Ein Überdeckungsproblem für beliebig dimensionierte Hyperquader

Diplomarbeit

Jörg Hoffmann

Albert-Ludwigs-Universität Freiburg
Fakultät für angewandte Wissenschaften
Institut für Informatik

März 1999

Danksagung

An dieser Stelle möchte ich mich bei allen Mitarbeitern der Abteilung *Grundlagen der künstlichen Intelligenz* bedanken, die bei Fragen und Problemen stets hilfsbereit waren. Mein besonderer Dank gilt Prof. Dr. Bernhard Nebel, der die Entstehung dieser Arbeit ermöglicht hat, Andreas Schön fürs Korrekturlesen und vor allem Dr. Jana Koehler für die Betreuung der Arbeit.

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe.

Freiburg, im März 1999 Jörg Hoffmann

Zusammenfassung

In der vorliegenden Arbeit wird ein geometrisches Überdeckungsproblem für beliebig dimensionierte Hyperquader behandelt. Es wird gezeigt, in welchem Zusammenhang Probleme dieser Art auftauchen, und ihre komplexitätstheoretischen Eigenschaften werden untersucht. Anschließend werden, aufbauend auf einer Übersicht über gängige algorithmische Methoden, zwei Lösungsansätze für das Problem entwickelt. Zum einen entsteht so ein inkrementelles Schnittverfahren, zum anderen ein verallgemeinertes Scan-line Verfahren. Beide Algorithmen werden anhand einer empirischen Untersuchung bezüglich ihrer Effizienz verglichen. Abschließend werden Zukunftsperspektiven der Methodik aufgezeigt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die vorliegende Arbeit beschäftigt sich mit einer Fragestellung, die in erster Linie ins Gebiet der Algorithmentheorie einzuordnen ist, genauer in den Bereich der Algorithmischen Geometrie. Es wird ein klar abgegrenztes algorithmisches Problem definiert; dessen theoretische Komplexitätseigenschaften werden untersucht, verschiedene Lösungsmöglichkeiten werden entwickelt und miteinander verglichen. Behandelt wird ein *Überdeckungsproblem für beliebig dimensionerte Hyperquader*. Dies läßt sich informell wie folgt charakterisieren:

Gegeben seien eine Menge $\mathcal{H}$ von n-dimensionalen Hyperquadern sowie ein weiterer Hyperquader Q. Wird Q von den Quadern in $\mathcal{H}$ überdeckt, d.h., ist Q ganz in den Quadern aus $\mathcal{H}$ enthalten ?

Hyperquader sind in dieser Fragestellung eine natürliche Verallgemeinerung von Quadern auf den n-dimensionalen Fall. Obwohl das Problem auf den ersten Blick ein Standardproblem sein könnte - Quader und Überdeckungen sind ja durchaus keine unbekannten Begriffe -, gibt es wenig anwendbare Literatur. Insbesondere gibt es keine dem Autor bekannte Arbeit, die sich mit *genau* diesem Problem beschäftigt. Klassische geometrische Algorithmen behandeln in der Hauptsache zwei-, höchstens dreidimensionale Objekte, aus dem naheliegenden Grund, daß real existierende Objekte eben höchstens dreidimensional sind. Trotzdem lassen sich natürlich die in der Algorithmischen Geometrie angewendeten Verfahren auf höher dimensionierte Probleme übertragen. Genaueres zur Einordnung dieser Arbeit in die Literatur findet sich in Kapitel 3. Zunächst versuchen wir, die Fragestellung zu motivieren, also aufzuzeigen, in welchem Zusammenhang Überdeckungsprobleme dieser Art auftauchen.

1.1 Motivation

Die einzige (dem Autor bekannte) *direkte* Anwendung des Problems findet sich in der KI-Handlungsplanung. Eine genaue Definition der im Gebiet der Handlungsplanung behandelten Probleme würde den Rahmen dieser Arbeit sprengen und wäre auch an dieser Stelle wenig sinnvoll. Grob gesagt, ist ein *initialer* Weltzustand gegeben, weiterhin ein erwünschter *Zielzustand* und eine Menge von möglichen Aktionen, also Handlungen. Es gibt verschiedene Ansätze zur Definition von Weltzuständen und Aktionen; eine davon ist die klassische STRIPS-Planung aus [Fikes and Nilsson, 1971]. In einer beweisbar konsistenten Variante hiervon, vorgeschlagen in [Lifschitz, 1990], sind Weltzustände Mengen von logischen Grundatomen.[1] Ebenso hat jede mögliche Aktion logische Atome als *Vorbedingungen*. Als Effekt fügen Aktionen einige neue logische Atome zum Weltzustand hinzu und löschen andere Atome aus demselben. Das Problem besteht nun darin, eine Folge von Aktionen zu finden, die den Initialzustand in den Zielzustand überführt.

Es gibt verschiedene Lösungsalgorithmen für dieses Problem, etwa das in [Blum and Furst, 1997] vorgestellte GRAPHPLAN System. Der GRAPHPLAN Algorithmus arbeitet mit *Zeitschritten*, d.h. auf Zeitschritt t werden alle Weltzustände repräsentiert, die vom Initialzustand aus in t Schritten erreichbar sind. Ebenso kann man auf jedem Zeitschritt unlösbare Zielmengen (d.h. Mengen von Atomen, für die man bereits erfolglos auf diesem Zeitschritt geplant hat) abspeichern, ein Verfahren, das *memoization* genannt wird. Bei genau diesem Verfahren, der memoization, tritt in einer GRAPHPLAN-Variante, IPP mit Ressourcen [Koehler, 1998], das Hyperquader-Überdeckungsproblem auf.

Die Ressourcen sind hierbei Entitäten in der modellierten Welt, die sich schlecht oder gar nicht durch logische Prädikate fassen lassen, etwa die verfügbare Menge Benzin oder die zeitliche Dauer von Aktionen. Zusätzlich zu den logischen Atomen hat man nun in Weltzuständen *Intervalle* von Werten, die die einzelnen Ressourcen annehmen können. Speichert man nun einen nicht erreichbaren Weltzustand (ein unlösbares Ziel auf einem Zeitschritt) ab, so bildet das Kreuzprodukt der möglichen Intervalle für die Ressourcen gerade einen Hyperquader. Für eine Menge $\mathcal{G}$ von logischen Atomen auf einem Zeitschritt haben wir also allgemein eine Menge $\mathcal{H}$ von abgespeicherten Hyperquadern, nämlich alle Ressourcenconstraints, die mit $\mathcal{G}$ unlösbar waren. Für einen neuen Zielzustand auf dem gleichen Zeitschritt, mit den gleichen logischen Zielatomen, ist nun genau dann bereits Unlösbarkeit nachgewiesen, wenn sein Ressourcenhyperquader Q von den Quadern in $\mathcal{H}$ überdeckt wird. Dies bedeutet nämlich, daß es keine Zuweisung von Werten an die Ressourcen gibt, die nicht bereits als unlösbar nachgewiesen wurde.

1.2 Vorgehensweise

Wir geben zunächst in Kapitel 2 einige grundlegende Definitionen; anhand dieser Definitionen wird dann die theoretische Komplexität des Problems untersucht. Anschließend verschaffen wir uns in Kapitel 3 einen Überblick über die verschiedenen in der Algorithmischen Geometrie zur Anwendung kommenden algorithmischen Prinzipien. Zwei dieser Prinzipien, das *Scan-line*- und das *Incremental Construction*-Prinzip werden zur Lösung des Überdeckungsproblems für Hyperquader ausgesucht. Die Verfahren, die durch Anwendung dieser beiden Prinzipien auf das Problem entstehen, werden in den Kapiteln 4 und 5 behandelt. In Kapitel 6 vergleichen wir die beiden Verfahren bezüglich ihrer Effizienz anhand einer empirischen Untersuchung. Kapitel 7 enthält eine Zusammenfassung der Resultate sowie einen Ausblick auf Anwendungsmöglichkeiten der Methodik.

[1] Weiterhin ist eine *Hintergrundtheorie* zugelassen, was aber hier keine Rolle spielt.

Kapitel 2

Theoretische Grundlagen

Wie wir in diesem Kapitel sehen werden, ist das Überdeckungsproblem für beliebig dimensionierte Hyperquader, wie es in dieser Arbeit behandelt wird, co-NP-vollständig. Zunächst definieren wir einige grundlegende Begriffe und geben eine formale Definition der Fragestellung.

2.1 Grundbegriffe und Konventionen

Wir präzisieren als erstes, was im Sinne dieser Arbeit unter einem *Intervall* zu verstehen ist.

Definition 1 *Ein Intervall I im Sinne dieser Arbeit ist ein abgeschlossenes, beschränktes Intervall über den rationalen Zahlen, d.h. zu I existieren rationale Zahlen $L, H \in \mathbb{Q}$, wobei $L \leq H$ mit: $I = [L, H] = \{x \in \mathbb{Q} \mid L \leq x \leq H\}$.*

Durch diese Definition ist es möglich, Intervalle als *Kanten* von Rechtecken aufzufassen, was die bevorzugte Sichtweise in dieser Arbeit sein wird. Man beachte auch, daß auf diese Art und Weise ein Intervall immer eine nicht-leere Menge ist. Haben wir für ein bestimmtes Intervall eine Bezeichnung, etwa Q, so bezeichnen wir die Endpunkte des Intervalls mit L und H, als Funktion vom Namen des Intervalls, also etwa $L(Q)$ und $H(Q)$. Wir definieren den Begriff *Hyperquader* wie folgt.

Definition 2 *Ein n-dimensionaler Hyperquader H ist definiert als n-faches Kreuzprodukt von Intervallen, d.h. zu H existieren n Intervalle I^d mit: $H = I^1 \times \ldots \times I^n = \{\overline{x} = (x^1, \ldots, x^n) \in \mathbb{Q}^n \mid \forall d \in \{1, \ldots, n\} : x^d \in I^d\}$.*

Unterscheiden wir zwischen *verschiedenen Hyperquadern*, so indizieren wir die Quader *unten*, wir verwenden also etwa die Bezeichnungen H_1 und H_2. Unterscheiden wir hingegen zwischen Intervallen oder sonstigen Objekten in *verschiedenen Dimensionen*, so indizieren wir *oben*, etwa I^1 und I^2.

Im eindimensionalen Fall sind Hyperquader einfach Intervalle, im zweidimensionalen Fall Rechtecke; allgemein ist es hilfreich, sich Hyperquader wie im dreidimensionalen Fall als Quader vorzustellen. Wenn die Dimension der Hyperquader aus dem Kontext hervorgeht, werden wir sie auch oft einfach als Quader bezeichnen. Wir definieren nun auf natürliche Art und Weise den Begriff *Überdeckung*.

Definition 3 *Gegeben sei eine Menge $\mathcal{H}$ von k n-dimensionalen Hyperquadern, $\mathcal{H} = \{H_1, \ldots, H_k\}$. Sei Q ein weiterer n-dimensionaler Hyperquader. Wir sagen, daß Q von $\mathcal{H}$ überdeckt wird, in Zeichen $Q \sqsubset \mathcal{H}$, falls gilt:*

$$Q \subseteq \bigcup_{i=1}^{k} H_i$$

Wir führen das Symbol $\sqsubset$ lediglich ein, um eine lesbare Notation zu gewährleisten. Handelt es sich bei $\mathcal{H}$ um eine ein-elementige Menge, also $\mathcal{H} = \{H_1\}$, so schreiben wir für $Q \sqsubset \{H_1\}$ auch kurz $Q \sqsubset H_1$. Haben wir eine Menge $\mathcal{H}$ von Mengen (etwa eine Menge von Hyperquadern) gegeben, so verwenden wir manchmal auch das Vereinigungssymbol $\bigcup$ ohne Indizierung. Vor $\mathcal{H}$ gestellt, ist damit die Vereinigung der Mengen in $\mathcal{H}$ gemeint; mit $\mathcal{H} = \{H_1, \ldots, H_k\}$ ist also $\bigcup \mathcal{H}$ eine Kurzschreibweise für $\bigcup_{i=1}^{k} H_i$.

Wir können nun eine formale Definition des behandelten Überdeckungsproblems angeben.

Definition 4 *Das Überdeckungsproblem für allgemeine Hyperquader,* QCOVER, *ist wie folgt definiert:*

Gegeben eine Menge $\mathcal{H} = \{H_1, \ldots, H_k\}$ von n-dimensionalen Hyperquadern, sowie einen n-dimensionalen Quader Q, wird Q von $\mathcal{H}$ überdeckt, d.h., gilt $Q \sqsubset \mathcal{H}$?

Wir bezeichnen die Quader $H_i \in \mathcal{H}$ als *Evidenzquader*, sowie den Quader Q als *Anfragequader*. Aus beweistechnischen Gründen definieren wir auch das zu QCOVER duale Problem.

Definition 5 *Das negative Überdeckungsproblem für allgemeine Hyperquader,* QNEGCOVER, *ist wie folgt definiert:*

Gegeben eine Menge $\mathcal{H} = \{H_1, \ldots, H_k\}$ von n-dimensionalen Hyperquadern, sowie einen n-dimensionalen Quader Q, wird Q **nicht** *von $\mathcal{H}$ überdeckt, d.h., gilt $Q \not\sqsubset \mathcal{H}$?*

Wir beweisen in den folgenden Abschnitten, daß QNEGCOVER NP-vollständig ist. Damit ist dann, siehe etwa [Garey and Johnson, 1979], die co-NP-Vollständigkeit von QCOVER gezeigt. Wir beginnen mit dem Beweis der NP-Härte.

2.2 NP-Härte

Wir beweisen die NP-Härte durch polynomiale Reduktion von 3SAT auf QNEGCOVER.

Die Grundidee des Beweises besteht darin, Variablenbelegungen mit gewissen Ausschnitten des n-dimensionalen Raumes zu identifizieren. Aus jeder Klausel kann man dann einen Raumausschnitt (einen Hyperquader) gewinnen, der genau *den* Belegungen entspricht, die die Klausel *nicht* erfüllen. Dann ist die Klauselmenge erfüllbar genau dann, wenn der Gesamtraum *nicht* von den Klauselquadern überdeckt wird.

Sei also eine 3SAT Instanz mit n Variablen $v_1, \ldots, v_n$ und k Klauseln $C_1, \ldots, C_k$ gegeben. Jede Klausel C_i hat die Form $C_i = l_{d_1} \vee l_{d_2} \vee l_{d_3}$ mit Literalen l_{d_j}, d.h., $l_{d_j} \in \{v_{d_j}, \neg v_{d_j}\}$.

Es bezeichne $\mathcal{A} := \{a \mid a = A^1 \times \ldots \times A^n,\ \forall d : A^d \in \{[-1,0],[0,1]\}\}$ die Menge aller *atomaren Hyperquader*, die aus den Intervallen $[-1,0]$ und $[0,1]$ gebildet werden können. Atomar bedeutet hier, daß diese Quader nicht weiter zerlegt werden können, bzw. müssen; diese Quader sind die *Bausteine*, aus denen im folgenden alle Hyperquader aufgebaut werden können. Wir werden den Begriff *atomar* in diesem Sinne noch häufiger verwenden.

Mit $\mathcal{B}$ bezeichnen wir die Menge aller Variablenbelegungen zu $\{v_1, \ldots, v_n\}$, also $\mathcal{B} := \{b : \{v_1, \ldots, v_n\} \to \{\perp, \top\}\}$. Offenbar kann $\mathcal{B}$ bijektiv auf $\mathcal{A}$ abgebildet werden, vermittels der Abbildung Φ, die wie folgt definiert ist:

$$\Phi(b) := \phi(v_1) \times \ldots \times \phi(v_n),\ \phi(v_d) := \begin{cases} \text{[-1,0]} & b(v_d) = \perp \\ \text{[0,1]} & b(v_d) = \top \end{cases} \tag{2.1}$$

Wir geben nun eine Reduktionsfunktion f von 3SAT auf QNEGCOVER an, indem wir die Klauseln C_i wie folgt auf Hyperquader abbilden:

$$f(l_{d_1} \vee l_{d_2} \vee l_{d_3}) := I^1 \times \ldots \times I^n,\ I^d := \begin{cases} \text{[-1,1]} & d \notin \{d_1, d_2, d_3\} \\ \text{[-1,0]} & d = d_j \wedge l_{d_j} = v_{d_j} \\ \text{[0,1]} & d = d_j \wedge l_{d_j} = \neg v_{d_j} \end{cases} \tag{2.2}$$

Man beachte hier, daß jeweils die dem Literal *entgegengesetzte* Seite des Raumes gewählt wird; intuitiv beschreibt also der entstehende Quader $f(l_{d_1} \vee l_{d_2} \vee l_{d_3})$ genau den Ausschnitt des Gesamtraumes, der von der Klausel *ausgeschlossen* wird. In Abbildung 2.1 sind die Klauselquader für ein einfaches Beispiel mit zwei Variablen und den Klauseln $C_1 = v_1$ sowie $C_2 = \neg v_1 \vee \neg v_2$ zu sehen.

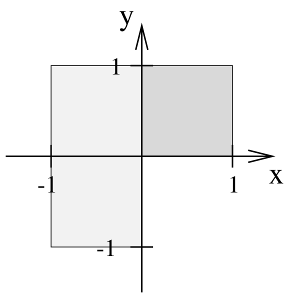

Abbildung 2.1: Die Klauselquader für die Klauseln v_1 (hell) und $\neg v_1 \vee \neg v_2$ (dunkel). Die Intervalle für v_1 sind auf der x-, die Intervalle für v_2 auf der y-Achse aufgetragen.

Wir geben nun noch eine weitere Hilfsnotation: Für einen beliebigen n-dimensionalen Quader $H = I^1 \times \ldots \times I^n$, der aus den Intervallen $[-1, 1]$, $[-1, 0]$ und $[0, 1]$ gebildet wird (d.h. $I^d \in \{[-1, 1], [-1, 0], [0, 1]\}$), definieren wir $\overline{H}$ als die Menge der atomaren Hyperquader, die von H überdeckt werden, also $\overline{H} := \{a \in \mathcal{A} \mid a \sqsubset H\}$. Es gilt dann:

$$\overline{H} = \{a \in \mathcal{A} \mid a = A^1 \times \ldots \times A^n,\ A^d \in \begin{cases} \{\text{[-1,0], [0,1]}\} & I^d = [-1, 1] \\ \{\text{[-1,0]}\} & I^d = [-1, 0] \\ \{\text{[0,1]}\} & I^d = [0, 1] \end{cases} \} \tag{2.3}$$

In dem Beispiel aus Abbildung 2.1 ist etwa $\overline{f(C_1)} = \{[-1, 0] \times [-1, 0], [-1, 0] \times [0, 1]\}$, was lediglich besagt, daß sich der Klauselquader zu C_1 aus den beiden linken Kästchen zusammensetzt: diese Kästchen stehen für Abbildungen, die v_1 mit $\perp$ belegen, die also C_1 nicht erfüllen. Es gilt das folgende Lemma.

