilt: $\mathbf{Pr}[R_i = R_j] = 0$ für $i \neq j$.

Es ergibt sich als untere Schranke für die Anzahl zu zeichnender Kugeln:

$$\mathbf{E}[\sum_{i=1}^{n} K_i] \geq \sum_{i=1}^{n} \frac{1}{i} = H_n \tag{4.35}$$

Die Erkenntnisse der letzten beiden Abschnitte in einem Satz zusammengefasst:

Satz 4.36: *Sei* ALG *ein Culling-Algorithmus mit unbegrenztem Zwischenspeicher, der jede gezeichnete Kugel in den Zwischenspeicher aufnimmt. Dann gelten folgende Schranken bei einer zufälligen Eingabe I von n zufällig unabhängig gleichverteilten Kugeln in $(1, r]_\bullet$, $r > 1$, und zufälliger Reihenfolge der Kugeln:*

i) $\mathbf{E}[\text{ALG}[I]] = o(n^{3/4} \cdot \log_{16} n)$

ii) $\mathbf{E}[\text{ALG}[I]] = \omega(\ln n)$

Insbesondere grenzen diese Schranken die Kosten eines optimalen Culling-Algorithmus ein.

Beweis: Die Schranken wurden in diesem und dem letzten Abschnitt gezeigt. Ferner wurde in Abschnitt 4 begründet, dass ALG optimal ist. □

5 Schranken im Modell mit beschränktem Zwischenspeicher

Das Einführen einer Obergrenze für die Anzahl von Objekten im Zwischenspeicher macht das Culling-Problem zu einem tatsächlichen Online-Problem. In diesem Fall muss der Algorithmus Entscheidungen treffen, die seine zukünftigen Kosten stark beeinflussen können, wie in Abschnitt 3.3 aufgezeigt wurde.

5.1 Konstant großer Zwischenspeicher

Es sind verschiedene Heuristiken denkbar, um im Erwartungswert zu einer zufälligen Anordnung von Objekten und zufälliger Reihenfolge möglichst viele Objekte aussortieren zu können:

- Maximiere die überdeckte Fläche: Entferne bei Erhalt einer Kugel jeweils die aus dem Zwischenspeicher, bei der die überdeckte Gesamtfläche aller Kugeln im Zwischenspeicher am größten bleibt beziehungsweise wird.
- Teile wie zuvor einen Kranz um das Projektionszentrum in m Boxen auf (m kleiner oder gleich der Größe des Zwischenspeichers) und versuche, für jede Box eine Kugel in den Zwischenspeicher zu bekommen. Fallen mehrere Kugeln in eine Box, so verwerfe die Kugel, die größeren Abstand vom Projektionszentrum hat.

Betrachtet wird hier der zweite Ansatz. Für jeden solchen Algorithmus ergeben sich asymptotisch für die Anzahl gezeichneter Kugeln:

- Als obere Schranke eine Abschätzung der Form, wie in (4.15) gegeben. Offenbar kann es nur eine begrenzte Anzahl von Boxen geben, so dass es einen kleinsten Kranz gibt, außerhalb dessen alle Kugeln garantiert verdeckt werden. Umgekehrt kann in der Abschätzung für einen konstanten Bruchteil der Kugeln nicht garantiert werden, dass sie durch ein Objekt im Zwischenspeicher verdeckt werden.

- Als untere Schranke weiter die in Satz 4.36 (ii) gegebene Abschätzung.

Sei ALG ein Online-Algorithmus wie gerade beschrieben. Als einfache Folgerung ergibt sich, dass für den in Abschnitt 3.3.3 vorgeschlagenen Quotienten (3.14) gilt:

$$\frac{\mathbf{E}[\text{ALG}[I]]}{\mathbf{E}[\text{OPT}[I]]} = \mathcal{O}\left(\frac{n}{\ln n}\right)$$

I ist hier wieder eine zufällige Eingabe mit n Kugeln.

5.2 Obere Schranke bei begrenztem Zwischenspeicher in Abhängigkeit der Anzahl von Kugeln

Wie gesehen, kann bei Zwischenspeicher der Größe $\mathcal{O}(1)$ mit dem gewählten Ansatz für keinen Online-Algorithmus eine bessere Schranke als $\mathcal{O}(n)$ zu zeichnende Kugeln gezeigt werden.

In Abschnitt 4 wurde für den Zwischenspeicher angenommen, dass seine Größe unbeschränkt ist. Bei der in dieser Arbeit verwendeten Modellierung von Occlusion Culling ist dies aber äquivalent zu einer auf n beschränkten Größe. Dies ist sofort einsehbar, denn nach Voraussetzung gibt es nur n Kugeln.

Zu klären bleibt daher die Frage, welche obere Schranke Online-Algorithmen erzielen können, für deren Zwischenspeicher man zwar Größe $o(n)$ verlangt, aber $\omega(1)$ zulässt. Es wird im Folgenden weiterhin der Ansatz aus dem vorangegangenen Abschnitt verwendet und der Fall mit Zwischenspeicher der Größe n^c untersucht, wobei $c \in (0, 1)$.

Nach (4.30) gilt:

$$\mathbf{E}[M] < \frac{4^i}{F(t_0)} \cdot 2^i m_0 \cdot i \cdot c + n_i \cdot 2^{-i} F(s_0)$$

Es gibt offenbar auch im Fall mit auf n^c beschränktem Speicher Online-Algorithmen, für die diese obere Schranke gilt, solange $n_i^c \geq 2^i m_0 = m_i$. Denn dann bleibt die Anzahl der Boxen ($= 2^i m_0$) unter der Größe des Zwischenspeichers ($= n_i^c$). Wählt man also die Folge $(n_i)_{i \in \mathbb{N}_0}$ etwas anders als zuvor, nämlich $n_i := (m_0 k)^i$ für ein $k \in \mathbb{R}_{>1}$, dann gilt: $n_i^c = (m_0 k)^{ic} = k^{ic} \cdot m_0^{ic}$. Wenn nun $k^c \geq 2 \Leftrightarrow k \geq 2^{1/c}$, so existiert ein $i_0 \in \mathbb{N}_0$ mit $n_i^c \geq 2^i m_0^{ic} \geq 2^i m_0$ für alle $i \geq i_0$. Die Gültigkeit von Satz (4.36) würde also auch hier folgen.

Man sieht, dass man durch genügend große Wahl von k eine Schranke $o(n)$ erhält: Offensichtlich ist $n^{3 \log_k 2} = o(n)$ für $k > 8$ und auch $n^{1 - \log_k 2} = o(n)$ für alle $k > 1$. Ein Beispiel soll diese Erkenntnis verdeutlichen:

Beispiel 5.1: Der Zwischenspeicher sei auf $n^{1/4}$ Kugeln beschränkt, das heißt $c = \frac{1}{4}$. Wegen $16^{1/4} = 2$ gilt (4.31) für $k \geq 16$. Damit folgt aber die obere Schranke von $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$ wie in Satz 4.36 (i) .

Ferner: Sei ALG ein Online-Algorithmus, der zu den zuvor beschriebenen Folgen $(t_i), (m_i)$ sowie (n_i) ein i so wählt, dass $n^i > n$. Sei $k = 16$. Die Fläche $(1, t_i]_\bullet$ wird also in m_i kongruente Boxen eingeteilt. Dann gilt mit dem oben beschrieben Ansatz für den Quotienten (3.14):

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

Es bleibt in zukünftigen Arbeiten zu zeigen, ob sich Online-Algorithmen mit besseren garantierbaren Schranken finden lassen. Es ist zwar einzusehen, dass obige Algorithmen in der Praxis bereits deutliche Verbesserungen erzielen würden, aber gemäß der Bewertung, wie sie in Abschnitt 3.3.3 vorgeschlagen wurde, kann nach wie vor kein konstantes Verhältnis zu einem optimalen Offline-Algorithmus angegeben werden.

6 Komplexität des Culling-Problems

Es bleibt noch zu überlegen, welche Aussagen sich zur Komplexität des Culling-Problems machen lassen.

6.1 Das Entscheidungsproblem

Die bekannten Klassen $\mathcal{P}$ und $\mathcal{NP}$ sind Einteilungen von *Entscheidungsproblemen* – also solchen Problemen, die als Antwort nur einen der beiden Boole'schen Wahrheitswerte ergeben und einer Turingmaschine codiert als endliche Eingabe gegeben werden können.

Offensichtlich ist das Culling-Problem aus Definition 3.5 kein Entscheidungsproblem, sondern ein Optimierungsproblem. Kann aber ein Entscheidungsproblem CULLING_{Ent} angegeben werden, so dass:

- CULLING_{Ent} $\mathcal{NP}$-vollständig ist und
- ein effizienter Algorithmus für das Culling-Problem sofort zu einem effizienten Lösungsverfahren für CULLING_{Ent} führen würde,

dann ist es sehr unwahrscheinlich, dass das Culling-Problem effizient, also in Polynomialzeit, lösbar ist – denn dies würde $\mathcal{P} = \mathcal{NP}$ implizieren.