Lemma 1 *Mit den obigen Bezeichnungen gilt folgende Gleichung:*

$$\Phi(\{b \in \mathcal{B} \mid b \not\models l_{d_1} \vee l_{d_2} \vee l_{d_3}\}) = \{a \in \mathcal{A} \mid a \sqsubset f(l_{d_1} \vee l_{d_2} \vee l_{d_3})\}$$

Beweis:
Es bezeichne $M_{\mathcal{B}} := \{b \in \mathcal{B} \mid b \not\models l_{d_1} \vee l_{d_2} \vee l_{d_3}\}$ die Menge auf der linken Seite obiger Gleichung. Man beachte, daß die Menge auf der rechten Seite genau die von $f(l_{d_1} \vee l_{d_2} \vee l_{d_3})$ überdeckten atomaren Hyperquader sind, also:

$$\{a \in \mathcal{A} \mid a \sqsubset f(l_{d_1} \vee l_{d_2} \vee l_{d_3})\} = \overline{f(l_{d_1} \vee l_{d_2} \vee l_{d_3})} \tag{2.4}$$

Wie üblich bezeichnen wir für ein Literal l sein Negat mit $\overline{l}$, also

$$\overline{l} := \begin{cases} v_d & l = \neg v_d \\ \neg v_d & l = v_d \end{cases}$$

Wir können nun $\Phi(M_{\mathcal{B}})$ wie folgt umformen:

$$\begin{aligned}
\Phi(M_{\mathcal{B}}) &= \Phi(\{b \in \mathcal{B} \mid b \not\models l_{d_1} \vee l_{d_2} \vee l_{d_3}\}) \\
&= \Phi(\{b \in \mathcal{B} \mid b \not\models \neg(\overline{l_{d_1}} \wedge \overline{l_{d_2}} \wedge \overline{l_{d_3}})\}) \\
&= \Phi(\{b \in \mathcal{B} \mid b \models \overline{l_{d_1}} \wedge \overline{l_{d_2}} \wedge \overline{l_{d_3}}\}) \\
&= \Phi(\{b \in \mathcal{B} \mid b(v_d) \in \begin{cases} \{\bot, \top\} & d \notin \{d_1, d_2, d_3\} \\ \{\bot\} & d = d_j \wedge l_{d_j} = v_{d_j} \\ \{\top\} & d = d_j \wedge l_{d_j} = \neg v_{d_j} \end{cases} \}) \\
&= \{a \in \mathcal{A} \mid a = A^1 \times \ldots \times A^n, A^d \in \begin{cases} \{[\text{-}1,0],\ [0,1]\} & d \notin \{d_1, d_2, d_3\} \\ \{[\text{-}1,0]\} & d = d_j \wedge l_{d_j} = v_{d_j} \\ \{[0,1]\} & d = d_j \wedge l_{d_j} = \neg v_{d_j} \end{cases} \} \\
&= \overline{f(l_{d_1} \vee l_{d_2} \vee l_{d_3})}
\end{aligned}$$

Hierbei folgt die letzte Gleichheit direkt aus der Definition 2.2 von f und Gleichung 2.3. Mit Gleichung 2.4 folgt die Behauptung. ■

Die gesamte QNEGCOVER Instanz ergibt sich nun, indem aus jeder Klausel C_i vermittels f ein Evidenzquader $H_i = f(C_i)$ gebildet wird; als Anfragequader wählen wir das n-fache Kreuzprodukt des Grundintervalls, d.h., $Q := [-1,1] \times \ldots \times [-1,1] = [-1,1]^n$. Intuitiv modelliert Q den gesamten Lösungsraum; wird dieser von den Evidenzquadern überdeckt, so bedeutet dies, daß alle Lösungen ausgeschlossen sind und die Klauselmenge somit unerfüllbar ist. Genauer gilt der folgende Satz.

Satz 1 *Mit den obigen Bezeichnungen gilt:*

$C_1 \wedge \ldots \wedge C_n$ *ist erfüllbar genau dann, wenn* $Q \not\sqsubset \{f(C_1), \ldots, f(C_k)\}$.

Beweis:
Es bezeichne $M_\mathcal{B} := \{b \in \mathcal{B} \mid b \not\models C_1 \wedge \ldots \wedge C_n\}$ die Menge aller Belegungen, die die Klauseln *nicht* erfüllen. Weiter bezeichne $M_\mathcal{A} := \{a \in \mathcal{A} \mid a \sqsubset \{f(C_1), \ldots, f(C_n)\}\}$ die Menge aller atomaren Quader, die durch die Evidenzquader überdeckt werden.
Wir zeigen:

$$\Phi(M_\mathcal{B}) = M_\mathcal{A} \tag{2.5}$$

Damit gilt dann:

$$\begin{aligned}
\Phi(\{b \in \mathcal{B} \mid b \models C_1 \wedge \ldots \wedge C_n\}) &= \Phi(\mathcal{B} \setminus M_\mathcal{B}) \\
&= \Phi(\mathcal{B}) \setminus \Phi(M_\mathcal{B}) \qquad \textit{da } \Phi \textit{ Bijektion} \\
&= \mathcal{A} \setminus \Phi(M_\mathcal{B}) \\
&= \mathcal{A} \setminus M_\mathcal{A} \qquad \textit{mit Gleichung } 2.5
\end{aligned} \tag{2.6}$$

Die linke Seite der Behauptung des Satzes ist offenbar äquivalent zu: $\{b \in \mathcal{B} \mid b \models C_1 \wedge \ldots \wedge C_n\} \neq \emptyset$. Die rechte Seite ist offenbar genau dann erfüllt, wenn gilt: $\exists a \in \mathcal{A} : a \not\sqsubset \{f(C_1), \ldots, f(C_n)\}$. Dies ist aber genau dann der Fall, wenn $\mathcal{A} \setminus M_\mathcal{A} \neq \emptyset$. Mit Gleichung 2.6 folgt also die Gesamtbehauptung. Zu zeigen bleibt noch Gleichung 2.5. Es gilt:

$$\begin{aligned}
\Phi(M_\mathcal{B}) &= \Phi(\textstyle\bigcup_{i=1}^n \{b \in \mathcal{B} \mid b \not\models C_i\}) \\
&= \textstyle\bigcup_{i=1}^n \Phi(\{b \in \mathcal{B} \mid b \not\models C_i\}) \quad \textit{da } \Phi \textit{ Bijektion} \\
&= \textstyle\bigcup_{i=1}^n \{a \in \mathcal{A} \mid a \sqsubset f(C_i)\} \quad \textit{mit Lemma } 1 \\
&= M_\mathcal{A}
\end{aligned}$$

Damit ist alles bewiesen. ∎

Im Beispiel aus Abbildung 2.1 etwa überdecken die Klauselquader alles bis auf die rechte untere Ecke des Koordinatensystems. Dieses Kästchen steht für die Belegung $b(v_1) = \top, b(v_2) = \bot$, was ja auch die einzige für $C_1 = v_1$ und $C_2 = \neg v_1 \vee \neg v_2$ erfüllende Belegung ist.

Da die Größe der QNEGCOVER Instanz offensichtlich polynomial in der Größe der 3SAT Instanz ist, haben wir gezeigt: 3SAT $\leq_p$ QNEGCOVER. Es gilt also das folgende Korollar.

Korollar 1 QNEGCOVER *ist* NP*-hart.*

2.3 Diskretisierung und NP-Mitgliedschaft

In diesem Abschnitt führen wir den Begriff der *Diskretisierung* einer QCOVER Instanz ein, der für das Verständnis der QCOVER Problematik sowie einiger Lösungsansätze von zentraler Bedeutung ist. Als Korollar daraus erhalten wir einen einfachen Beweis für die NP-Mitgliedschaft von QNEGCOVER.

Sei eine QCOVER Instanz durch $\mathcal{H} = \{H_1, \ldots, H_k\}$ und Q gegeben. Jeder Evidenzquader H_i ist ein Kreuzprodukt von Intervallen, d.h. $H_i = I_i^1 \times \ldots \times I_i^n$, ebenso $Q = Q^1 \times \ldots \times Q^n$.

Die entscheidende Beobachtung ist nun die folgende: Jede Dimension d (die man sich als *Achse* vorstellen kann) wird durch *die Endpunkte der Intervalle aller Quader* in d in endlich viele *Abschnitte* unterteilt. Präziser betrachten wir in jeder Dimension die folgende Menge:

$$\{P_1^d, \ldots, P_{m^d}^d\} := \{L(I_1^d), H(I_1^d), \ldots, L(I_k^d), H(I_k^d), L(Q^d), H(Q^d)\} \tag{2.7}$$

Die Punkte P_i^d bezeichnen also gerade die oberen und unteren Endpunkte der k Evidenzquader sowie des Anfragequaders der Instanz in Dimension d. Die Anzahl m^d dieser Punkte hängt von d ab, da in jeder Dimension unterschiedlich viele Endpunkte zusammenfallen können. Wir gehen davon aus, daß die Punkte P_i^d aufsteigend sortiert sind, also $P_i^d < P_j^d$ für $i < j$ gilt, und betrachten nun die *atomaren* Intervalle $A_i^d := [P_i^d, P_{i+1}^d]$.

Unsere Absicht dabei ist es, eine Menge *kleinster Bausteine* der Instanz zu identifizieren, also eine Menge von Quadern, aus denen alle Evidenz- sowie der Anfragequader zusammengesetzt werden können. Wir verwenden dazu Quader, die sich als *Kreuzprodukt von atomaren Intervallen* darstellen lassen. Bis jetzt enthält die Menge $\mathcal{A}^d$ der atomaren Intervalle die oben eingeführten Abschnitte: $\mathcal{A}^d := \{A_1^d, \ldots, A_{m^d}^d\}$. Dies reicht aber im allgemeinen noch nicht aus: Ist das Intervall eines beliebigen Quaders $H = I^1 \times \ldots \times I^n$ der Instanz in einer Dimension d lediglich ein *Punkt*, d.h. $L(I^d) = H(I^d)$, so kann dieses Intervall aus den bisherigen Intervallen in $\mathcal{A}^d$ nicht aufgebaut werden, ein einzelner Punkt wird beim Bilden der Abschnitte einfach übersehen. Wir nehmen deshalb jedes solche Intervall I^d der Instanz zusätzlich als atomares Intervall zu $\mathcal{A}^d$ hinzu, setzen also jeweils $\mathcal{A}^d := \mathcal{A}^d \cup \{I^d\}$. Die Diskretisierung der Instanz ist nun wie folgt definiert.

Definition 6 *Es sei eine* QCOVER *oder* QNEGCOVER *Instanz gegeben. Mit den obigen Bezeichnungen ist ihre Diskretisierung* $\mathcal{D}$ *definiert als die Menge:*

$$\mathcal{D} := \{a \mid a = A^1 \times \ldots \times A^n, \ \forall d : A^d \in \mathcal{A}^d\}$$

In Abbildung 2.2 ist die Diskretisierung einer Instanz mit lediglich einem Evidenzquader zu sehen.

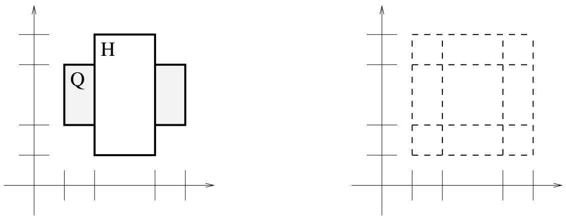

Abbildung 2.2: Diskretisierung einer zweidimensionalen Instanz mit nur zwei Rechtecken. Links sind die beiden Rechtecke sowie die entstehende Unterteilung der Achsen, rechts die Diskretisierungsrechtecke abgebildet.

Offensichtlich gibt es in jeder Dimension höchstens $2k+2$ verschiedene Intervallendpunkte, d.h., m_d ist durch $2k + 2$ nach oben beschränkt. Damit gibt es auch höchstens $2k + 1$ Abschnitte in $\mathcal{A}^d$. Ist ein Intervall lediglich ein Punkt, so wird ein zusätzliches Intervall zu $\mathcal{A}^d$ hinzugenommen; andererseits fallen dann aber auch mindestens zwei Punkte in d zusammen, es gibt also einen Abschnitt weniger. Insgesamt können wir somit folgern, daß für alle Dimensionen $|\mathcal{A}^d| \leq 2k + 1$ gilt; für die Diskretisierung folgt damit sofort

$$|\mathcal{D}| \leq (2k + 1)^n \tag{2.8}$$

Im worst-case, wenn zum Beispiel die Endpunkte der Intervalle in jeder Dimension paarweise verschieden sind, gilt hier Gleichheit, also $|\mathcal{D}| = (2k+1)^n$.

Es gilt nun tatsächlich, daß sich alle Quader der Instanz aus den Quadern in $\mathcal{D}$, die wir künftig auch als *Diskretisierungsquader* bezeichnen, aufbauen lassen. Präziser definieren wir, ähnlich wie im vorangehenden Abschnitt, für einen beliebigen Quader H der Instanz (d.h., für einen Evidenz- oder den Anfragequader) die Menge $\overline{H}$ als die Menge der Diskretisierungsquader, die von H überdeckt werden, also $\overline{H} := \{a \in \mathcal{D} \mid a \sqsubset H\}$. Damit gilt dann

$$H = \bigcup \overline{H}. \tag{2.9}$$

Wir verzichten hier auf einen formalen Beweis dieser Feststellung und geben stattdessen eine grobe Beweisskizze wie folgt: Daß $\bigcup \overline{H} \subseteq H$ gilt, ist klar, da per definitionem alle Quader aus $\overline{H}$ in H enthalten sind. Zu zeigen ist also nur $H \subseteq \bigcup \overline{H}$. Ist nun ein Punkt $\overline{x}$ aus H gegeben, so liegt die Koordinate dieses Punktes in jeder Dimension d in einem Intervall aus $\mathcal{A}^d$, unter Umständen können das auch mehrere Intervalle sein. Es ist nicht schwer zu sehen, daß mindestens eines der Kreuzprodukte dieser Intervalle von H überdeckt wird, also in $\overline{H}$ liegt. Damit ist dann die Argumentation abgeschlossen.

Die Vereinigung $\bigcup \overline{H}$ ist formal nicht disjunkt, da die Diskretisierungsquader an den Rändern überlappen (wir verwenden ja ausschließlich abgeschlossene Intervalle); es ist aber hilfreich, sich die Quader als disjunkte Bausteine vorzustellen: auch alle Schnitte und Vereinigungen von Quadern der Instanz können aus ihnen aufgebaut werden. Intuitiv enthält die Diskretisierung alle Information über den Raum, die für eine Instanz relevant ist. Ihre entscheidende Eigenschaft ist, daß sie, im Gegensatz zu $\mathbb{Q}^n$, endlich ist, wenn auch exponentiell groß, siehe Gleichung 2.8.

Wir bezeichnen die Menge $\overline{Q}$, also die Menge der Diskretisierungsquader, aus denen der Anfragequader aufgebaut ist, künftig als *Anfragediskretisierung* und zeigen zunächst ein einfaches Lemma.

Lemma 2 *Es sei eine* QCOVER *oder* QNEGCOVER *Instanz gegeben. Mit den obigen Bezeichnungen gilt:*

Der Anfragequader wird genau dann überdeckt, wenn alle Quader der Anfragediskretisierung überdeckt werden.

Beweis:
Nach Voraussetzung gilt $Q \subseteq \bigcup \mathcal{H}$. Da mit Gleichung 2.9 $Q = \bigcup \overline{Q}$ gilt, ist das gleichbedeutend mit $\bigcup \overline{Q} \subseteq \bigcup \mathcal{H}$. Dies wiederum ist genau dann der Fall, wenn $\forall a \in \overline{Q} : a \subseteq \bigcup \mathcal{H}$ erfüllt ist, sprich, wenn alle Quader der Anfragediskretisierung überdeckt werden. ■

Der entscheidende Vorteil, den wir aus dieser Feststellung ziehen, ist dieser: wird ein Diskretisierungsquader überdeckt, so ist er bereits in *einem* Evidenzquader ganz enthalten. Dies folgt einfach aus der Definition der Diskretisierung: die Diskretisierungsquader sind bereits die kleinsten Bausteine und werden durch Quader der Instanz nicht mehr weiter unterteilt. Wir halten fest:

$$\forall a \in \mathcal{D} : a \sqsubset \mathcal{H} \Rightarrow \exists H \in \mathcal{H} : a \sqsubset H \tag{2.10}$$

Wir kommen auf diesen Sachverhalt im Kontext von Lemma 3 (siehe unten) noch einmal zu sprechen und beweisen zunächst die NP-Mitgliedschaft von QNEGCOVER durch ein einfaches *guess and check* Argument:

- Rate nicht-deterministisch alle Quader der Anfragediskretisierung.
- Überprüfe für diese Quader, ob sie in einem der Evidenzquader enthalten sind; dazu muß man lediglich alle Evidenzquader der Reihe nach abarbeiten und in jeder Dimension prüfen, ob das Intervall des Diskretisierungsquaders ganz im entsprechenden Intervall des Evidenzquaders enthalten ist. Die Zeitkomplexität ist also $O(kn)$, d.h., polynomial.
- Wird *ein* Diskretisierungsquader *nicht* überdeckt, so antworte $\perp$.

Mit Lemma 2 und Aussage 2.10 wissen wir, daß dies ein gültiger Lösungsalgorithmus für QNEGCOVER ist. Wir können also eine nicht-deterministische Turingmaschine angeben, die QNEGCOVER in polynomialer Zeit löst. Zusammen mit Korollar 1 folgt sofort:

Korollar 2 QNEGCOVER *ist* NP*-vollständig.*

Per definitionem ist damit die co-NP-Vollständigkeit des dualen Problems QCOVER gezeigt.

Korollar 3 QCOVER *ist* co-NP*-vollständig.*

Abschließend, um die Natur der Diskretisierungsquader noch etwas genauer kennenzulernen, sowie um Aussage 2.10 zu untermauern, zeigen wir noch ein Lemma über die *Mittelpunkte* der Diskretisierungsquader. Dabei ist der Mittelpunkt $M(a)$ eines Diskretisierungsquaders $a = A^1 \times \ldots \times A^n$ als

$$M(a) := (\frac{L(A^1) + H(A^1)}{2}, \ldots, \frac{L(A^n) + H(A^n)}{2}) \tag{2.11}$$

definiert. Der Diskretisierungsquader wird nun genau dann überdeckt, wenn das für seinen Mittelpunkt der Fall ist, d.h., es gilt folgendes Lemma.

Lemma 3 *Mit den obigen Bezeichnungen gilt für jeden Diskretisierungsquader* $a = A^1 \times \ldots \times A^n \in \mathcal{D}$*:*

$$a \sqsubset \mathcal{H} \Leftrightarrow \exists H \in \mathcal{H} : M(a) \in H$$

Beweis:
$\Rightarrow$: Mit $a \subseteq \bigcup \mathcal{H}$ gilt natürlich auch $M(a) \in \bigcup \mathcal{H}$. Da der Mittelpunkt $M(a)$ ein einzelner Punkt ist, ist er bereits in einem einzigen $H \in \mathcal{H}$ enthalten.

$\Leftarrow$: Es gelte etwa $M(a) \in H = I^1 \times \ldots \times I^n$; es genügt zu zeigen, daß die Intervalle von H in jeder Dimension das Intervall von a umfassen, also $A^d \subseteq I^d$ gilt. Nach Definition von $M(a)$ haben wir in jeder Dimension d: $\frac{L(A^d)+H(A^d)}{2} \in I^d$. Wir unterscheiden je Dimension d zwei Fälle:

1. Das Intervall von a ist ein Punkt, d.h., $L(A^d) = H(A^d)$. Dann ist $A^d = \{L(A^d)\} = \{\frac{L(A^d)+H(A^d)}{2}\} \subseteq I^d$ und wir sind fertig.

2. Es ist $L(A^d) < H(A^d)$. Da $\frac{L(A^d)+H(A^d)}{2} \in I^d$ gilt, umfaßt I^d bereits das ganze Intervall; denn nach Definition der Diskretisierungsquader können die Intervallgrenzen von I^d nicht *in* das Intervall A^d hineinfallen. Es gilt also $L(I^d) \leq L(A^d)$ und $H(I^d) \geq H(A^d)$, also wie gewünscht $A^d \subseteq I^d$.

Damit ist gezeigt, daß $a \sqsubset H$ gilt. Insbesondere gilt also auch $a \sqsubset \mathcal{H}$. ■

Die entscheidende Eigenschaft, die der Mittelpunkt eines Diskretisierungsquaders hat, ist, daß er ein *innerer* Punkt ist. Tatsächlich können *Randpunkte* eines Quaders überdeckt werden, ohne daß der ganze Quader überdeckt wird. Das ist intuitiv völlig klar: wegen der abgeschlossenen Intervalle können benachbarte Quader Randpunkte überdecken. Um genau zu sein, ist für Randpunkte das Argument im zweiten Fall der Rückrichtung obigen Lemmas ungültig. Für beliebige innere Punkte hingegen bleibt die Gültigkeit des Lemmas erhalten.

Wir erhalten aus Lemma 3 auch einen Beweis für Aussage 2.10: wird ein Diskretisierungsquader überdeckt, gilt also $a \sqsubset \mathcal{H}$, so gibt es mit Lemma 3 einen Evidenzquader $H \in \mathcal{H}$ mit $M(a) \in H$. Im Beweis zu Lemma 3 wird gezeigt, daß dann für diesen Quader bereits $a \sqsubset H$ gilt.

Kapitel 3

Verwandte Arbeiten

Wie bereits erwähnt, ist, trotz unzähliger Veröffentlichungen auf diesem Gebiet, in der Algorithmischen Geometrie keine Arbeit zu finden, die sich *direkt* mit der Überdeckung von Hyperquadern beschäftigt. Dies mag daran liegen, daß die meisten geometrischen Probleme in höchstens drei Dimensionen definiert sind. Im zweidimensionalen Fall sind Hyperquader nun einfach Rechtecke, im eindimensionalen Fall sogar lediglich kolineare Liniensegmente. Dadurch werden viele der Einzelheiten, auf die wir im Verlauf der Diskussion in dieser Arbeit stoßen werden, aus Sicht der klassischen Algorithmischen Geometrie uninteressant, etwa was den Schnitt zweier Hyperquader betrifft: Das Schneiden zweier Rechtecke kann in konstanter Zeit durchgeführt werden.

Algorithmen, die in der Literatur zu finden sind, befassen sich daher für gewöhnlich mit weitaus komplexeren Objekten, etwa mit Mengen beliebiger Geraden; die behandelten Probleme werden durch die innere und gegenseitige Struktur der untersuchten Objekte interessant. Im Gegensatz dazu beziehen Hyperquader ihre Komplexität einzig und allein aus der unbeschränkten Zahl von Dimensionen. Nichtsdestotrotz können viele algorithmische Prinzipien, die in der Algorithmischen Geometrie zur Anwendung kommen, auch auf das vorliegende Problem angewandt werden.

3.1 Algorithmische Prinzipien

In [Lee and Preparata, 1984] wird eine Übersicht der wichtigsten Methoden zum Lösen geometrischer Probleme gegeben. Es werden sieben Prinzipien unterschieden:

- **Incremental Construction**: Die Grundidee besteht hier darin, die Berechnung inkrementell vorzunehmen, also etwa aus einer Menge von geometrischen Objekten jeweils das nächste zu betrachten. Ein einfaches Beispiel ist die Berechnung der konvexen Hülle einer Menge von Punkten, die wie folgt durchgeführt werden kann: Nimm jeweils einen Punkt zu den bereits ausgewählten hinzu; ist der Punkt bereits von der bisherigen konvexen Hülle überdeckt, so ist nichts zu tun, andernfalls berechne die beiden neu hinzu kommenden Kanten der Hülle.

- **Plane Sweep**: Dieses Verfahren ist auch unter dem Namen *Scan-line* Mehode bekannt. Die Idee ist im zweidimensionalen Fall intuitiv am einfachsten einzusehen: Man schwenkt eine vertikale Linie von links nach rechts über die Ebene; dabei hält man immer die jeweils aktuelle Information über das zu lösende Problem aufrecht.

Auf diese Art kann man ein statisches zweidimensionales Problem auf eine dynamische Folge eindimensionaler Probleme zurückführen.

- **Locus**: Hier versucht man, den Gesamtraum in *Zellen* aufzuteilen, deren Inhaltspunkte im Hinblick auf das Problem äquivalent sind; jede solche Zelle kann dann durch *einen* ihrer Punkte repräsentiert werden. Dadurch werden klassische Suchbaumverfahren anwendbar, da nur noch endlich viele Punkte zu betrachten sind.

- **Divide and Conquer**: Diese algorithmische Methode beruht darauf, das Problem aufzuteilen und die Unterprobleme rekursiv auf die gleiche Weise zu lösen (divide), sowie die Gesamtlösung aus den Teillösungen zusammenzusetzen (conquer). Es gibt viele Anwendungen in der Geometrie, etwa das Berechnen der konvexen Hülle einer Menge von Punkten; die Divide and Conquer Technik führt dafür in sehr natürlicher Weise zu einer effizienteren Lösung als die inkrementelle Methode.

- **Geometric Transformations**: Durch Transformation der Fragestellung, d.h., durch die bijektive Abbildung der behandelten Objekte auf andere Objekte, kann manchmal eine intuitivere Darstellung des Problems erreicht werden. In [Edelsbrunner, 1987] wird etwa gezeigt, wie man feststellen kann, ob es für n Liniensegmente eine Gerade gibt, die alle diese Segmente schneidet. Kurz zusammengefaßt, werden dort Punkte auf Geraden und umgekehrt transformiert; alle Geraden, die ein Segment schneiden, entsprechen dann einer bestimmten Fläche. Der Schnitt aller dieser Flächen enthält genau die Punkte, deren zugehörige Geraden alle Segmente schneiden. Damit ist das ursprüngliche Problem auf ein Flächenschnittproblem zurückgeführt.

- **Prune and Search**: Diese Methode beruht darauf, die für ein Problem relevanten Daten kontinuierlich zu reduzieren. Man kann die Vorgehensweise etwa wie folgt beschreiben: Führe gerade soviele Berechnungen durch, wie nötig sind, um einen bestimmten Bruchteil α der Daten als irrelevant zu identifizieren; entferne diese Daten (prune) und wende das Verfahren erneut an (search). Da nun die Anzahl der Daten in der Anzahl der Durchläufe geometrisch abnimmt, benötigt das Gesamtverfahren asymptotisch genauso viel Zeit wie *ein* Durchlauf. Zum Beispiel kann auf diese Weise auch obiges Liniensegment-Schnittproblem gelöst werden.

- **Dynamization Technique**: Hierbei handelt es sich weniger um einen geometrischen Ansatz, als vielmehr um eine allgemeine Methodik, eine statische Datenstruktur zu *dynamisieren*. Die Datenstruktur wird also dahingehend modifiziert, daß dynamische insert- und delete- Aktionen ausgeführt werden können. Dies geschieht ohne wesentliche Einbußen bei der Laufzeit zur Beantwortung von Anfragen etc. Anwendbar ist die Technik auf Probleme, die in einem gewissen Sinn *dekomponierbar* sind, d.h., in Teilprobleme aufgespalten werden können.

3.2 Anwendbarkeit auf QCOVER

Wir betrachten nun die verschiedenen Prinzipien aus der algorithmischen Geometrie bezüglich ihrer Anwendbarkeit auf QCOVER.

3.2.1 Incremental Construction

Eine Idee, das QCOVER Problem zu lösen, indem man nach und nach Informationen dazunimmt, ist die folgende:

- Arbeite der Reihe nach alle Evidenzquader ab;
- *Subtrahiere* jeweils den aktuellen Evidenzquader vom Anfragequader;
- Der Anfragequader wird genau dann überdeckt, wenn am Ende nichts mehr von ihm *übrig* ist.

Der entscheidende Schritt hierbei ist das Subtrahieren eines Evidenzquaders vom Anfragequader. Das Ergebnis dieser Subtraktion ist im allgemeinen kein Quader mehr, es müßte also eine Methode gefunden werden, den Rest nach Abzug eines Evidenzquaders darzustellen.

3.2.2 Plane Sweep

Wie wir aus Lemma 2 wissen, genügt es zur Lösung von QCOVER, die Quader der Anfragediskretisierung zu betrachten. Aus Lemma 3 wiederum wissen wir, daß nur deren Mittelpunkte betrachtet werden müssen. Der Gedanke liegt also nahe, diese Mittelpunkte als *Haltepunkte* für eine Scan-line zu verwenden.

Merkt man sich etwa an jedem Haltepunkt die Menge der Evidenzquader, die gerade von der Scan-line geschnitten werden, so könnte man auf diese Art herausfinden, ob es einen Quader der Anfragediskretisierung gibt, dessen Mittelpunkt in *keinem* Evidenzquader enthalten ist; genau dann wird Q ja, gemäß den Erkenntnissen aus Abschnitt 2.3, nicht überdeckt.

3.2.3 Locus

Wiederum sieht man die Anwendbarkeit der Methode unter Bezugnahme auf die Diskretisierung der Instanz: Die Diskretisierungsquader sind ja gerade die gesuchten Zellen, deren Inhaltspunkte in Bezug auf das vorliegende Problem äquivalent sind. Man kann also etwa jeden Diskretisierungsquader durch seinen Mittelpunkt repräsentieren.

Es gibt geometrische Verfahren, die folgende Operation unterstützen: Organisiere zunächst eine endliche Menge von Punkten im Raum als spezielle Suchstruktur. Gegeben einen Hyperquader, finde dann in kurzer Zeit alle Punkte in der Struktur, die innerhalb des Hyperquaders liegen. Dadurch werden zwei Verfahren zur Lösung von QCOVER vorstellbar:

1. Speichere die Mittelpunkte aller Quader der Anfragediskretisierung in einer geomtrischen Suchstruktur S. Gehe dann (ähnlich wie beim Incremental Construction Ansatz) der Reihe nach die Evidenzquader durch und entferne diejenigen Punkte aus S, die im aktuellen Evidenzquader enthalten sind. Q wird genau dann überdeckt, wenn S am Ende leer ist.

2. Speichere die Mittelpunkte aller *Evidenzquader* in einer geometrischen Suchstruktur S. Bestimme dann die Anzahl m der Punkte in S, die in Q enthalten sind. Man kann leicht ausrechnen, wie viele Diskretisierungsquader von Q überdeckt werden. Ist m gleich dieser Anzahl, so wird Q überdeckt, andernfalls nicht.

Ein Unterschied zwischen den beiden Lösungsansätzen ist, daß man im ersten Fall eine *dynamische*, im zweiten Fall lediglich eine *statische* Suchstruktur benötigt.

3.2.4 Divide and Conquer

Auch die Anwendbarkeit des Divide and Conquer Ansatzes ist relativ klar: Man könnte den Anfragequader entlang der Grenzen der Diskretisierungsquader aufspalten, also etwa wie folgt vorgehen:

- Ist der aktuelle Anfragequader ein Diskretisierungsquader, so überprüfe, ob dieser Quader überdeckt wird und antworte entsprechend.
- Teile den Anfragequader entlang der aktuellen Dimension in zwei Teile auf, so daß beide Teile gleich viele Diskretisierungsquader enthalten.
- Rufe die Routine für beide Teile mit einer beliebigen Spaltungsdimension auf; wird mindestens einer der Teile nicht überdeckt, so antworte mit $\perp$, sonst antworte mit $\top$.

Wie bereits in Abschnitt 2.3 erwähnt, kann für einen Diskretisierungsquader recht schnell beantwortet werden, ob er überdeckt wird. Die nächste Spaltungsdimension ist prinzipiell frei wählbar, es sollte aber Sorge getragen werden, daß alle Dimensionen gleichmäßig berücksichtigt werden.

3.3 Auswahl der Lösungsverfahren

Im vorangehenden Abschnitt wurden die Methoden *Geometric Transformations*, *Prune and Search* sowie *Dynamization Technique* nicht erwähnt. Das hat folgende Gründe:

- Quader sind bereits sehr intuitive und einfache Objekte. Es ist daher nicht klar, inwiefern man aus einer geometrischen Transformation, also aus dem *Geometric Transformations* Ansatz, einen Vorteil ziehen könnte.
- Auch die Anwendbarkeit des *Prune and Search* ist in Frage zu stellen: Die einzig irrelevante Information in einer Instanz sind diejenigen Evidenzquader, die einen leeren Schnitt mit dem Anfragequader haben und daher für eine Überdeckung nicht in Betracht kommen. Diese Quader können aber mit einem einzigen Durchlauf, d.h., durch einen *Preprocessing*-Schritt, entfernt werden.
- Bei der *Dynamization Technique* geht es ja nicht um eine algorithmische Methode, sondern darum, eine vorhandene Datenstruktur zu dynamisieren. Dies kann zwar bei Verwendung des *Locus* Ansatzes von Nutzen sein (siehe Abschnitt 3.2.3), kann aber zur direkten Lösung von QCOVER offensichtlich nicht herangezogen werden.

Wir haben also noch aus vier verschiedenen Ansatzmöglichkeiten auszuwählen. Betrachten wir zunächst die *Locus* Methode.

In [Mehlhorn, 1984] wird eine Datenstruktur, die sogenannten *range trees*, eingeführt, die für Anfragen der folgenden Art verwendet werden können:

Gegeben eine (endliche) Menge S von Punkten in $\mathbb{Q}^n$, die als range tree abgespeichert ist, sowie einen Hyperquader H, finde alle Punkte aus S, die in H liegen.

Eine solche Datenstruktur könnte man also verwenden, um die beiden Ansätze für die *Locus* Methode zu realisieren. Allerdings ist in beiden Fällen die zu erwartende Laufzeit nicht besonders ermutigend: eine Anfrage an einen n-dimensionalen *range tree*, in dem $|S|$ Punkte gespeichert sind, kann in Zeit $O((log|S|)^n)$ bewerkstelligt werden. Nun ist bei beiden Ansätzen $|S|$ nur durch die Anzahl der Diskretisierungsquader nach oben beschränkt, also gemäß Gleichung 2.8 $|S| \leq (2k+1)^n$. Wir erhalten also für *eine* Anfrage eine Abschätzung von $O((2kn+n)^n)$.

Beim ersten Ansatz, wo zunächst Q als *range tree* abgelegt wird und dann die Punkte, die in den Evidenzquadern enthalten sind, nach und nach gestrichen werden, müssen k solcher Anfragen durchgeführt werden, hinzu kommen noch die Kosten für die *delete*-Operationen. Dieser Ansatz ist also völlig abzulehnen. Jedoch auch der zweite Ansatz, der schon um einiges vielversprechender aussieht, ist aufgrund der Größe der Diskretisierung nicht empfehlenswert. Zusätzlich zur Anfragezeit müssen ja zunächst einmal alle Evidenzquader als *range tree* organisiert werden, was auch, um die gewünschte Anfragezeit zu gewährleisten, mit Balancierungsoperationen etc. verbunden ist.[1] Zudem benötigen *range trees* auch einigen Speicherplatz: in unserem Fall wäre der Speicherbedarf lediglich durch $O((2k+1)^n(2kn+n)^{n-1})$ nach oben abzuschätzen.

Es bleiben also noch drei Ansätze zur Auswahl: *Incremental Construction*, *Plane Sweep* und *Divide and Conquer*. Bei allen drei Methoden ist es denkbar, daß sie zu einer (vergleichsweise) effizienten Lösung führen können. Wir entwickeln in Kapitel 4 einen Algorithmus (namens Continuous Subtraction), der den *Incremental Construction* Ansatz realisiert. In Kapitel 5 stellen wir den Dimension Scanning Algorithmus vor, der auf einer Erweiterung der *Plane Sweep* Methodik ins n-dimensionale beruht.

Haupsächlich aus Zeit- und Platzgründen behandeln wir in dieser Arbeit keinen Algorithmus, der QCOVER mit Hilfe eines *Divide and Conquer* Ansatzes löst. Allerdings ist auch hier beim Betrachten der Grundidee Vorsicht angebracht: Im schlimmsten Fall müssen (auf der untersten Rekursionstufe) *alle* Diskretisierungsquader betrachtet werden. Das kostet jedesmal $O(kn)$ Aufwand, was insgesamt zu einer Abschätzung von $O((2k+1)^n kn)$ führt, und das allein für die unterste Rekursionsstufe. Man müßte also zumindest eine Methode finden, wie man die Diskretisierungsquader an der jeweiligen Stelle in der Rekursion schneller auf Überdeckung prüfen kann.

[1] Ein *range tree* ist eine Art n-fach geschachtelte Struktur aus balancierten Suchbäumen.

Kapitel 4

Continuous Subtraction

In diesem Kapitel verwenden wir den *Incremental Construction* Ansatz, um QCOVER zu lösen. Es entsteht ein naheliegendes Verfahren, dessen Grundidee, wie in Abschnitt 3.2 angedeutet, darin besteht, die Evidenzquader der Reihe nach mit Q zu schneiden und den Schnitt von Q zu *subtrahieren*. Ist nach Abarbeitung aller Evidenzquader noch etwas von Q *übrig*, so wird Q nicht überdeckt. Abbildung 4.1 illustriert das Verfahren an einem einfachen zweidimensionalen Beispiel.

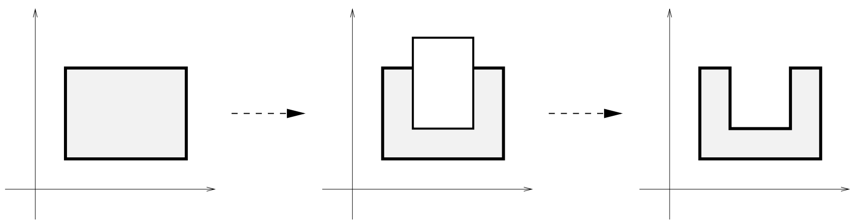

Abbildung 4.1: Ein einfaches Beispiel für zweidimensionales Subtracting mit nur einem Evidenzrechteck. Von links nach rechts haben wir Q, Q und H, $Q \setminus H$.

4.1 Ein abstrakter Algorithmus

Anhand von Abbildung 4.1 wird bereits das Hauptproblem des Ansatzes deutlich: Der Rest $Q \setminus H$ von Q nach Abzug eines Evidenzquaders H ist im allgemeinen kein Quader mehr. Wir benötigen also eine Methode, um diesen Rest darzustellen.