In der Komplexitätstheorie heißt ein Entscheidungsproblem A *in polynomieller Zeit reduzierbar* auf ein anderes Entscheidungsproblem B, wenn es eine Turingmaschine gibt, die

- bei jeder Eingabe nach polynomiell vielen Schritten bezüglich der Eingabelänge anhält,

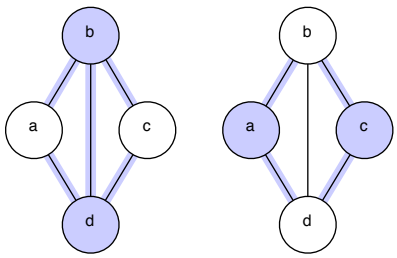
- jede codierte Eingabe a einer Frage aus A in eine codierte Eingabe b einer Frage aus B transformiert, so dass die Antwort für b genau dann wahr ist, wenn die Antwort für a wahr ist.

Eine solche Transformation heißt *Reduktion*. Üblicherweise weist man die $\mathcal{NP}$-Vollständigkeit (sofern gegeben) für ein neues Problem A durch eine polynomielle Reduktion eines Problems L auf A nach, von dem $\mathcal{NP}$-Vollständigkeit bereits bekannt ist. Denn: Nach Voraussetzung ist jedes Problem in $\mathcal{NP}$ in polynomieller Zeit auf L reduzierbar. L kann aber in polynomieller Zeit auf A reduziert werden. Verkettung der beiden Abbildungen (Hintereinanderausführung der entsprechenden Turingmaschinen) liefert die Behauptung.

Im Folgenden wird eine Reduktion des Entscheidungsproblems Knotenüberdeckung verwendet:

Definition 6.1: Das Entscheidungsproblem KNOTENÜBERDECKUNG$_{Ent}$ ist definiert als die Frage, ob es zu einem ungerichteten Graphen $G = (V, E)$ und einem $k \in \mathbb{N}_0$ eine Teilmenge $V' \subseteq V$ gibt, so dass $|V'| \leq k$ und für jede Kante $\{e, d\} \in E$ wenigstens e oder d in V' enthalten ist.

Beispiel 6.2: Betrachte Abbildung 14.



(a) (b)

Abbildung 14: Beispiel für eine Knotenüberdeckung

Während in (a) die Menge der markierten Knoten eine 2-Knotenüberdeckung ergibt, ist (b) sicherlich keine 2-Knotenüberdeckung, da die mittlere Kante nicht überdeckt wird. ◊

Von KNOTENÜBERDECKUNG$_{Ent}$ ist die $\mathcal{NP}$-Vollständigkeit bekannt. Einen Beweis geben beispielsweise Garey u. Johnson (1979, Seite 54 f.).

6.2 Das Problem Knotenaussortierung

Definition 6.3: Das zum Problem Knotenaussortierung (Definition 3.7) gehörige Entscheidungsproblem, KNOTENAUSSORTIERUNG$_{Ent}$, ist definiert als die Frage, ob es zu einem gegebenen Verdeckungsgraphen $G = (V, E)$, einer Knotenreihenfolge $\rho : V \to N_{\leq |V|}$

bijektiv, und einem Zwischenspeicher der Größe c eine Verdrängungssequenz gibt, so dass $\geq m$ Knoten aussortiert werden.

Satz 6.4: KNOTENAUSSORTIERUNG$_{Ent}$ *ist $\mathcal{NP}$-vollständig. Insbesondere ist das Optimierungsproblem Knotenaussortierung daher nicht in Polynomialzeit lösbar, falls $\mathcal{P} \neq \mathcal{NP}$.*

Beweis: Ist KNOTENAUSSORTIERUNG$_{Ent}$ $\mathcal{NP}$-vollständig, so folgt daraus sofort, dass das Problem Knotenaussortierung nicht in Polynomialzeit lösbar ist, falls $\mathcal{P} \neq \mathcal{NP}$. Denn andernfalls könnte sicherlich die optimale Lösung daraufhin überprüft werden, ob $\geq m$ Knoten aussortiert werden. Zu zeigen ist also nur die $\mathcal{NP}$-Vollständigkeit. Der Beweis werde durch Reduktion von KNOTENÜBERDECKUNG$_{Ent}$ geführt, von dem bereits die $\mathcal{NP}$-Vollständigkeit bekannt ist.