Wir wählen den naheliegendsten Ansatz: Man sieht leicht, daß $Q \setminus H$ in Abbildung 4.1 als *Vereinigung* dreier Rechtecke dargestellt werden kann, siehe Abbildung 4.2.

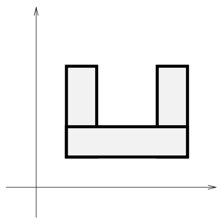

Abbildung 4.2: Darstellung des Restes aus Abbildung 4.1 mit Hilfe von drei Rechtecken.

Eine mögliche Verfahrensweise ist also, den Abzug des Evidenzrechtecks zu realisieren, indem man das ursprüngliche Anfragerechteck Q durch diese drei Rechtecke ersetzt. Mit dieser Methode erhält man nun anstelle *eines* Anfragequaders im allgemeinen eine *Menge* von Anfragequadern. Im nächsten Schritt müssen nun *alle* Quader in der Anfragemenge mit dem nächsten Evidenzquader geschnitten werden. Damit können wir bereits einen abstrakten Algorithmus formulieren, siehe die Definition des Continuous Subtraction Algorithmus in Abbildung 4.3.

```
Qset := {Q}
for all H_i ∈ H do /* H = {H_1, ..., H_k} */
    for all Q_j ∈ Qset do /* Qset = {Q_1, ..., Q_m} */
        Rset_j := a minimal exact covering of Q_j \ H_i with n-dimensional hypercubes (*)
    endfor
    Qset := ⋃_{j=1}^{m} Rset_j
    if Qset = ∅ then answer ⊤ endif
endfor
answer ⊥
```

Abbildung 4.3: Der Continuous Subtraction Algorithmus. Eingabe ist eine QCOVER Instanz, d.h., der Anfragequader Q sowie die Menge $\mathcal{H}$ der Evidenzquader.

Man beachte, daß mit Abbildung 4.3 lediglich ein *abstrakter* Algorithmus definiert ist, da nicht klar ist, wie die mit (*) markierte Zeile genau ausgeführt werden soll.

Die jeweils aktuelle Menge von Anfragequadern ist in Abbildung 4.3 mit $Qset$ bezeichnet. Sie wird, wie erwartet, als die Menge initialisiert, die als einziges Element Q, den ursprünglichen Anfragequader, enthält. Dann werden die Evidenzquader H_i der Reihe nach abgearbeitet. Von jedem Quader Q_j in $Qset$ wird nun H_i abgezogen. Dazu muß jeweils zu Q_j eine Menge $Rset_j$ von n-dimensionalen Quadern berechnet werden, die eine *minimale exakte Überdeckung* des Restes $Q_j \setminus H_i$ ist, d.h.:

- $Rset_j$ ist eine *exakte Überdeckung*: Es gilt $\bigcup Rset_j = Q_j \setminus H_i$.
- $Rset_j$ ist *minimal*: Keine andere exakte Überdeckung kommt mit weniger Quadern aus.

Nachdem diese Mengen für alle Quader Q_j berechnet sind, erhält man die neue Anfragemenge $Qset$, indem man alle Überdeckungen miteinander vereinigt. Dieser Prozeß hat offenbar genau den erwünschten Effekt, daß eben jeder Quader Q_j in der ursprünglichen Anfragemenge durch seine Überdeckungsquader $Rset_j$ *ersetzt* wird. Ist Q_j ganz in H_i enthalten, so ist $Q_j \setminus H_i = \emptyset$ und die einzige minimale exakte Überdeckung offenbar $Rset_j = \emptyset$.

Die neu erzeugten Quader in $Rset_j$ haben, aufgrund ihrer Konstruktion, einen leeren Schnitt mit allen bisher abgearbeiteten Evidenzquadern $H_1, \ldots, H_i$. Sie müssen deshalb nur noch mit den verbleibenden Evidenzquadern verglichen werden, d.h., der Algorithmus kann mit der neuen Anfragemenge (die ja ausschließlich aus neu erzeugten Quadern besteht) und dem nächsten Evidenzquader H_{i+1} fortgesetzt werden. Wir präzisieren diese Aussage in Abschnitt 4.3 zu einem formalen Beweis dafür, daß der Continuous Subtraction Algorithmus korrekt arbeitet, solange die *Überdeckungen der Reste gewissen Korrektheitsansprüchen genügen.*

Damit kommen wir zum entscheidenden Schritt des Prozesses für jedes Paar H_i und Q_j, und zur entscheidenden Frage für die Ausführung des Continuous Subtraction Algorithmus, wie er in Abbildung 4.3 definiert ist, überhaupt:

Wie berechnet man eine minimale exakte Überdeckung von $Q_j \setminus H_i$ *?*

Die ausführliche Beantwortung dieser Frage ist die Aufgabe des nächsten Abschnitts.

4.2 Ein *anderes* Überdeckungsproblem

Mit der exakten Überdeckung des Restes $Q_i \setminus H_j$ stehen wir vor einem grundlegend *anderem* Überdeckungsproblem als der eigentlichen QCOVER Problematik, mit der wir uns bisher beschäftigt haben. Während bei der ursprünglichen QCOVER Fragestellung *alle Objekte von vornherein gegeben* sind, und "nur" entschieden werden muß, ob eine (möglicherweise nicht exakte, d.h., zu großzügige) Überdeckung vorliegt, haben wir nun lediglich *ein* Objekt vorgegeben und müssen eine *Menge von anderen Objekten* explizit konstruieren. In der Literatur, etwa [Lee and Preparata, 1984], nennt man die ursprüngliche QCOVER Problematik ein *Entscheidungsproblem* und die Aufgabenstellung, vor der wir hier stehen, ein *Berechnungsproblem*.

Der Einfachheit halber sprechen wir im folgenden nur noch von der *Überdeckung* des Restes und meinen damit die *exakte Überdeckung*. Ebenfalls zur Simplifizierung lassen wir die Indizes i und j weg, d.h., wir untersuchen die Überdeckung von $Q \setminus H$.

4.2.1 Polygone

Das Objekt $Q \setminus H$, welches zu überdecken ist, ist von komplizierterer Natur als die Quader, mit denen wir es bisher zu tun hatten. Im zweidimensionalen Fall kann $Q \setminus H$ als ein *isoorientiertes, einfaches Polygon* aufgefaßt werden. Hierbei bedeutet *isoorientiert*, daß alle Kanten des Polygons parallel zu einer der Achsen sind, und *einfach*, daß bestimmte Sonderfälle ausgeschlossen sind, etwa der Schnitt zweier Kanten in einem Nicht-Endpunkt. Zugelassen sind hingegen *Löcher*, d.h. weitere Polygone im Inneren des Polygons, die bestimmte Regionen herausschneiden. In Abbildung 4.4 ist ein beliebiges Beispiel eines isoorientierten, einfachen Polygons zu sehen, die darunterstehende Abbildung 4.5 zeigt zwei mögliche Formen von $Q \setminus H$.

In [Masek, 1979] wird gezeigt, daß das minimale Überdecken beliebiger isoorientierter, einfacher Polygone durch Rechtecke NP-hart ist. Im zweidimensionalen Fall ist das genau die Aufgabenstellung, die wir zu bewältigen haben: das Polygon $Q \setminus H$ muß mit Quadern, also Rechtecken, minimal überdeckt werden. Allerdings sollte anhand der Abbildungen 4.4 und 4.5 deutlich geworden sein, daß es sich bei $Q \setminus H$ um ein sehr einfach strukturiertes Polygon handelt.[1] Wir werden sehen, daß man in diesem Spezialfall eine Überdeckung effizient konstruieren kann.

Zur Überdeckung von isoorientierten Polygonen durch Rechtecke gibt es eine reichhaltige Auswahl an Literatur, wohl vor allem, weil dieses Problem beim *Entwurf von Schaltkreisen* in massiver Form auftritt (Ein Schaltkreispolygon muß durch das Einbrennen von Rechtecken in eine Grundplatte realisiert werden). Allerdings sind ja, anders-

[1] Tatsächlich kann $Q \setminus H$ auch aus zwei Polygonen bestehen, diese sind dann bereits selbst Rechtecke. Dies spielt hier aber keine Rolle.

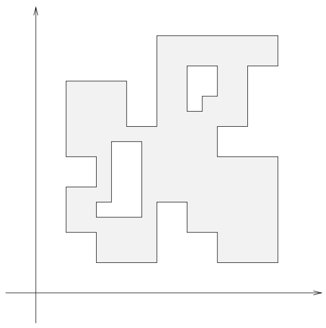

Abbildung 4.4: Ein (beliebiges) Beispiel eines isoorientierten, einfachen Polygons.

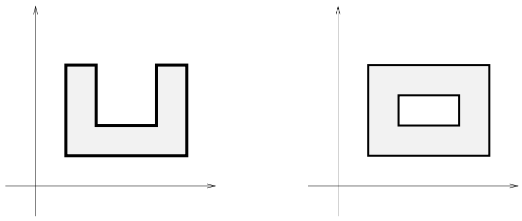

Abbildung 4.5: Zwei mögliche Formen von $Q \setminus H$.

herum betrachtet, auch die Polygone nur ein Spezialfall unseres Problems: wir müssen das n-dimensionale Objekt $Q \setminus H$ mit Hyperquadern überdecken, während Polygone und Rechtecke lediglich zweidimensional sind. Wir werden uns daher im folgenden nur grob an den Begriffen aus der Literatur orientieren und entstehende Querbezüge aufzeigen. Ansonsten geben wir eine eigenständige Darstellung der Problematik.

4.2.2 Atomare Überdeckungen

Ein erster Versuch zur Konstruktion einer Überdeckung beruht darauf, daß wir für jede Dimension d eine Menge $\mathcal{A}^d$ atomarer Intervalle identifizieren. Diese Intervalle beschreiben intuitiv die gegenseitige Lage von Q und H in jeder Dimension.

Es gelte $Q = Q^1 \times \ldots \times Q^n$ und $H = I^1 \times \ldots \times I^n$. Wir erhalten in jeder Dimension d drei atomare Intervalle wie folgt.

1. $C^d := Q^d \cap I^d$
2. $A^d_< := \{x \in Q^d \mid x < L(I^d)\}$
3. $A^d_> := \{x \in Q^d \mid x > H(I^d)\}$

Wir setzen $\mathcal{A}^d := \{C^d, A^d_<, A^d_>\}$. Das Intervall C^d enthält alle Punkte aus Q^d, die *innerhalb* von I^d liegen, das Intervall $A^d_<$ enthält alle Punkte *links* von I^d, das Intervall $A^d_>$ schließlich enthält alle Punkte, die *rechts* von I^d liegen. Wir halten fest:

- In jeder Dimension d gilt: $Q^d = \bigcup \mathcal{A}^d$.
- In jeder Menge $\mathcal{A}^d$ hat nur das Intervall C^d möglicherweise nicht-leeren Schnitt mit I_d, genauer gilt: $C^d \subseteq I^d$. Wir bezeichnen daher C^d als *das überdeckte Intervall.* Sowohl der Schnitt von $A^d_<$ mit I_d als auch der Schnitt $A^d_>$ mit I_d ist leer, es gilt $A^d_< \cup A^d_> = Q^d \setminus I^d$.

- Das Kreuzprodukt $C := C^1 \times \ldots \times C^n$ der überdeckten Intervalle ist genau der Teil von Q, der *herausgeschnitten* wird, d.h. es gilt $C = Q \cap H$.

Man beachte, daß diese atomaren Intervalle keine Intervalle im strengen Sinn von Definition 1 sind, insbesondere kann jedes der drei Intervalle unter Umständen leer sein. Die Intervalle $A^d_<$ und $A^d_>$ sind in jedem Fall halboffen; das ist nötig, um zu gewährleisten, daß diese Intervalle einen leeren Schnitt mit I^d haben.

In Abbildung 4.6 sind zwei mögliche gegenseitige Positionen von I^d und Q^d, sowie die daraus resultierenden atomaren Intervalle $\mathcal{A}^d$ verdeutlicht.

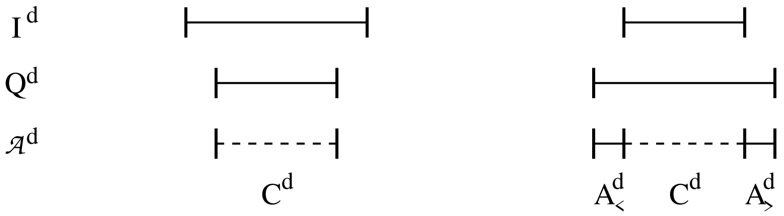

Abbildung 4.6: Mögliche gegenseitige Lagen von I^d (oben), Q^d (mitte) und den zugehörigen atomaren Intervallen. Im linken Bild ist Q^d ganz in I^d enthalten, nur ein atomares Intervall ist nicht-leer; im rechten Bild ragt Q^d an beiden Seiten über I^d hinaus, alle atomaren Intervalle enthalten Punkte. Das *überdeckte* atomare Intervall ist jeweils gestrichelt gezeichnet.

Wir definieren nun die Menge $\mathcal{A}$ der atomaren Hyperquader, die die kleinsten Bausteine in unserer Situation darstellen, wie folgt:

$$\mathcal{A} := \{a \mid a = A^1 \times \ldots \times A^n,\ \forall d : A^d \in \mathcal{A}^d\} \tag{4.1}$$

Wiederum sind diese Quader keine Hyperquader im strengen Sinn von Definition 2 mehr, da wir halboffene und unter Umständen leere Intervalle verwenden. Es kann jetzt bereits direkt eine Überdeckung $Rset_A$ von $Q \setminus H$ angegeben werden.

$$Rset_A := \mathcal{A} \setminus \{C\} \tag{4.2}$$

Wir bezeichnen die Überdeckung $Rset_A$ auch als die *atomare* Überdeckung. Abbildung 4.7 illustriert die atomare Überdeckung von $Q \setminus H$ für die Quader aus Abbildung 4.1.

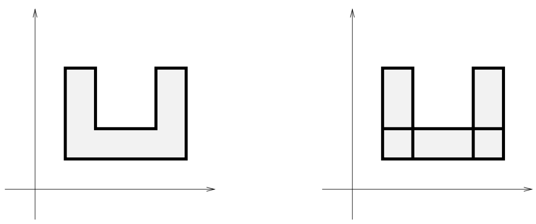

Abbildung 4.7: Der Rest von Q aus Abbildung 4.1 (linkes Bild), sowie seine atomare Überdeckung (rechtes Bild).

Bemerkt sei hier, daß im Zweidimensionalen diese Überdeckung des Restes nach Abzug eines Evidenzquaders, wenn man den Rest als orthogonales Polygon auffaßt, genau der Aufteilung des Polygons in *Zellen* entspricht, wie sie in [Culberson and Reckhow, 1994] beschrieben wird. Ein Polygon wird dort in Zellen aufgeteilt, indem die Kanten des Polygons solange nach innen verlängert werden, bis sie auf eine andere Kante stoßen.

4.2.3 Konstruktion einer minimalen Überdeckung

Wie man anhand Abbildung 4.7 sofort sieht, ist die atomare Überdeckung nicht minimal. In diesem Sinn haben wir also noch keine Lösung für unser Problem gefunden. Man gelangt zu einer minimalen Überdeckung, indem man wie folgt vorgeht:

- Gehe alle Dimensionen d in einer beliebigen Reihenfolge durch.
- Nimm, falls nötig, *möglichst große* Quader zur Überdeckung hinzu: wenn Q^d links über I^d hinausragt, $A^d_<$ also nicht leer ist, so gehören alle Punkte in Q, deren Koordinate in d innerhalb von $A^d_<$ liegt, zu $Q \setminus H$. Diese Punkte bilden gerade einen Hyperquader, nämlich den Quader $Q^1 \times \ldots \times Q^{d-1} \times A^d_< \times Q_{d+1} \times \ldots \times Q^n$. Genauso kann der Hyperquader aus allen Punkten in Q, deren Koordinate in d rechts von I^d (also in $A^d_>$) liegt, zur Überdeckung hinzugenommen werden.

Abbildung 4.8 demonstriert den Verlauf des Verfahrens auf dem Rest von Q aus Abbildung 4.7.

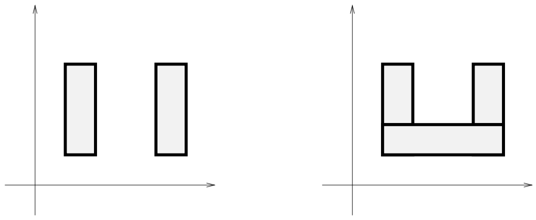

Abbildung 4.8: Konstruktion einer minimalen Überdeckung des Restes aus Abbildung 4.7. Zunächst werden zwei Quader für die x-Achse (linkes Bild), danach ein Quader für die y-Achse (rechtes Bild) gebildet.

Ist der Schnitt von Q mit H leer, so würden auf diese Art und Weise unnötig viele Quader zur Überdeckung hinzugenommen werden; in diesem Fall wäre in jeder Dimension d, in der $Q^d \cap I^d = \emptyset$ gilt, der hinzugenommene Quader bereits der ganze Anfragequader Q (Das Intervall von Q liegt ja dann ganz links (rechts) von I^d, fällt also vollständig mit $A^d_<$ ($A^d_>$) zusammen). Ist aber der Schnitt der beiden Quader Q und H leer, so kann als minimale Überdeckung von $Q \setminus H = Q$ einfach Q selbst zurückgegeben werden. Wir gelangen so zu dem in Abbildung 4.9 dargestellten Minimal Cover Algorithmus.

Man beachte, daß die Intervalle $A^d_<$ bzw. $A^d_>$ halboffen sind; dadurch werden halboffene Quader konstruiert. Andernfalls wäre der Schnitt der konstruierten Quader mit H nicht leer. Dies ist nicht zulässig, da ja $Q \setminus H$ überdeckt werden muß.

Einen formalen Beweis dafür, daß der Minimal Cover Algorithmus tatsächlich eine Überdeckung berechnet, daß also für die zurückgegebene Menge *Rset* die Gleichung $\bigcup Rset = Q \setminus H$ gilt, geben wir in Abschnitt 4.3. Daß die berechnete Überdeckung minimal ist, sehen wir im folgenden Abschnitt.

4.2.4 Minimalität

Der überdeckte Quader C hat eine bestimmte Anzahl von *Nachbarn*; als Nachbarn sind hierbei atomare Quader bezeichnet, deren Intervall sich nur in einer Dimension vom In-

```
if Q ∩ H = ∅ then return {Q} endif
Rset := ∅
for d = 1...n do
    if A^d_< ≠ ∅ then
      Rset := Rset ∪ {Q^1 × ... × Q^{d-1} × A^d_< × Q^{d+1} × ... × Q^n}
    endif
    if A^d_> ≠ ∅ then
      Rset := Rset ∪ {Q^1 × ... × Q^{d-1} × A^d_> × Q^{d+1} × ... × Q^n}
    endif
endfor
return Rset
```

Abbildung 4.9: Der Minimal Cover Algorithmus. Eingabe sind ein Anfragequader $Q = Q^1 \times \ldots \times Q^n$ und ein Evidenzquader $H = I^1 \times \ldots \times I^n$.

tervall des überdeckten Quaders unterscheidet, also genau die Quader N, die wie folgt aufgebaut sind:

$$N = C^1 \times \ldots \times C^{d-1} \times A^d \times C^{d+1} \times \ldots \times C^n,\ A^d \in \{A^d_<, A^d_>\} \tag{4.3}$$

Im allgemeinen sind viele dieser Nachbarquader leer: Jedesmal, wenn ein Intervall $A^d_<$ oder $A^d_>$ leer ist, enthält natürlich auch der resultierende Nachbarquader keinen einzigen Punkt. Die Nachbarquader in der Situation aus Abbildung 4.7 sind in Abbildung 4.10 zu sehen.

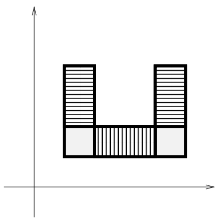

Abbildung 4.10: Die Nachbarrechtecke von C in der Situation aus Abbildung 4.7. Die zwei Nachbarn in x-Richtung sind waagerecht, der eine nicht-leere Nachbar in y-Richtung ist senkrecht gestreift.

Man überzeugt sich anhand von Abbildung 4.10 leicht, daß je zwei der Nachbarrechtecke in der Situation aus Abbildung 4.7 nicht durch *ein* Rechteck überdeckt werden können. Ein solches Rechteck würde immer auch einen Teil des 'Lochs', sprich, des überdeckten Rechtecks C mit abdecken. Tatsächlich haben die Punkte in den (nicht-leeren) Nachbarquadern von C im allgemeinen die Eigenschaft, daß sie nicht mit einem einzigen Quader überdeckt werden können. In der Literatur, etwa [Franzblau and Kleitman, 1984], bezeichnet man solche Punkte als *unabhängig.*

Lemma 4 *Ist der Schnitt von Q und H nicht leer, so gilt:*

Jede Überdeckung des Restes $Q \setminus H$ muß mindestens soviele Quader enthalten, wie der überdeckte Quader $C = Q \cap H$ nicht-leere Nachbarn hat.

Beweis:
Da nach Voraussetzung $C = Q \cap H \neq \emptyset$ gilt, haben wir auch in jeder Dimension $C^d = Q^d \cap I^d \neq \emptyset$.

Wir zeigen folgendes: Jeder Quader V, der zwei Punkte $\overline{x_1} \in N_1$ und $\overline{x_2} \in N_2$ aus verschiedenen Nachbarquadern N_1 und N_2 von C enthält, hat einen nicht-leeren Schnitt mit C. Da wir $Q \setminus H = Q \setminus C$ überdecken müssen, sind solche Quader unzulässig. Also müssen alle Punkte in verschiedenen (nicht-leeren) Nachbarquadern auch mit verschiedenen Überdeckungsquadern berücksichtigt werden. Es folgt die Behauptung.

Es seien also N_1 und N_2 nicht-leere Nachbarquader von C. Wir betrachten zunächst den Fall, daß beide Quader Nachbarn in derselben Dimension d sind; dann ist etwa $N_1 = C^1 \times \ldots \times C^{d-1} \times A^d_< \times C^{d+1} \times \ldots \times C^n$ und $N_2 = C^1 \times \ldots \times C^{d-1} \times A^d_> \times C^{d+1} \times \ldots \times C^n$. Die Koordinaten von $\overline{x_1}$ und $\overline{x_2}$ in allen von d verschiedenen Dimensionen d' liegen also in den Intervallen des überdeckten Quaders; in d selbst liegen sie zu beiden Seiten von C^d. Die Intervalle eines Überdeckungsquaders V für $\overline{x_1}$ und $\overline{x_2}$ müssen also in d das ganze, nach Voraussetzung nicht-leere Intervall C^d umfassen und in den anderen Dimensionen einen nicht-leeren Schnitt mit $C^{d'}$ haben; insgesamt ist somit $V \cap C \neq \emptyset$.

Im anderen Fall sind N_1 und N_2 aus verschiedenen Dimensionen, etwa d_1 und d_2. Wir haben dann $N_1 = C^1 \times \ldots \times C^{d_1-1} \times A^{d_1} \times C^{d_1+1} \times \ldots \times C^n$ und $N_2 = C^1 \times \ldots \times C^{d_2-1} \times A^{d_2} \times C^{d_2+1} \times \ldots \times C^n$. Dabei sind $A^{d_1} \neq C^{d_1}$ und $A^{d_2} \neq C^{d_2}$ atomare Intervalle. Wiederum muß ein Quader V, der beide Punkte überdeckt, in allen Dimensionen $d' \neq d_1, d_2$ einen nicht-leeren Schnitt mit $C^{d'}$ haben; in den anderen Dimensionen, etwa d_1, ebenfalls: die Koordinate von $\overline{x_2}$ in d_1 liegt ja per definitionem in C^{d_1}. Das Intervall von V in d_1 muß diese Koordinate enthalten, also hat es insbesondere einen nicht-leeren Schnitt mit C^{d_1}. Genauso argumentiert man für d_2, es folgt die Behauptung. ■

Man kommt nun durch eine einfache Überlegung zum Beweis der Minimalität der von Minimal Cover berechneten Überdeckung. Wir unterscheiden zwei Fälle.

- $Q \cap H = \emptyset$: In diesem Fall gibt der Algorithmus die Menge $\{Q\}$ zurück. Das ist offensichtlich eine minimale Überdeckung.
- $Q \cap H \neq \emptyset$: In diesem Fall ist Lemma 4 anwendbar. Jede Überdeckung von $Q \setminus H$ enthält also mindestens soviele Quader, wie C Nachbarn hat. Genau aus so vielen Quadern besteht aber auch die von Minimal Cover berechnete Überdeckung: Durch jedes nicht-leere atomare Intervall $A^d_<$ bzw. $A^d_>$ in jeder Dimension ist ja ein nicht-leerer Nachbar von C definiert; für jedes solche Intervall nimmt der Algorithmus aus Abbildung 4.9 aber offensichtlich auch genau einen Quader zur Überdeckung hinzu.

Wir halten noch einmal fest: Keine Überdeckung von $Q \setminus H$ kann weniger Quader enthalten als die von Minimal Cover berechnete Menge. Da der Algorithmus in jeder Dimension höchstens zwei Quader zu dieser Menge hinzunimmt, wissen wir außerdem:

Die von Minimal Cover berechnete Menge enthält höchstens $2n$ Quader. (4.4)

Sobald bewiesen ist, daß Minimal Cover auch tatsächlich eine *exakte Überdeckung* von $Q \setminus H$ liefert, ist dann also insbesondere gezeigt, daß im allgemeinen bereits $2n$ Quader ausreichen, um $Q \setminus H$ zu überdecken.

4.3 Korrektheit

Wir beweisen zunächst die Korrektheit des Minimal Cover Algorithmus. Die Aussagen beziehen sich auf die in Abbildung 4.9 gegebene Definition des Verfahrens. Es gilt der folgende Satz.

Satz 2 *Bei Eingabe zweier Quader $Q = Q^1 \times \ldots \times Q^n$ und $H = I^1 \times \ldots \times I^n$ gilt nach Ausführung des* Minimal Cover *Algorithmus für die zurückgegebene Menge Rset:*

$$\bigcup Rset = Q \setminus H$$

Beweis:

Für den Fall $Q \cap H = \emptyset$ ist die Aussage trivial. Wir betrachten deshalb nur den Fall, daß die Überdeckung $Rset$ explizit konstruiert wird.

$\bigcup Rset \subseteq Q \setminus H$: Für jeden Quader $R \in Rset$ gilt $R \subseteq Q \setminus H$. Das sieht man so: Werde R im d-ten Durchlauf der Schleife zu $Rset$ hinzugenommen. Es gilt dann $R = R^1 \times \ldots \times R^n := Q^1 \times \ldots \times Q^{d-1} \times A^d \times Q^{d+1} \times \ldots \times Q^n$ mit $A^d \in \{A^d_<, A^d_>\}$. Also gilt in jeder Dimension d': $R^{d'} \subseteq Q^{d'}$, somit insgesamt $R \subseteq Q$. Zusätzlich gilt in Dimension d: $R^d \cap I^d = A^d \cap I^d = \emptyset$, also $R \cap H = \emptyset$. Es folgt wie gewünscht $R \subseteq Q \setminus H$ für $R \in Rset$ und damit $\bigcup Rset \subseteq Q \setminus H$.

$\bigcup Rset \supseteq Q \setminus H$: Es sei $\overline{x} = (x^1, \ldots, x^n)$ ein Punkt in $Q \setminus H$. Wegen $\overline{x} \in Q$ liegen alle Koordinaten x^d des Punktes in den Intervallen Q^d des Anfragequaders. Wegen $\overline{x} \notin H$ gibt es mindestens eine Dimension d, in der $x^d \notin I^d$ erfüllt ist. In dieser Dimension gilt dann offenbar $x^d \in A^d_< \vee x^d \in A^d_>$, da ja x^d entweder links oder rechts von I^d liegen muß. Der Punkt $\overline{x}$ ist also in einem der Quader enthalten, die der Minimal Cover Algorithmus beim Bearbeiten von Dimension d zu $Rset$ hinzunimmt. Es folgt die Behauptung. ■

Man erhält nun ein korrekt arbeitendes Gesamtverfahren, indem man an der in Abbildung 4.3 mit (*) markierten Stelle, also der Berechnung einer minimalen exakten Überdeckung, den Minimal Cover Algorithmus als Unterprogramm aufruft. Die entstehende Endfassung des Continuous Subtraction Verfahrens zeigt Abbildung 4.11. Zur besseren Übersicht im nachfolgenden Beweis sind die Zeilen des Algorithmus durchnummeriert.

```
1 Qset := {Q}
2 for H_i := H_1 ... H_k do /* 𝓗 = {H_1, ..., H_k} */
3     for Q_j := Q_1 ... Q_m do /* Qset = {Q_1, ..., Q_m} */
4         Rset_j := Minimal Cover(Q_j, H_i)
5     endfor
6     Qset := ⋃_{j=1}^{m} Rset_j
7     if Qset = ∅ then answer ⊤ endif
8 endfor
9 answer ⊥
```

Abbildung 4.11: Endfassung des Continuous Subtraction Algorithmus.

Satz 3 *Der* Continuous Subtraction *Algorithmus, wie er in Abbildung 4.11 definiert ist, antwortet $\top$ genau dann, wenn $Q \sqsubset \mathcal{H}$.*

Beweis:
Wir bezeichnen die Anfragemenge $Qset$ beim i-ten Erreichen der Zeile 7 mit $Qset_i$ und zeigen durch Induktion über i: $\bigcup Qset_i = Q \setminus (H_1 \cup \ldots \cup H_i)$.

Induktionsanfang $i = 1$: In diesem Fall wird Schleife 3 ... 5 nur einmal durchlaufen. In Zeile 6 wird $Qset_1$ zu $Qset_1 := \bigcup_{j=1}^{1} Rset_j = Rset_1$ berechnet. Mit Satz 2 gilt $\bigcup Rset_1 = Q \setminus H_1$. Also gilt auch $\bigcup Qset_1 = Q \setminus H_1$.

Induktionsschritt $i \longrightarrow i+1$: Gilt bereits beim i-ten Erreichen von Zeile 7 $Qset_i = \emptyset$, so wird kein $i+1$-ter Durchlauf mehr durchgeführt, dann ist also nichts zu zeigen.

Sei nun also nach i Durchläufen $Qset_i = \{Q_1, \ldots, Q_m\}$. Schleife 2 ... 8 wird dann mit dem neuen Evidenzquader H_{i+1} durchlaufen. Im Zuge der Konstruktion von $Qset_{i+1}$ wird jeder Quader Q_j durch das zugehörige $Rset_j$ ersetzt: Nach Ausführung von Zeile 6 gilt $Qset_{i+1} = \bigcup_{j=1}^{m} Rset_j$. Da mit Satz 2 $\bigcup Rset_j = Q_j \setminus H_{i+1}$ für alle j gilt, ist somit $\bigcup Qset_{i+1} = \bigcup_{j=1}^{m}(Q_j \setminus H_{i+1}) = (\bigcup_{j=1}^{m} Q_j) \setminus H_{i+1} = \bigcup Qset_i \setminus H_{i+1}$. Nach Induktionsvoraussetzung gilt $\bigcup Qset_i = Q \setminus (H_1 \cup \ldots \cup H_i)$, womit sofort die Behauptung folgt.

Der Algorithmus antwortet nun genau dann $\top$, wenn für ein $i \leq k$ die Anfragemenge leer wird, also $Qset_i = \emptyset$ gilt. Wie man sich leicht überzeugt, sind alle Quader, die von Minimal Cover erzeugt werden, nicht-leer. Da dies genau die Quader sind, die im Verlauf des Algorithmus zu $Qset$ hinzugenommen werden, ist $Qset_i = \emptyset$ äquivalent zu $\bigcup Qset_i = \emptyset$. Wir haben bewiesen, daß $\bigcup Qset_i = Q \setminus (H_1 \cup \ldots \cup H_i)$ gilt. Also

$$\begin{aligned} Qset_i = \emptyset \quad &\Leftrightarrow \quad \bigcup Qset_i = \emptyset \\ &\Leftrightarrow \quad Q \setminus (H_1 \cup \ldots \cup H_i) = \emptyset \\ &\Leftrightarrow \quad Q \subseteq (H_1 \cup \ldots \cup H_i) \\ &\Leftrightarrow \quad Q \sqsubset \{H_1, \ldots, H_i\} \end{aligned}$$

Offenbar gibt es ein $i \leq k$, für das diese Bedingung erfüllt ist, genau dann, wenn $Q \sqsubset \{H_1, \ldots, H_k\}$ gilt. Es folgt die Gesamtbehauptung. ∎

4.4 Laufzeit

Zum Abschätzen des Laufzeitverhaltens von Continuous Subtraction im worst-case betrachten wir Abbildung 4.11. Die äußere Schleife wird offensichtlich genau k-mal durchlaufen. Die Anzahl der jeweiligen Durchläufe von Schleife 3 ... 5 ist allerdings variabel; die Schleife wird genau so oft durchlaufen, wie die Anfragemenge Elemente hat. Wir bezeichnen den Betrag der Anfragemenge $Qset$ nach i Durchläufen der Schleife 2 ... 8 im folgenden mit m_i; es gilt etwa $m_0 = 1$.[2]

In Zeile 4 wird der Minimal Cover Algorithmus aufgerufen. Man überzeugt sich leicht, daß ein einzelner Aufruf von Minimal Cover in Zeit $O(n)$ durchgeführt werden kann: Um zu prüfen, ob $Q \cap H = \emptyset$ gilt, genügt es, in allen Dimensionen zu prüfen, ob der Schnitt von Q^d und I^d leer ist. Ist das in keiner Dimension der Fall, so haben Q und H einen nicht-leeren Schnitt und es muß eine Überdeckung von $Q \setminus H$ konstruiert werden, d.h., die Schleife über alle Dimensionen ist auszuführen. In jedem Durchlauf der Schleife werden unter bestimmten Bedingungen einzelne Elemente (Überdeckungsquader) zu einer Menge ($Rset$) hinzugenommen. Implementiert man dies etwa mit Hilfe einer linearen Liste, so

[2] Unter Bezugnahme auf die Argumentation im Beweis zu Satz 3 ist also $m_i = |Qset_i|$ für $i \geq 1$.

erhält man für jeden Durchlauf der Schleife konstante Laufzeit, insgesamt benötigt der Algorithmus also nicht mehr als lineare Laufzeit $O(n)$ in der Anzahl der Dimensionen. Da die Schleife $3 \ldots 5$ jeweils genau m_i mal durchlaufen wird, ergibt sich somit für ihre Ausführung eine asymptotische Laufzeit von $m_i O(n)$.

In Zeile 6 erhält man auf den ersten Blick ebenfalls eine Laufzeit in der Größenordnung $m_i O(n)$: Für jeden Quader $Q_1 \ldots Q_{m_i}$ werden von Minimal Cover ja gemäß Aussage 4.4 maximal $2n$ Überdeckungsquader gebildet. Diese Quader werden in Zeile 6 zur neuen Anfragemenge zusammengefaßt. Allerdings resultiert dieser scheinbare Aufwand zum Erstellen des neuen $Qset$ nur daraus, wie der Algorithmus (der Übersichtlichkeit wegen) aufgeschrieben ist. Tatsächlich kann man bereits innerhalb des Minimal Cover Algorithmus die benötigten Überdeckungsquader direkt zur Anfragemenge hinzunehmen, also ganz auf die Hilfsstrukturen $Rset_j$ verzichten und somit den zusätzlichen Aufwand in Zeile 6 gänzlich umgehen.

Zeile 7 ist offenbar in konstanter Zeit ausführbar. Wir erhalten damit für jeden Evidenzquader, also für jeden Durchlauf der äußeren Schleife im Continuous Subtraction Algorithmus, eine Laufzeit in Höhe von $m_i O(n)$. Für die maximale Gesamtlaufzeit $T_{CS}(n, k)$ des Verfahrens ergibt sich dadurch folgende Abschätzung:

$$T_{CS}(n, k) = m_0 O(n) + m_1 O(n) + \ldots + m_{k-1} O(n) \tag{4.5}$$

Man kann in Gleichung 4.5 offenbar O(n) "ausklammern" (also die größte gefundene Konstante wählen) und erhält dann:

$$T_{CS}(n, k) = O(n(m_0 + m_1 + \ldots + m_{k-1})) = O(n \sum_{i=0}^{k-1} m_i) \tag{4.6}$$

Die entscheidende Abschätzung, die also vorgenommen werden muß, ist der Betrag des Anfragesets nach i Durchläufen der äußeren Schleife. Wir zeigen im folgenden zwei mögliche Methoden dies zu tun.

4.4.1 Naive Multiplikation

Eine naheliegende Möglichkeit, m_i nach oben abzuschätzen, ergibt sich durch Aussage 4.4: Durch den Minimal Cover Algorithmus werden für einen Quader jeweils höchstens $2n$ neue Quader zu $Qset$ hinzugenommen. Es gilt somit $m_0 = 1$ und $m_1 \leq 2n$; eine Abschätzung für m_2 erhält man durch Multiplikation der Abschätzung für m_1 mit $2n$, also $m_2 \leq 4n^2$. Allgemein gilt $m_{i+1} \leq m_i * 2n$ und damit $m_i \leq (2n)^i$ für $0 \leq i \leq k-1$. Wir haben also:

$$\sum_{i=0}^{k-1} m_i \leq \sum_{i=0}^{k-1} (2n)^i = \frac{(2n)^k - 1}{2n - 1} \tag{4.7}$$

Insgesamt erhalten wir also für den Continuous Subtraction Algorithmus mit Gleichung 4.6:

$$T_{CS}(n, k) = O(n \frac{(2n)^k - 1}{2n - 1}) = O((2n)^k) \tag{4.8}$$

Wie man leicht einsieht, ist diese Abschätzung, die wir im folgenden *Naive Multiplikation* nennen, zu pessimistisch; es werden nur im worst-case tatsächlich $2n$ Quader zu $Qset$

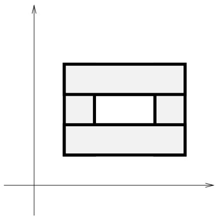

Abbildung 4.12: Der worst-case für eine zweidimensionale minimale Überdeckung.

hinzugenommen. Der worst-case tritt ein, wenn der Evidenzquader echt im Anfragequader enthalten ist, genauer wenn in jeder Dimension sowohl $A^d_<$ als auch $A^d_>$ nicht leer sind. Ein Beispiel für diesen worst-case im Zweidimensionalen ist in Abbildung 4.12 dargestellt.