Es gebe einen Algorithmus KNOTENAUSSORTIERUNGLÖSER für das Entscheidungsproblem KNOTENAUSSORTIERUNG$_{Ent}$:

Eingabe: Verdeckungsgraph $G = (V, E)$, $n := |V|$, Reihenfolge $\rho : V \to \mathbb{N}_{\leq n}$ bijektiv, Größe des Zwischenspeichers c, Mindestanzahl m aussortierter Knoten.

Ausgabe: Wahr, wenn es für den Zwischenspeicher der Größe c eine Verdrängungssequenz gibt, so dass die Anzahl aussortierter Knoten $\geq m$ ist. Falsch andernfalls.

Betrachte folgende Reduktion f für KNOTENÜBERDECKUNG$_{Ent}$:

Eingabe: Ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}_0$. 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: Gerichteter azyklischer Graph $G' = (V', E')$ 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, also

$$\rho(v') = \begin{cases} i & \text{wenn } v' = v_i \\ n + i & \text{wenn } v' = e_i \end{cases}.$$

$c = k$. $m = |E|$. Für ein Beispiel siehe Abbildung 15.

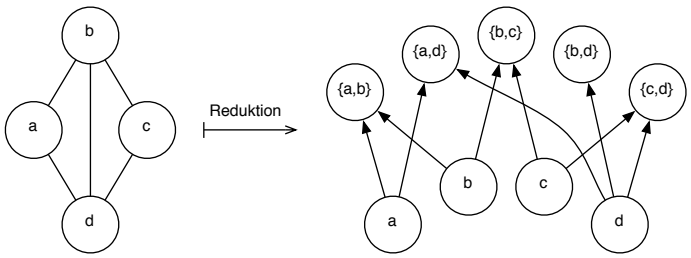


Abbildung 15: Beispiel einer Reduktion von KNOTENÜBERDECKUNG$_{Ent}$ auf KNOTENAUSSORTIERUNG$_{Ent}$

Für die Reduktion f ließe sich eine Turingmaschine angeben, die stets in Polynomialzeit bezüglich der Eingabelänge anhält: Die Überprüfung auf korrekte Codierung kann sicherlich in Polynomialzeit geschehen, denn andernfalls läge KNOTENÜBERDECKUNG$_{Ent}$ nicht in $\mathcal{NP}$.

Der aufgebaute gerichtete azyklische Graph hat $|V| + |E|$ Knoten und $2 \cdot |E|$ Kanten. Offensichtlich kann er also in polynomieller Zeit konstruiert werden.

Genauer: Ein Algorithmus für f könnte zuerst für G eine Adjazenzmatrix berechnen, sofern die Eingabe nicht ohnehin als solche Matrix vorliegt. Anschließend wird eine Adjazenzmatrix für G' generiert. Dazu wird eine quadratische Tabelle mit insgesamt $(|V|+|E|)^2$ Feldern anfänglich mit Nullen initialisiert und anschließend werden für jede Kante $\{v_j, v_k\} \in E$ die beiden Zellen auf 1 gesetzt, die eine Kante in G' von v_j zu $\{v_j, v_k\}$ beziehungsweise v_k zu $\{v_j, v_k\}$ repräsentieren. Offensichtlich sind diese Operationen alle in $\mathcal{O}((|V| + |E|)^k)$ durchführbar, wobei $k \in \mathbb{N}$. Auch ρ lässt sich in Polynomialzeit bezüglich Eingabelänge codieren – die Codierung von ρ ist sogar mit Länge $\mathcal{O}(|V| \cdot \log |V|)$ möglich. Folglich ist die gesamte Ausgabecodierung nur polynomiell in der Eingabelänge, denn bereits die Länge der Binärcodierung (nur) für G ist mindestens $\omega(|V| + |E|)$.

Korrektheit der Reduktion vorausgesetzt, könnte nun ein Algorithmus KNOTENÜBERDECKUNGLÖSER für das Problem Knotenüberdeckung angegeben werden:

$$\begin{aligned} &\text{KNOTENÜBERDECKUNGLÖSER}(G, k) \\ = \ &\text{KNOTENAUSSORTIERUNGLÖSER}(f(G, k)) \end{aligned}$$

Für die Korrektheit der Reduktion ist also zu zeigen:

$$\begin{aligned} &\text{KNOTENAUSSORTIERUNGLÖSER}(G', \rho, c, m) = \text{wahr} \\ \Leftrightarrow \ &\text{KNOTENÜBERDECKUNGLÖSER}(G, k) = \text{wahr}, \end{aligned}$$

wobei die Bezeichnungen wie zuvor beschrieben seien, insbesondere $(G', \rho, c, m) := f(G, k)$.