Nun ist aber klar, daß ein weiterer Quader, bzw. im Beispiel aus Abbildung 4.12 ein weiteres Rechteck, nicht bezüglich *allen* so entstandenen Quadern in worst-case Position liegen kann. Läge der weitere Quader H bezüglich eines der entstandenen Quader in dieser Position, so wäre H vollständig in diesem Quader enthalten und damit sein Schnitt mit den meisten anderen Quadern leer, siehe die Situation in Abbildung 4.13.

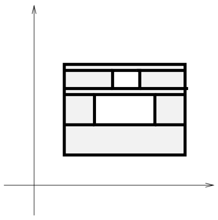

Abbildung 4.13: Das nächste Evidenzrechteck liegt in worst-case Position bezüglich des oberen Überdeckungsrechtecks.

Wie man sich leicht überlegt, reichen in Abbildung 4.13 bereits 7 Rechtecke aus, um das neue Ergebnis zu überdecken.

Es gibt eine Vielzahl von anderen Positionen, in denen H relativ zu den Überdeckungsquadern liegen kann; eine andere Möglichkeit, das Beispiel aus Abbildung 4.12 fortzuführen, sieht man in Abbildung 4.14.

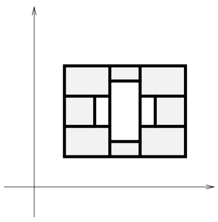

Abbildung 4.14: Eine ungünstige Lage für das nächste Evidenzrechteck im Beispiel aus Abbildung 4.12.

In dieser Situation benötigt man bereits 8, statt, wie im vorherigen Fall, 7 Überdeckungsquader für das neue Ergebnis. Im allgemeinen erweist es sich als äußerst schwierig, aufgrund der gegenseitigen Lage von neuem Evidenzquader und vorhandenen Überdeckungsquadern irgendwelche genaueren Oberschranken für den Betrag von $Qset$ abzuleiten. Ob und wie dies möglich ist, ist ein offenes Problem.

4.4.2 Abschätzung der Diskretisierung

Indem wir die Perspektive ändern und wieder einmal die *Diskretisierung* der Instanz betrachten, gelangen wir zu einer anderen Abschätzung, die in den Fällen, in denen es mehr Evidenzquader als Dimensionen gibt (was der Regelfall sein sollte), besser ist als *Naive Multiplikation*: Jeden Überdeckungsquader, der im i-ten Durchlauf der äußeren Schleife zu $Qset$ hinzugenommen wird, kann man als Vereinigung von Diskretisierungquadern gemäß Definition 6 darstellen.

Nun wird nach i Durchläufen der äußeren Schleife die Situation nur durch den Anfragequader sowie die bisher abgearbeiteten i Evidenzquader diskretisiert, d.h., in jeder Dimension haben wir maximal $2i + 2$ verschiedene Intervallendpunkte, die die Dimensionsachse in maximal $2i + 1$ Abschnitte unterteilen. Eine Darstellung für das Beispiel aus Abbildung 4.14 ist in Abbildung 4.15 gegeben.

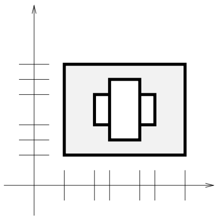

Abbildung 4.15: Unterteilung der Achsen durch die Endpunkte zweier Evidenzquader und des Anfragequaders.

Man überlegt sich leicht, daß die Quader, die von Minimal Cover geliefert werden, aus Kreuzprodukten dieser Abschnitte aufgebaut werden können, daß sie also als Vereinigung von aktuellen Diskretisierungsquadern darstellbar sind.[3] Weiter gibt es zu jedem von Minimal Cover gelieferten Quader Q mindestens einen Diskretisierungsquader, der *nur* in Q enthalten ist. Insgesamt kann es also nach i Durchläufen nicht mehr Überdeckungsquader als Diskretisierungsquader geben. Da die maximale Anzahl von Diskretisierungsquadern nach i Durchläufen $(2i + 1)^n$ ist, haben wir also:

$$m_i \leq (2i + 1)^n \tag{4.9}$$

Wenn wir über i aufsummieren, ergibt sich folgendes:

$$\sum_{i=0}^{k-1} m_i \leq \sum_{i=0}^{k-1} (2i + 1)^n \leq \sum_{i=0}^{k-1} (2k - 1)^n = k(2k - 1)^n \tag{4.10}$$

Die Abschätzung der Summe ist natürlich nicht besonders gut; aus den ein-, zwei- und dreidimensionalen Fällen ist aber bekannt, daß die Summe $\sum_{i=1}^{k} i^n$ in $\Omega(k^{n+1})$ ist, was darauf schließen läßt, daß wir $\sum_{i=0}^{k-1} m_i$ hier tatsächlich nicht auf $O((2k-1)^n)$ abschätzen können. Es ergibt sich insgesamt für den Continuous Subtraction Algorithmus mit Gleichung 4.6 folgende weitere Abschätzung:

$$T_{CS}(n, k) = O(nk(2k - 1)^n) \tag{4.11}$$

Im Gegensatz zu Gleichung 4.8, wo $T_{CS}(n, k)$ zu $T_{CS}(n, k) = O((2n)^k)$ abgeschätzt wird, ist diese Abschätzung exponentiell in n, statt in k.

[3] Streng genommen ist das nicht ganz richtig, wegen der *Randpunkte* der Quader: Die von Minimal Cover gelieferten Quader sind ja halboffen, die Diskretisierungsquader hingegen sind alle abgeschlossen. Für die hier gegebene Argumentation macht das keinen Unterschied.

Kapitel 5

Dimension Scanning

In diesem Kapitel stellen wir einen Algorithmus zur Lösung von QCOVER vor, dem ein in der algorithmischen Geometrie oft verwendetes Prinzip zugrundeliegt: Die *Scan-Line* Methode. Das entstehende Verfahren ist zwar intuitiv weniger klar als Continuous Subtraction, läßt aber eine bessere worst-case Laufzeitabschätzung zu.

5.1 Grundlagen

Wir orientieren uns bei der Beschreibung des Scan-line Prinzips an der Darstellung in [Ottmann and Widmayer, 1997]. Gegeben eine Menge von achsenparallelen Objekten in der Ebene (etwa im zweidimensionalen QCOVER Fall), schwenkt man eine vertikale Linie, die Scan-line, von links nach rechts über die gegebene Menge und zerlegt damit ein statisches zweidimensionales Problem in eine dynamische Folge eindimensionaler Probleme. Die Scan-line unterteilt dabei die Menge von Objekten in drei disjunkte Teilmengen: die *toten* Objekte, die bereits vollständig von der Scan-line überstrichen wurden, die *aktiven* Objekte, die gerade von ihr geschnitten werden, sowie die *inaktiven* Objekte, die erst künftig geschnitten werden. Während des Schwenks berechnet man nun dynamisch alle zur Lösung des jeweiligen Problems benötigten Informationen an der jeweiligen Schwenkposition. Im einfachsten Fall kann das etwa gerade die Menge der aktiven Objekte sein. Entscheidend ist dabei, daß der Schwenk nicht kontinuierlich sein muß; es genügt, die Punkte zu betrachten, an denen sich die Situation *ändern* kann. Hat man etwa eine Menge von Rechtecken gegeben, und ist $\mathcal{P} := \{P_1, \ldots, P_m\}$ die aufsteigend sortierte Menge aller x-Koordinaten der Eckpunkte, so reicht es aus, $P_1 \ldots P_m$ als *Haltepunkte* der Scan-line zu wählen. Man positioniert also die Scan-line anfangs gerade auf P_1 und springt dann jeweils vom Haltepunkt P_i zum nächsten Haltepunkt P_{i+1}.

Das Dimension Scanning Verfahren, das wir in diesem Kapitel entwickeln wollen, ist eine rekursive Verallgemeinerung dieser Methodik: mittels einer Scan-line wird ein statisches n-dimensionales Problem in eine dynamische Folge n-1-dimensionaler Probleme aufgespalten. Aus der vertikalen Linie im zweidimensionalen Fall wird im allgemeinen Fall eine Hyperebene, d.h., wenn P die Koordinate des aktuellen Haltepunktes ist, so enthält die "Scan-line" gerade alle Punkte des Raums, deren erste Koordinate P ist, also die Menge $\{(x^1, \ldots, x^n) \in \mathbb{Q}^n \mid x^1 = P\}$. Im Zweidimensionalen ist so eine (achsenparallele) Gerade, im dreidimensionalen Fall eine Ebene definiert. Man beachte, daß ein Quader $H = I^1 \times \ldots \times I^n$ von dieser Scan-line genau dann geschnitten wird, wenn sein Intervall in Dimension 1 den Punkt P enthält, also $P \in I^1$ gilt; dies ist wiederum mit der üblichen Notation äquivalent zu $L(I^1) \leq P \leq H(I^1)$.

Wie wir in Abschnitt 2.3 gesehen haben, sind die entscheidenden Punkte, mit denen wir uns beschäftigen müssen, die *Mittelpunkte* der Quader in der Anfragediskretisierung: Aus Lemma 2 wissen wir, daß Q genau dann überdeckt wird, wenn dies für die gesamte Anfragediskretisierung der Fall ist; aus Lemma 3 wiederum wissen wir, daß jeder dieser Diskretisierungsquader genau dann überdeckt wird, wenn dies für seinen Mittelpunkt der Fall ist. Wir werden also, grob zusammengefaßt, die Scan-line an den Mittelpunkten der Quader der Anfragediskretisierung anlegen; dann sind nur noch die jeweils aktiven Evidenzquader zu bestimmen: sind keine aktiven Evidenzquader vorhanden, so wird Q nicht überdeckt.

5.2 Der eindimensionale Fall

Wir betrachten zunächst den eindimensionalen Fall, um den prinzipiellen Verlauf des Dimension Scanning zu verdeutlichen. Der allgemeine Fall ergibt sich aus dem eindimensionalen Verfahren in sehr natürlicher Art und Weise, wie wir in Abschnitt 5.3 sehen werden.

5.2.1 Ein einfaches Verfahren

Gegeben sind im eindimensionalen Fall k Evidenzintervalle $H_i = [L(H_i), H(H_i)]$ sowie ein Anfrageintervall $Q = [L(Q), H(Q)]$. Die "Scan-line" ist im eindimensionalen einfach ein Punkt, nämlich der gegenwärtige Haltepunkt. Ein Intervall ist also genau dann aktiv, wenn es den gegenwärtigen Haltepunkt enthält.

Wie bereits erwähnt, orientieren wir uns mit den Haltepunkten an den Mittelpunkten der Quader in der Anfragediskretisierung; im Eindimensionalen sind diese Quader einfach Intervalle, deren Mittelpunkte wir folgerichtig als Haltepunkte für unsere Scan-line benutzen. Damit können wir das Verfahren im Eindimensionalen bereits grob beschreiben.

- Gehe über alle Mittelpunkte M von Intervallen der Anfragediskretisierung.
- Bestimme die Menge $Hset$ der in M aktiven Evidenzintervalle (Das sind genau diejenigen $H \in \mathcal{H}$, für die $M \in H$ gilt).
- Gibt es an irgendeinem Punkt M keine aktiven Evidenzintervalle, ist bei M also $Hset = \emptyset$, so antworte $\perp$. Sonst antworte $\top$.

Um dieses Verfahren formal korrekt aufschreiben zu können, benötigen wir noch den Begriff der *Scan-Liste* einer eindimensionalen Instanz. Das sind die zu betrachtenden Haltepunkte, sprich die Menge $\mathcal{M}$ der Mittelpunkte von Intervallen der Anfragediskretisierung.

5.2.2 Die Scan-Liste einer Instanz

Zunächst haben wir die Menge aller Endpunkte von Intervallen der Instanz, also $\{P_1, \ldots P_m\} := \{L(H_1), H(H_1), \ldots, L(H_k), H(H_k), L(Q), H(Q)\}$ gegeben. Wir bezeichnen diese Menge künftig mit $\mathcal{P}$, also $\mathcal{P} := \{P_1, \ldots P_m\}$. Wie bei der Definition der Diskretisierung aus Abschnitt 2.3 verwenden wir atomare Intervalle $[P_i, P_{i+1}]$, allerdings hier nur diejenigen solchen Intervalle, die von Q überdeckt werden. Das sind die Intervalle, die von den Punkten $L(Q) \le P_i \le H(Q)$ gebildet werden.

Für geeignete Indizes $1 \le l, h \le m$ fallen die Grenzen von Q auf die entsprechenden Punkte, d.h. $P_l = L(Q)$ und $P_h = H(Q)$. Mit diesen Bezeichnungen definieren wir die Menge $\mathcal{M}$ der zu betrachtenden Mittelpunkte wie folgt:

$$\mathcal{M} := \begin{cases} \{\frac{P_i+P_{i+1}}{2} \mid l \le i \le h-1\} & L(Q) \ne H(Q) \\ \{L(Q)\} & L(Q) = H(Q) \end{cases} \tag{5.1}$$

Alle Punkte in $\mathcal{M}$ sind offenbar Mittelpunkte von Intervallen in der Anfragediskretisierung; gilt $L(Q) = H(Q)$, so ist $L(Q)$ natürlich der einzige solche Punkt. Andererseits enthält $\mathcal{M}$ nicht *alle* Mittelpunkte von Intervallen der Anfragediskretisierung gemäß den Definitionen aus Abschnitt 2.3: In dem Fall, daß ein *Evidenzintervall* H nur ein Punkt $\{L(H)\}$ ist, wird dieses Intervall ja zur Diskretisierung hinzugenommen. Der Mittelpunkt $L(H)$ dieses Intervalls taucht nicht in $\mathcal{M}$ auf. Wir werden beim Beweis der Korrektheit von Dimension Scanning in Satz 4 feststellen, daß es für unsere Zwecke unnötig ist, solche Punkte zu betrachten.

Wir können nun eine erste formale Version des Dimension Scanning Verfahrens im Eindimensionalen angeben, siehe Abbildung 5.1.

```
for all stopping points M_i ∈ M do
    Hset := {H ∈ H | M_i ∈ H}
    if Hset = ∅ then answer ⊥ endif
endfor
answer ⊤
```

Abbildung 5.1: Erste Fassung des Dimension Scanning Algorithmus im Eindimensionalen. Eingabe ist die Scan-Liste einer (eindimensionalen) QCOVER Instanz.

Daß der Algorithmus aus Abbildung 5.1 korrekt arbeitet, sollte klar sein. Man erkennt auch sofort, daß das Dimension Scanning ein gewisses Preprocessing benötigt: zunächst einmal muß für jede Dimension die Scan-Liste erstellt werden; dazu müssen die Endpunkte aller Intervalle eingesammelt und sortiert werden. Die Laufzeit hierfür wird asymptotisch von der Laufzeit des eigentlichen Algorithmus dominiert, siehe dazu Abschnitt 5.5.

Obwohl der Algorithmus auf den ersten Blick sehr einfach, beinahe trivial erscheint, besitzt er eine gewisse Schwierigkeit: Es wird ja nichts darüber ausgesagt, wie die Menge $Hset$ jeweils zu berechnen ist. Wir wenden uns dieser Fragestellung im nächsten Abschnitt zu. Zunächst betrachten wir ein einfaches Beispiel.

Die gegenseitige Lage von Anfrage- und zwei Evidenzintervallen ist in Abbildung 5.2 zu sehen. Man beachte dabei, daß die Intervalle nur zur besseren Übersicht übereinander angeordnet sind; tatsächlich liegen sie alle auf *einer* Achse.

Zur besseren Übersicht geben wir, wie in Abbildung 5.2 ersichtlich, jedem Punkt in $\mathcal{M}$ die Nummer der unteren Grenze des Intervalls, dessen Mittelpunkt er ist. Führt man das Dimension Scanning Verfahren auf dem Beispiel aus Abbildung 5.2 aus, so wird $Hset$ am Haltepunkt M_3 leer und man erhält die korrekte Antwort $\perp$, daß also das Anfrageintervall *nicht* überdeckt wird.

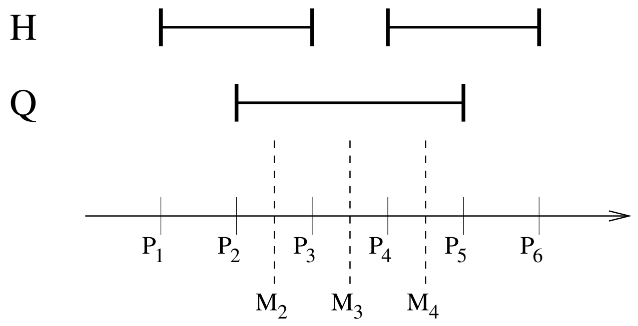

Abbildung 5.2: Ein Beispiel für die gegenseitige Lage von zwei Evidenzintervallen (oben) und dem Anfrageintervall (Mitte). Unten sind die Punkte in $\mathcal{P}$ sowie die drei Haltepunkte des Dimension Scanning dargestellt; die Numerierung der Haltepunkte orientiert sich an der unteren Grenze des jeweiligen Intervalls.

5.2.3 Aktive Evidenzintervalle

Die Frage ist nun, wie die Menge $\{H \in \mathcal{H} \mid M_i \in H\}$ der aktiven Evidenzintervalle effizient berechnet werden kann. Wir verwenden dazu eine Methode, die für Algorithmen, die nach dem Scan-line Prinzip arbeiten, typisch ist: Wir halten ständig eine Struktur mit Informationen über das aktuelle Problem aufrecht und passen diese im Verlauf des Scans dynamisch an die Situation an; dadurch muß an jedem Haltepunkt nur ein *Update* auf der entsprechenden Struktur ausgeführt werden.

In unserem Fall ist die aufrechtzuerhaltende Information schlicht die Menge $Hset$ der aktiven Evidenzintervalle; einfach gesagt, fügen wir an jedem Punkt diejenigen Intervalle zu $Hset$ hinzu, deren untere Intervallgrenze wir gerade überschritten haben und ziehen diejenigen Intervalle ab, deren obere Grenze überschritten ist. Genauer bezeichnen wir an jedem Punkt $P_i \in \mathcal{P}$ die Menge der Evidenzintervalle H, deren untere Grenze bei P_i liegt, mit IN_i, also $IN_i := \{H \in \mathcal{H} \mid L(H) = P_i\}$. Analog definieren wir die Menge OUT_i der Intervalle, die hier ihre obere Grenze haben als $OUT_i := \{H \in \mathcal{H} \mid H(H) = P_i\}$.

Die IN/OUT Information muß, genau wie die Scan-Liste, vom Preprocessing bereitgestellt werden; dies kann ohne asymptotisch höheren Zeitaufwand geschehen, mehr dazu in Abschnitt 5.5. Wir gehen also im folgenden davon aus, daß für jeden Punkt $P_i \in \mathcal{P}$ die Mengen IN_i und OUT_i zur Verfügung stehen. Um mit Hilfe dieser Information eine Darstellung für die jeweils bei M_i aktiven Evidenzintervalle zu erhalten, müssen wir zwei Fälle unterscheiden.

- $L(Q) = H(Q)$: In diesem Fall ist der einzige Punkt $M_i = L(Q)$; es sind alle Evidenzintervalle aktiv, die an den Punkten 1 bis i angefangen haben und bis $i-1$ noch nicht aufgehört haben, also $Hset = \{H \in \mathcal{H} \mid M_i \in H\} = \bigcup_{j=1}^{i} IN_j \setminus \bigcup_{j=1}^{i-1} OUT_j$. Die Menge OUT_i kann *nicht* abgezogen werden, da diese Intervalle zwar in $P_i = M_i$ aufhören, aber den Punkt M_i natürlich noch enthalten.

- $L(Q) \neq H(Q)$: Hier haben wir an jedem Punkt $M_i \in \mathcal{M}$ die Menge $Hset = \{H \in \mathcal{H} \mid M_i \in H\} = \bigcup_{j=1}^{i} IN_j \setminus \bigcup_{j=1}^{i} OUT_j$. Der Unterschied zum ersten Fall liegt darin, daß wir auch die Intervalle abziehen müssen, die bei i aufgehört haben, denn der Punkt M_i ist ja Mittelpunkt eines echten Intervalls und liegt somit rechts von P_i. Daher werden die Mengen OUT_1 bis OUT_i abgezogen, statt, wie oben, nur die Mengen OUT_1 bis OUT_{i-1}.

Daß diese Beobachtung tatsächlich richtig ist, also im ersten Fall

$$\{H \in \mathcal{H} \mid M_i \in H\} = \bigcup_{j=1}^{i} IN_j \setminus \bigcup_{j=1}^{i-1} OUT_j$$

und im zweiten Fall

$$\{H \in \mathcal{H} \mid M_i \in H\} = \bigcup_{j=1}^{i} IN_j \setminus \bigcup_{j=1}^{i} OUT_j$$

gilt, ist der Inhalt von Lemma 5, das wir in Abschnitt 5.4 beweisen werden.

Man kann nun die Mengen IN_i (OUT_i) an jedem Haltepunkt sukzessive zu $Hset$ hinzufügen (von $Hset$ abziehen) und erhält so den Algorithmus aus Abbildung 5.3.

$Hset := \bigcup_{j=1}^{l-1} IN_j \setminus \bigcup_{j=1}^{l-1} OUT_j$ /* $P_l = L(Q)$ */
for all stopping points $M_i \in \mathcal{M}$ **do**
  $Hset := Hset \cup IN_i$
  if $L(Q) \neq H(Q)$ **then** $Hset := Hset \setminus OUT_i$
  if $Hset = \emptyset$ **then** answer $\perp$ **endif**
endfor
answer $\top$

Abbildung 5.3: Der Dimension Scanning Algorithmus im Eindimensionalen. Eingabe ist die Scan-Liste einer (eindimensionalen) QCOVER Instanz.

Es bezeichnet l in Abbildung 5.3 wie oben den Index des Punktes $P_l \in \mathcal{P}$, der gleich der unteren Grenze des Anfrageintervalls ist, für den also $P_l = L(Q)$ gilt. Die Menge $Hset$ wird an jedem Haltepunkt M_i genau wie gewünscht berechnet:

- Ist $L(Q) = H(Q)$, so wird am einzigen Haltepunkt $M_l = L(Q) \in \mathcal{M}$ die Menge OUT_l *nicht* von $Hset$ abgezogen, es berechnet sich also $Hset$ korrekt zu $Hset = \bigcup_{j=1}^{l} IN_j \setminus \bigcup_{j=1}^{l-1} OUT_j$.
- Ist hingegen $L(Q) \neq H(Q)$, so wird an jedem Punkt OUT_i von $Hset$ abgezogen und wir erhalten jedesmal korrekt $Hset = \bigcup_{j=1}^{i} IN_j \setminus \bigcup_{j=1}^{i} OUT_j$.

Daß der Algorithmus korrekt arbeitet, sollte also nach den bisherigen Überlegungen klar sein. Der Beweis dafür ist ein (nämlich der eindimensionale) Spezialfall des Korrektheitsbeweises für den allgemeinen Dimension Scanning Algorithmus. Wir beweisen die Korrektheit von Dimension Scanning in Abschnitt 5.4.

5.3 Der allgemeine Fall

Die Ausführungen im vorangehenden Abschnitt mögen unnötig ausführlich erschienen sein, da ja die Überdeckung eines einzelnen Intervalls eine recht triviale Angelegenheit ist; wir können aber tatsächlich aus dem eindimensionalen Verfahren ohne großen Arbeitsaufwand ein allgemeines Verfahren machen. Wie bisher gehen wir dazu über die Scan-Liste, merken uns jeweils die aktiven Evidenzquader und prüfen an bestimmten Punkten, ob Q dort nicht überdeckt wird. Im Eindimensionalen ist das klar: Q wird nicht überdeckt, wenn die Menge der aktiven Intervalle (bzw. eindimensionalen Quader) leer ist. Im Mehrdimensionalen muß man sich das etwas genauer überlegen.

5.3.1 Einschränkung der Scan-line

Zunächst bezeichnen wir mit $\mathcal{M}^d$ die Scan-Liste *in Dimension d*. Diese Scan-Liste ist vollkommen analog zu Abschnitt 5.2.2 definiert; für die Unterteilung der jeweiligen Achse in Diskretisierungsintervalle werden genau die Endpunkte der Intervalle aller Quader in Dimension d herangezogen. Wir bezeichnen die Menge aller Intervallendpunkte in Dimension d erwartungsgemäß mit $\mathcal{P}^d$. Weiter gehen wir davon aus, daß die Dimensionen in einer beliebigen Reihenfolge von 1 bis n durchnummeriert sind.

Der springende Punkt ist nun, sich zu überlegen, welche Evidenzquader denn *überhaupt noch aktiv sein können*. Im Eindimensionalen sind das genau die, die den aktuellen Haltepunkt enthalten. Im Mehrdimensionalen muß man sich hingegen auch auf die bereits abgearbeiteten Dimensionen beziehen: Die Intervalle aktiver Evidenzquader müssen alle bisher, d.h. in den unteren Dimensionen, ausgewählten Haltepunkte enthalten. Anders ausgedrückt: Wir beginnen mit der ersten Dimension und scannen diese, wie in Abschnitt 5.1 angedeutet, mit der Scan-line $\{(x^1, \ldots, x^n) \in \mathbb{Q}^n \mid x^1 = M^1\}$, die wir an jedem Punkt $M^1 \in \mathcal{M}^1$ anlegen; dabei bestimmen wir jeweils die Menge der Evidenzquader, die von dieser Scan-line geschnitten werden (das ist genau die Menge der Evidenzquader, deren erstes Intervall M^1 enthält). Anschließend scannen wir die nächsthöhere Dimension, wobei nur noch Evidenzquader in Frage kommen, die in der vorherigen Dimension aktiv waren; wir *schränken also die Scan-line ein*, nämlich auf die Menge $\{(x^1, \ldots, x^n) \in \mathbb{Q}^n \mid x^1 = M^1 \wedge x^2 = M^2\}$. An jedem Punkt $M^2 \in \mathcal{M}^2$ wird nun bestimmt, welche Evidenzquader auch diese eingeschränkte Scan-line noch schneiden. Das sind nämlich genau diejenigen Evidenzquader, deren erste Intervalle M^1 und M^2 enthalten, die also geeignet sind, Punkte mit den Koordinaten M^1 und M^2 in den ersten beiden Dimensionen zu überdecken. Dies sind die Punkte, auf die es ankommt: Es sei M^1 der Mittelpunkt des Diskretisierungsintervalls A^1, und ebenso M^2 Mittelpunkt von A^2; der Algorithmus konzentriert sich an diesem Punkt in der Rekursion auf die Diskretisierungsquader *an dieser Stelle*, d.h. auf die Quader, die in Dimension 1 und 2 von A^1 und A^2 gebildet werden. Die Mittelpunkte all dieser Quader haben natürlich die ersten beiden Koordinaten M^1 und M^2.

Wir fahren genauso fort, schränken also mit jedem Schritt in der Rekursion die Scan-line weiter ein, und erhalten dadurch jeweils in der nächsthöheren Dimension ein weniger hoch dimensioniertes geometrisches Objekt; wie gewünscht wird also jeweils das n-dimensionale Problem in eine dynamische Folge n-1-dimensionaler Probleme aufgespalten. Im Dreidimensionalen erhält man etwa in der ersten Dimension eine Ebene, in der zweiten eine Gerade und in der dritten nur noch einen Punkt als Scan-line.

Das Einschränken der Scan-line kann programmtechnisch so realisiert werden, daß der nächsthöheren Dimension jeweils eine eingeschränkte *Grundmenge* von Evidenzquadern übergeben wird, nämlich die Evidenzquader, die *bis zu dieser Dimension* noch aktiv waren. Die erste Dimension übergibt also an die zweite Dimension jeweils die Menge der Evidenzquader, deren Intervall in Dimension 1 den Punkt M^1 enthält, die zweite Dimension übergibt dann wiederum an die dritte alle Quader *aus dieser Menge*, die zusätzlich in der zweiten Dimension den Punkt M^2 enthalten usw. Wird an irgendeinem Punkt M^d die Menge der Quader, die alle bisherigen Haltepunkte enthalten, sprich die Menge der Quader, die von der eingeschränkten Scan-line geschnitten werden, leer, so haben wir mindestens einen Mittelpunkt eines Diskretisierungsquaders gefunden, der in keinem Evidenzquader enthalten ist; genauer werden *alle* Mittelpunkte mit den ersten d Koordinaten

$M^1 \dots M^d$ nicht überdeckt. Wir können also in diesem Fall mit $\perp$ antworten. Erreicht hingegen die Rekursion die n+1-te Dimension, so wurde kein nicht-überdeckter Mittelpunkt gefunden, wir müssen also mit $\top$ antworten und weitersuchen.

Natürlich wird die Grundmenge der Evidenzquader in der ersten Dimension mit $\mathcal{H}$, also der Menge aller Evidenzquader, initialisiert. Wir bezeichnen die jeweils zur Verfügung stehende Grundmenge von Evidenzquadern mit $\mathcal{H}set$ und formulieren das Dimension Scanning Verfahren gemäß Abbildung 5.3. Dabei bezeichnet l^d den Index des Punktes $P^d_{l^d}$ in $\mathcal{P}^d$, der dem unteren Endpunkt des Anfragequaders in d entspricht, für den also $P^d_{l^d} = L(Q^d)$ gilt.

```
if d = n + 1 then answer ⊤ endif
Hset := ⋃_{j=1}^{l^d−1} (IN_j^d ∩ ℋset) \ ⋃_{j=1}^{l^d−1} OUT_j^d  /* P_{l^d}^d = L(Q^d) */
for all stopping points M_i^d ∈ ℳ^d do
    Hset := Hset ∪ (IN_i^d ∩ ℋset)
    if L(Q^d) ≠ H(Q^d) then Hset := Hset \ OUT_i^d endif
    if Hset = ∅ then answer ⊥ endif
    if Dimension Scanning(d + 1, Hset) answers ⊥ then
      answer ⊥
    endif
endfor
answer ⊤
```

Abbildung 5.4: Der allgemeine Dimension Scanning Algorithmus. Eingabe sind die Scan-Listen einer QCOVER Instanz, die aktuelle Dimension d sowie die (eingeschränkte) Grundmenge $\mathcal{H}set$ von Evidenzquadern.

Durch den Schnitt der IN Mengen mit $\mathcal{H}set$ werden offenbar immer nur Elemente aus $\mathcal{H}set$ zu $Hset$ hinzugenommen. Der Anfragequader Q setzt sich im Kontext von Abbildung 5.4 wie üblich aus den Intervallen $Q = Q^1 \times \dots \times Q^n$ zusammen. Mit $\mathcal{M}^d$ bezeichnen wir, wie bereits erwähnt, die Menge der Haltepunkte in jeder Dimension, also die Mittelpunkte der Diskretisierungsintervalle; diese sind vollkommen analog zum eindimensionalen Fall definiert, also $M^d_i = \frac{P^d_i + P^d_{i+1}}{2}$, falls das Anfrageintervall ein echtes Intervall ist, und $M^d_i = L(Q^d)$ falls $L(Q^d) = H(Q^d)$ gilt.

Genau wie im eindimensionalen Fall gehen wir davon aus, daß in jeder Dimension d für jeden Punkt $P^d_i \in \mathcal{P}^d$ die Information IN^d_i und OUT^d_i vorliegt, also die Menge der Evidenzquader, deren Intervall in dieser Dimension seine untere bzw. obere Grenze bei P^d_i hat. Wir betrachten zunächst ein Beispiel, um den Verlauf des Algorithmus zu verdeutlichen.

5.3.2 Ein Beispiel

Dieser Abschnitt illustriert den Algorithmus aus Abbildung 5.4 anhand eines kleinen zweidimensionalen Beispiels. Siehe dazu Abbildung 5.5.

Es ist also eine zweidimensionale QCOVER Instanz mit zwei Evidenzquadern (bzw. Rechtecken) gegeben; die Scan-Listen in x- und y-Richtung ergeben sich aus den Projektionen der Rechtecke auf die jeweilige Achse, siehe Abbildung 5.6.

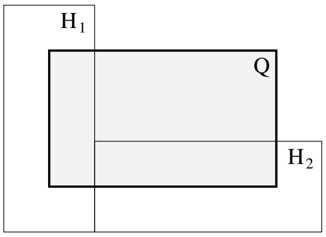

Abbildung 5.5: Beispiel für die gegenseitige Lage von zwei Evidenzrechtecken und dem Anfragerechteck.

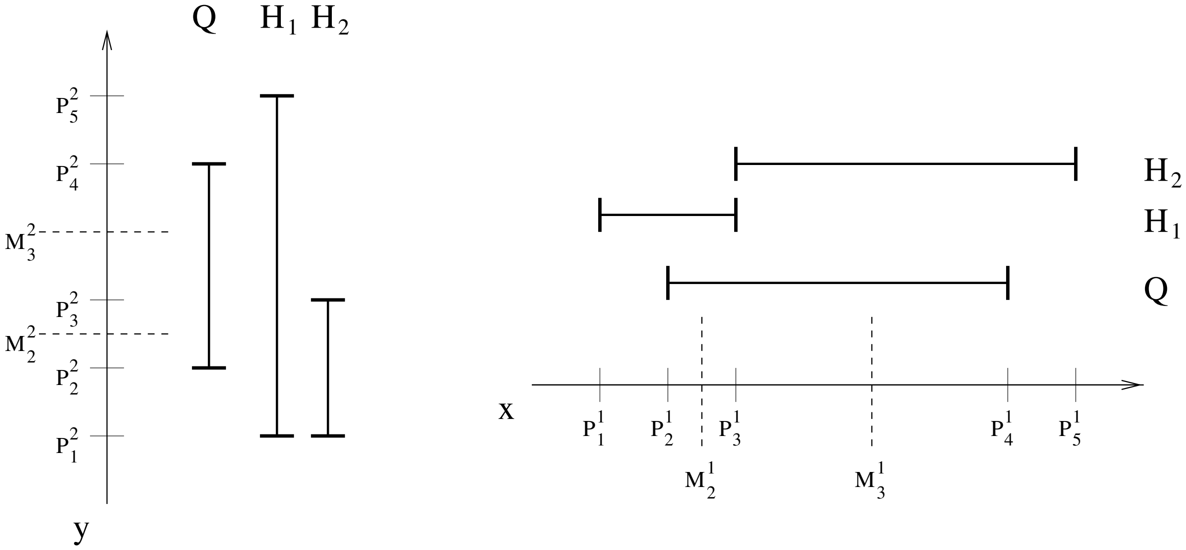

Abbildung 5.6: Die Scan-Listen der Rechtecke aus Abbildung 5.5; die Darstellung orientiert sich an Abbildung 5.2.

Ähnlich wie in Abbildung 5.2 sind die Intervalle der beiden Evidenz- und des Anfragerechtecks zur besseren Übersicht getrennt dargestellt. Wir geben der x-Achse die Nummer 1 sowie der y-Achse die Nummer 2. Damit ergibt sich $l^1 = 2$ weil in x-Richtung der zweite Punkt P_2^1 mit der unteren Grenze von Q zusammenfällt; genauso erhalten wir $l^2 = 2$. Weiter ergeben sich die Scan-Listen zu $\mathcal{M}^1 = \{M_2^1, M_3^1\}$ und $\mathcal{M}^2 = \{M_2^2, M_3^2\}$.

In Abbildung 5.7 sind die Punkte dargestellt, an denen sich die senkrechten (bzw. waagerechten) Linien durch die Punkte M_2^1 und M_3^1, bzw. M_2^2 und M_2^3 kreuzen.

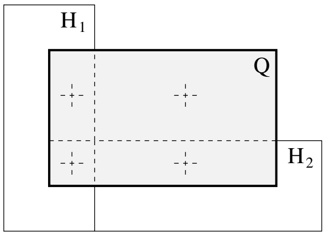

Abbildung 5.7: Die Rechtecke in der Anfragediskretisierung des Beispiels aus Abbildung 5.5 und ihre Mittelpunkte.

Diese Punkte sind genau die Mittelpunkte der Quader in der Anfragediskretisierung, die vom Dimension Scanning Verfahren geprüft werden. Für die IN/OUT Information an jedem Punkt $P_i^d \in \mathcal{P}^d$ erhalten wir folgende Werte:

- in x-Richtung: $IN_1^1 = \{H_1\}$, $OUT_1^1 = \emptyset$, $IN_2^1 = \emptyset$, $OUT_2^1 = \emptyset$, $IN_3^1 = \{H_2\}$, $OUT_3^1 = \{H_1\}$, $IN_4^1 = \emptyset$, $OUT_4^1 = \emptyset$, $IN_5^1 = \emptyset$, $OUT_5^1 = \{H_2\}$.
- in y-Richtung: $IN_1^2 = \{H_1, H_2\}$, $OUT_1^2 = \emptyset$, $IN_2^2 = \emptyset$, $OUT_2^2 = \emptyset$, $IN_3^2 = \emptyset$, $OUT_3^2 = \{H_2\}$, $IN_4^2 = \emptyset$, $OUT_4^2 = \emptyset$, $IN_5^2 = \emptyset$, $OUT_5^2 = \{H_1\}$.

Wir verfolgen nun den Verlauf des Dimension Scanning Algorithmus auf diesem Beispiel. Im initialen Aufruf ist $d = 1$ und $\mathcal{H}set = \mathcal{H} = \{H_1, H_2\}$. Es wird also zunächst $Hset$ zu $Hset = \bigcup_{j=1}^{l^1-1}(IN_j^1 \cap \{H_1, H_2\}) \setminus \bigcup_{j=1}^{l^1-1} OUT_j^1 = \bigcup_{j=1}^{1} IN_j^1 \setminus \bigcup_{j=1}^{1} OUT_j^1 = \{H_1\}$ berechnet. Beim ersten Haltepunkt $M_2^1 \in \mathcal{M}^1$ ändert sich hieran nichts, da sowohl IN_2^1 als auch OUT_2^1 leer sind. Es erfolgt nun ein rekursiver Aufruf mit den Parametern $\mathcal{H}set = \{H_1\}$ und $d = 2$; das Verfahren scannt also nun die y-Richtung mit auf H_1 eingeschränkter Grundmenge. Dies macht Sinn, da ja das andere Rechteck "weiter rechts" liegt.