„⇒“ Es gibt zu der Knotensequenz ρ eine Verdrängungsreihenfolge aus dem Zwischenspeicher der Größe c, so dass $\geq m$ Knoten aussortiert werden. Offensichtlich können nur die Knoten $e_1, \ldots, e_s$ aussortiert werden, denn nur sie haben eingehende Kanten. Wenn also $|E|$ Knoten aussortiert werden, so folgt, dass dies genau die Knoten $e_1, \ldots, e_s$ sein müssen. Nach Wahl der Reihenfolge ρ gibt es also eine Teilmenge von V' der Größe $\leq c$, die alle Knoten $e_1, \ldots, e_s$ verdeckt. Dann gibt es nach Konstruktion aber eine Teilmenge $S \subseteq V$, so dass für jede Kante aus E ein Endpunkt in S liegt. Folglich wäre auch KNOTENÜBERDECKUNGLÖSER$(G, k) =$ wahr.

„⇐“ Es gibt eine k-Knotenüberdeckung. Dementsprechend gibt es eine Teilmenge $S \subseteq V$, so dass jede Kante aus E einen Endpunkt in S hat. Folglich genügt es, k der Kugeln $v_1, \ldots, v_n$ im Zwischenspeicher zu behalten, damit bei der Knotenaussortierung $|E|$ Knoten aussortiert würden. Da $c = k$ und $m = |E|$, folgt die Behauptung. □

Es ist also nach derzeitigem Erkenntnisstand der Komplexitätstheorie sehr unwahrscheinlich, für das Problem Knotenaussortierung einen effizienten Algorithmus zu finden. Aufgrund von Bemerkung 3.9 muss sich diese Eigenschaft natürlich nicht zwangsläufig auf das

Culling-Problem übertragen. Der Nachweis der Vermutung, dass die $\mathcal{NP}$-Vollständigkeit auch für das Culling-Problem gelte – beziehungsweise dessen Widerlegung –, wären in einer weiteren Arbeit noch zu zeigen.

7 Fazit und Ausblick

In dieser Arbeit wurde eine zweidimensionale Modellierung von Occlusion Culling vorgenommen, bei der der Culling-Algorithmus seine Entscheidungen aufgrund von Objekten in einem Zwischenspeicher treffen kann. In diesem Puffer können ausschließlich bereits gezeichnete Objekte enthalten sein. Eine Formalisierung des Problems Online Occlusion Culling kann auf den ersten Blick an das Paging-Problem erinnern, für das sowohl ein optimaler Offline Algorithmus als auch Online-Algorithmen mit konstantem Competitive Ratio bekannt sind.

Es wurde jedoch gezeigt, dass Algorithmen für das Online Occlusion Culling nicht mittels der üblichen Competitive Analysis bewertet werden können, da kein Online Algorithmus $o(n)$-competitive sein kann. Aus diesem Grund wurde vorgeschlagen, den Erwartungswert über das Verhältnis der Kosten von Online- zu Offline-Algorithmus bei einer zufälligen Eingabe zu betrachten.

Zunächst wurde für unbegrenzten Zwischenspeicher eine obere und untere Schranke für die durchschnittlich zu zeichnenden Kugeln aufgestellt, falls der Algorithmus jede gezeichnete Kugel anschließend in seinen Zwischenspeicher aufnimmt. Ein solcher Algorithmus ist bei unbegrenztem Zwischenspeicher stets optimal bezüglich des Kostenmaßes „gezeichnete Kugeln“. Als untere Schranke konnte $\omega(\ln n)$ und als obere Schranke $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$ aufgezeigt werden.

Im Falle des begrenzten Zwischenspeichers wurde der Online-Algorithmus betrachtet, der einen Kranz um das Projektionszentrum in kongruente Boxen aufteilt und nur jene Kugeln wieder aus dem Zwischenspeicher verdrängt, die nicht in den Kranz fallen. Bei konstant großem (von n unabhängigem) Zwischenspeicher kann auf diese Weise prinzipiell keine bessere obere Schranke als $\mathcal{O}(n)$ aufgezeigt werden. Erlaubt man jedoch einen Zwischenspeicher der Größe n^c, wobei $c \in (0, 1)$, so wurde gezeigt, dass die Anzahl gezeichneter Kugeln $o(n)$ sein muss. Falls $c \geq \frac{1}{4}$, gilt sogar weiterhin die obere Schranke von $\mathcal{O}(n^{3/4} \cdot \log_{16} n)$ wie im Falle des unbegrenzten Zwischenspeichers.

Schließlich wurde im letzten Abschnitt die $\mathcal{NP}$-Vollständigkeit des mit dem Culling-Problem verwandten Graphen-Problems KNOTENAUSSORTIERUNG$_{Ent}$ gezeigt.

In folgenden Arbeiten gilt es insbesondere zu untersuchen, ob es noch bessere Online-Algorithmen für das Occlusion Culling gibt, beziehungsweise ob bessere Schranken aufgezeigt werden können. Ferner ist von Interesse, ob auch das Culling-Problem selbst $\mathcal{NP}$-vollständig ist. Schließlich gilt es, die Erkenntnisse auf den üblichen dreidimensionalen Fall zu übertragen und Praxistests zu unterziehen.

Abbildungsverzeichnis

Literatur

Die Angabe verwendeter Literatur erfolgt in alphabetisch sortierter Reihenfolge. Hinter jedem Eintrag ist eine Liste der Seiten angegeben, von denen ein Verweis erfolgte.

Borodin u. El-Yaniv 1998
BORODIN, Allan ; EL-YANIV, Ran: *Online Computation and Competitive Analysis.* Cambridge University Press, 1998 14

Garey u. Johnson 1979
GAREY, Michael R. ; JOHNSON, David S.: *Computers and Intractability.* W. H. Freeman and Company, New York, 1979 32

Heuser 1991
HEUSER, Harro: *Lehrbuch der Analysis.* Teubner, Stuttgart, 1991 (2. Teil) 7

Hübner 2003
HÜBNER, Gerhard: *Stochastik – Eine anwendungsorientierte Einführung für Informatiker, Ingenieure und Mathematiker.* 4. Auflage. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2003 5, 6

Karlin u. a. 1988
KARLIN, Anna ; MANASSE, Mark ; RUDOLPH, Larry ; SLEATOR, Daniel D.: Competitive snoopy caching. In: *Algorithmica* 3 (1988), Nr. 1, S. 79–119 13

Koutsoupias u. Papadimitriou 2000
KOUTSOUPIAS, Elias ; PAPADIMITRIOU, Christos H.: Beyond Competitive Analysis. In: *SIAM Journal on Computing* 30 (2000), Nr. 1, 300–317 (Vorläufige Version veröffentlicht in Proc. 35th Symp. Foundations of Computer Science, IEEE, 1994). http://epubs.siam.org/sam-bin/dbq/article/29954 15, 16

Papoulis 1984
PAPOULIS, Athanasios: *Probability, Random Variables, and Stochastik Processes*. McGraw Hill, New York, 1984 6

Raab u. Steger 1998
RAAB, Martin ; STEGER, Angelika: "Balls into Bins" - A Simple and Tight Analysis. In: *RANDOM '98: Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, Springer-Verlag, 1998. – ISBN 3–540–65142–X, S. 159–170 22

Sleator u. Tarjan 1985
SLEATOR, Daniel D. ; TARJAN, Robert E.: Amortized efficiency of list update and paging rules. In: *Commun. ACM* 28 (1985), Nr. 2, 202–208. http://doi.acm.org/10.1145/2786.2793. – ISSN 0001–0782 13, 15

Watt 2002
WATT, Alan: *3D-Computergrafik*. 3. Auflage. Pearson Studium, München, 2002 4, 5

Danksagung

Ganz besonders bedanken möchte ich mich bei Herrn Dr. Christian Sohler für die Betreuung und seine zahlreichen Anregungen. Außerdem danke ich meiner Familie und meinen Kommilitonen, die mich beim Korrekturlesen auf zahlreiche Tippfehler hingewiesen und mir in der letzten Phase des Aufschreibens geholfen haben, unverständliche Passagen aufzuspüren und (auch für fachfremde Personen) nachvollziehbarer aufzuschreiben.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe sowie ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet. Die Arbeit hat, soweit mir bekannt, in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Paderborn, den 6. Januar 2005

_____

Florian Schoppmann