Beim Scannen der zweiten Dimension wird nun zu Beginn $Hset = \bigcup_{j=1}^{1}(IN_j^2 \cap \{H_1\}) \setminus \bigcup_{j=1}^{1} OUT_j^2 = \{H_1\}$ berechnet. Genau wie in x-Richtung ändert sich daran wegen der leeren IN/OUT Mengen am ersten Haltepunkt M_2^2 nichts. Es bleibt also $Hset = \{H_1\} \neq \emptyset$ und ein weiterer rekursiver Aufruf mit $d = 3$ wird getätigt, der sofort $\top$ zurückgibt. Dies entspricht der Tatsache, daß der Mittelpunkt des unteren linken Diskretisierungsrechtecks (siehe Abbildung 5.7) von H_1 überdeckt wird. Am nächsten Haltepunkt M_3^2 ändert sich an $Hset$ wieder nichts: die Menge OUT_3^2, die abgezogen wird, enthält nur H_2; dieses Rechteck ist aber ohnehin nicht in $Hset$ enthalten. Wiederum stellt der Algorithmus fest, daß H_1 für die Überdeckung ausreicht, bzw. daß der Mittelpunkt des oberen linken Diskretisierungsrechtecks aus Abbildung 5.7 in H_1 enthalten ist.

Das Verfahren erreicht nun in y-Richtung das Ende der Scan-Liste und gibt $\top$ zurück. Damit kehren wir in den initialen Aufruf auf Dimension 1 zurück und gehen weiter zum nächsten Haltepunkt M_3^1. Hier wird wegen $IN_3^1 = \{H_2\}$ das zweite Rechteck H_2 zu $Hset$ hinzugenommen, gleich darauf aber auch mit $OUT_3^1 = \{H_1\}$ das erste Rechteck abgezogen, es ergibt sich also $Hset = \{H_2\}$. Wiederum erfolgt ein rekursiver Aufruf für Dimension 2, diesmal mit der Grundmenge $\mathcal{H}set = Hset = \{H_2\}$.

Beim Scan in y-Richtung wird nun zu Beginn $Hset = \bigcup_{j=1}^{1}(IN_j^2 \cap \{H_2\}) \setminus \bigcup_{j=1}^{1} OUT_j^2 = \{H_2\}$ berechnet. Da die IN/OUT Information bei M_2^2 leer ist, bleibt das beim ersten Haltepunkt auch so, wir haben $Hset = \{H_2\} \neq \emptyset$, was korrekt reflektiert, daß der Mittelpunkt des unteren rechten Diskretisierungsrechtecks aus Abbildung 5.7 in H_2 enthalten ist. Beim nächsten Haltepunkt M_3^2 nun geschieht das Entscheidende: wegen $H_2 \in OUT_3^2$ wird H_2 von $Hset$ abgezogen und wir erhalten $Hset = \emptyset$. Das ist richtig: der Mittelpunkt des oberen rechten Diskretisierungsrechtecks aus Abbildung 5.7 ist in keinem Evidenzrechteck enthalten. Der rekursive Aufruf in y-Richtung bemerkt, daß $Hset$ leer ist und antwortet mit $\perp$. Diese Antwort wird vom initialen Aufruf hochpropagiert und wir erhalten die richtige Gesamtantwort, nämlich, daß das Anfragerechteck *nicht* überdeckt wird.

5.3.3 Optimierung

Wie wir in Abschnitt 5.2 gesehen haben, kann man in einer Dimension mit einem einzigen Durchlauf über die Scan-Liste feststellen, ob das Anfrageintervall überdeckt wird. Insbesondere konstruiert man dabei für jeden Punkt M_i in der Scan-Liste die Menge $\{H \in \mathcal{H} \mid M_i \in H\}$ der aktiven Intervalle. Führt man sich dies vor Augen, so wird deutlich, daß der Algorithmus aus Abbildung 5.4 unnötige Arbeit verrichtet.

Zunächst einmal wird jedesmal, wenn im Verlauf der Rekursion die Funktion auf einer Dimension d aufgerufen wird, die Menge

$$\bigcup_{j=1}^{l^d-1}(IN_j^d \cap \mathcal{H}set) \setminus \bigcup_{j=1}^{l^d-1} OUT_j^d$$

berechnet. Offenbar kann man $\mathcal{H}set$ ausklammern, die berechnete Menge ist also genau

$$(\bigcup_{j=1}^{l^d-1} IN_j^d \setminus \bigcup_{j=1}^{l^d-1} OUT_j^d) \cap \mathcal{H}set$$

Nun ist $\mathcal{H}set$ die einzige Größe in dieser Formel, die sich von Aufruf zu Aufruf ändern kann; die beiden Mengenvereinigungen und ihre Differenz führen jedesmal zum selben Ergebnis. Es würde also ausreichen, die Mengenvereinigungen *einmal* zu berechnen und dann später nur noch den jeweiligen Schnitt mit $\mathcal{H}set$ auszuführen.

Man kann sogar noch einen Schritt weiter gehen: Im Zuge des Preprocessings können für jede Dimension d an allen Punkten $M_i^d \in \mathcal{M}^d$ die Mengen ACT_i^d der Evidenzquader, deren Intervall in Dimension d den Punkt M_i^d enthält, berechnet werden. Genauer definieren wir:

$$ACT_i^d := \begin{cases} \bigcup_{j=1}^{i} IN_j^d \setminus \bigcup_{j=1}^{i-1} OUT_j^d & L(Q^d) = H(Q^d) \\ \bigcup_{j=1}^{i} IN_j^d \setminus \bigcup_{j=1}^{i} OUT_j^d & L(Q^d) \neq H(Q^d) \end{cases} \tag{5.2}$$

Wir erhalten diese Mengen ACT_i^d, indem wir für jede Dimension einmal über die Scan-Liste gehen und sukzessive die entsprechenden IN (OUT) Mengen hinzunehmen (abziehen). Die Laufzeit für diese Berechnungen wird asymptotisch von der Laufzeit des Gesamtalgorithmus dominiert, siehe dazu Abschnitt 5.5. Wir gehen also davon aus, daß wir an allen Punkten M_i^d die entsprechende Menge ACT_i^d der potentiell aktiven Evidenzquader zur Verfügung stehen haben. Daß ACT_i^d korrekt definiert ist, also $ACT_i^d = \{H = I^1 \times \ldots \times I^n \in \mathcal{H} \mid M_i^d \in I^d\}$ gilt, sehen wir in Lemma 5.

```
if d = n + 1 then answer ⊤ endif
for all stopping points M_i^d ∈ 𝓜^d do
    Hset := ACT_i^d ∩ 𝓗set
    if Hset = ∅ then answer ⊥ endif
    if Dimension Scanning(d + 1, Hset) answers ⊥ then
      answer ⊥
    endif
endfor
answer ⊤
```

Abbildung 5.8: Endgültige Fassung des Dimension Scanning Algorithmus. Eingabe sind die Scan-Listen einer QCOVER Instanz, die aktuelle Dimension d sowie die (eingeschränkte) Grundmenge $\mathcal{H}set$ von Evidenzquadern.

Abbildung 5.8 zeigt die endgültige Fassung des Dimension Scanning Verfahrens. Die Quader in ACT_i^d sind *potentiell aktiv* in dem Sinn, daß sie in Dimension d aktiv sind, aber in anderen, bereits in der Rekursion abgearbeiteten Dimensionen, nicht notwendig auch aktiv sein müssen. Um dies zu prüfen, muß lediglich noch der Schnitt mit $\mathcal{H}set$ berechnet werden. Wir haben also nicht nur die anfängliche Berechnung der Mengenvereinigung in jeder Dimension gespart, sondern auch das ständige Hinzufügen und Abziehen der IN/OUT Mengen.

5.4 Korrektheit

Wir beweisen nun, daß der Dimension Scanning Algorithmus korrekt arbeitet, d.h., daß er genau dann $\top$ antwortet, wenn Q von $\mathcal{H}$ überdeckt wird, also $Q \sqsubset \mathcal{H}$ gilt. Dazu müssen wir im Grunde nichts weiter tun, als einige Eigenschaften des Algorithmus formal auszudrücken.

Zunächst werden wir sehen, daß das sukzessive Hinzunehmen und Abziehen der IN/OUT Information zum gewünschten Ergebnis führt. Es sei noch einmal bemerkt, daß die Mengen ACT_i^d genau diejenigen Mengen sind, die an jedem Punkt M_i^d durch dieses sukzessive Hinzunehmen und Abziehen entstehen.

Lemma 5 *In jeder Dimension d gilt an beliebigen Haltepunkten $M_i^d \in \mathcal{M}^d$ mit Definition 5.2:*

$$ACT_i^d = \{H = I^1 \times \ldots \times I^n \in \mathcal{H} \mid M_i^d \in I^d\}$$

Beweis:
Wir unterscheiden zwei Fälle.

Fall 1: $L(Q^d) = H(Q^d)$. In diesem Fall enthält $\mathcal{M}^d$ lediglich *einen* Haltepunkt, nämlich $M_i^d = L(Q^d)$. Die Intervalle I^d, die diesen Punkt enthalten, sind genau die, deren untere Grenze höchstens bei $L(Q^d)$ und deren obere Grenze mindestens bei $L(Q^d)$ liegt, also die I^d mit $L(I^d) \le L(Q^d)$ und $H(I^d) \ge L(Q^d)$. Man erhält alle I^d mit $L(I^d) \le L(Q^d)$ durch die Vereinigung der IN-Mengen, genauer sind in $\bigcup_{j=1}^{i} IN_j^d$ alle Evidenzquader enthalten, deren untere Intervallgrenze höchstens bei $L(Q^d)$ liegt. Andererseits sind in $\bigcup_{j=1}^{i-1} OUT_j^d$ alle Quader enthalten, deren obere Grenze *echt vor* $L(Q^d)$ liegt. Insgesamt erhält man also das korrekte Ergebnis, wenn man die zweite von der ersten Menge abzieht, d.h., die Behauptung ist gültig für $ACT_i^d = \bigcup_{j=1}^{i} IN_j^d \setminus \bigcup_{j=1}^{i-1} OUT_j^d$.

Fall 2: $L(Q^d) \neq H(Q^d)$. Es sei M_i^d ein beliebiger Haltepunkt in $\mathcal{M}^d$. Da das Anfrageintervall ein echtes Intervall ist, ist auch M_i^d Mittelpunkt eines echten Intervalls, d.h., $M_i^d = \frac{P_i^d + P_{i+1}^d}{2}$ für zwei aufeinanderfolgende Punkte $P_i^d, P_{i+1}^d \in \mathcal{P}^d$. Wir argumentieren nun ganz ähnlich wie im obigen ersten Fall: Die Quader, deren Intervall in Dimension d den Punkt M_i^d enthält, sind offenbar genau die, für die die Bedingungen

1. $L(I^d) \le M_i^d$ und
2. $H(I^d) \ge M_i^d$

gelten. Wir erhalten alle Quader, die die erste Bedingung erfüllen, indem wir die bisherigen IN-Mengen vereinigen, also mit $\bigcup_{j=1}^{i} IN_j^d$; es genügt, alle Quader bis zu IN_i^d zu betrachten, da die Quader aus IN_{i+1}^d ja per definitionem ihre untere Intervallgrenze bei P_{i+1}^d haben; dies ist bereits größer als M_i^d, siehe oben. Andererseits kann $\bigcup_{j=1}^{i} IN_j^d$ aber auch zu viele Quader enthalten, nämlich die, die vor M_i^d bereits wieder aufhören, die also die zweite Bedingung nicht erfüllen. Dies sind genau die Quader, die ihre obere Intervallgrenze an einem der Punkte $P_1^d \dots P_i^d$ haben, die also in $\bigcup_{j=1}^{i} OUT_j^d$ enthalten sind. Subtrahiert man nun diese Menge von der Vereinigung der IN-Mengen, so erhält man genau die Quader, deren Intervall in Dimension d den Punkt M_i^d enthält. Da ACT_i^d gerade als $ACT_i^d = \bigcup_{j=1}^{i} IN_j^d \setminus \bigcup_{j=1}^{i} OUT_j^d$ definiert ist, folgt die Behauptung. ■

Wie wir in Abschnitt 5.3.1 gesehen haben, wählt der Dimension Scanning Algorithmus auf jeder Rekursionsstufe d einen Haltepunkt $M_{i^d}^d$ und schränkt die Scan-line in d auf diesen Punkt ein, bis schließlich in Dimension n nur noch der Punkt $(M_{i^1}^1, \dots M_{i^n}^n)$ übrig ist. Bezogen auf Abbildung 5.8 bedeutet das, daß an jeder Stelle in der Rekursion die neu berechnete Menge $Hset := ACT_i^d \cap \mathcal{H}set$ genau die Evidenzquader enthält, die von der eingeschränkten Scan-line geschnitten werden. Das folgende Lemma enthält eine formale Präzisierung dieser Aussage.

Lemma 6 *Bearbeitet der* Dimension Scanning *Algorithmus die Dimension d, und sind* $M_{i^1}^1 \dots M_{i^d}^d$ *die auf den Rekusionsstufen bis d gewählten Haltepunkte, so gilt:*

$$ACT_{i^d}^d \cap \mathcal{H}set = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^1}^1 \in I^1 \wedge \dots \wedge M_{i^d}^d \in I^d\}.$$

Beweis:
Wir beweisen die Behauptung durch eine einfache Induktion über die Dimension d. Die Grundmenge $\mathcal{H}set$ der Evidenzquader in Dimension d bezeichnen wir mit $\mathcal{H}set^d$.

Induktionsanfang $d = 1$: Die Behauptung ist hier $ACT_{i^1}^1 \cap \mathcal{H}set^1 = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^1}^1 \in I^1\}$. Wegen $\mathcal{H}set^1 = \mathcal{H}$ ist das gleichbedeutend mit $ACT_{i^1}^1 = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^1}^1 \in I^1\}$ und die Behauptung ergibt sich direkt aus Lemma 5.

Induktionsschritt $d \longrightarrow d + 1$: Aufgrund der Struktur des Algorithmus ist $\mathcal{H}set^{d+1} = ACT_{i^d}^d \cap \mathcal{H}set^d$: Diese Menge wird ja als neue Grundmenge übergeben. Nach Induktionsvorraussetzung ist $ACT_{i^d}^d \cap \mathcal{H}set^d = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^1}^1 \in I^1 \wedge \dots \wedge M_{i^d}^d \in I^d\}$. Nach Lemma 5 ist andererseits $ACT_{i^{d+1}}^{d+1} = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^{d+1}}^{d+1} \in I^{d+1}\}$. Damit erhalten wir für den Schnitt dieser beiden Mengen $ACT_{i^{d+1}}^{d+1} \cap \mathcal{H}set^{d+1} = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^1}^1 \in I^1 \wedge \dots \wedge M_{i^d}^d \in I^d \wedge M_{i^{d+1}}^{d+1} \in I^{d+1}\}$, womit die Induktion bereits abgeschlossen ist. ■

Nachdem nun die Brücke zwischen den im Algorithmus erzeugten Mengen und ihren semantischen Eigenschaften geschlagen ist, können wir die Korrektheit von Dimension Scanning durch eine relativ einfache Betrachtung des Programmflusses beweisen. Wir benutzen dabei die in Abschnitt 2.3 gezeigten Lemmata 2 und 3, die zusammengenommen besagen, daß der Anfragequader genau dann überdeckt wird, wenn die Mittelpunkte aller Quader der Anfragediskretisierung überdeckt werden. Es gilt der folgende Satz.

Satz 4 *Der* Dimension Scanning *Algorithmus antwortet* $\perp$ *genau dann, wenn* $Q \not\subseteq \mathcal{H}$.

Beweis:
$\Rightarrow$: Der Algorithmus antwortet genau dann $\perp$, wenn bei irgendeinem Punkt $M_{i^d}^d$ einmal $Hset$ zu $Hset = \emptyset$ berechnet wird. Es seien $M_{i^1}^1 \dots M_{i^{d-1}}^{d-1}$ die aktuellen Haltepunkte in den unter d liegenden Dimensionen. Gemäß Lemma 6 ist $Hset = ACT_{i^d}^d \cap \mathcal{H}set = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M_{i^1}^1 \in I^1 \wedge \dots \wedge M_{i^d}^d \in I^d\}$. Mit $Hset = \emptyset$ ist diese Menge leer.

Es gibt also keine Evidenzquader, die irgendwelche Punkte mit den ersten d Koordinaten $M^1_{i^1} \dots M^d_{i^d}$ überdecken. Da nach Wahl der Haltepunkte in jeder Dimension $1 \leq d' \leq d$ insbesondere $M^{d'}_{i^{d'}} \in Q^{d'}$ gilt, ist die Menge $\{M^1_{i^1}\} \times \dots \times \{M^d_{i^d}\} \times Q^{d+1} \times \dots \times Q^n$ in Q enthalten. Jeder Punkt in dieser Menge wird nicht überdeckt, also gilt erst recht $Q \not\sqsubseteq \mathcal{H}$.

$\Leftarrow$: Es gelte nun umgekehrt $Q \not\sqsubseteq \mathcal{H}$. Mit Lemma 2 wird dann ein Quader a aus der Anfragediskretisierung nicht überdeckt, mit Lemma 3 wird sein Mittelpunkt nicht überdeckt. Es sei $a = A^1 \times \dots \times A^n$. Wir können ohne Einschränkung davon ausgehen, daß der Mittelpunkt von a als Haltepunkt benutzt wird:

$$M(a) = (M^1_{i^1}, \dots, M^n_{i^n}) \text{ mit } \forall d : M^d_{i^d} \in \mathcal{M}^d \tag{5.3}$$

Ist dies gezeigt, so verläuft die Argumentation wie folgt: Der Algorithmus erreicht irgendwann die Stelle in der Rekursion, an der die Haltepunkte $M^1_{i^1} \dots M^n_{i^n}$ ausgewählt sind. Wird diese Stelle *nicht* erreicht, so wurde bereits vorher $\perp$ geantwortet und es ist nichts mehr zu zeigen. Mit Lemma 6 wird an dieser Stelle $Hset := ACT^n_{i^n} \cap \mathcal{H}set$ zu $Hset = \{H = I^1 \times \dots \times I^n \in \mathcal{H} \mid M^1_{i^1} \in I^1 \wedge \dots \wedge M^n_{i^n} \in I^n\}$ berechnet. Da nach Voraussetzung, siehe oben, der Punkt $(M^1_{i^1}, \dots, M^n_{i^n})$ nicht überdeckt wird, ist diese Menge leer. Der Algorithmus bemerkt dies und gibt $\perp$ zurück, die Antwort wird durch die Rekursionsstruktur hochpropagiert. Es folgt die Behauptung.

Zu zeigen ist jetzt noch Aussage 5.3. In den Dimensionen d, wo das Intervall A^d von a mehr als einen Punkt enthält, ist das klar: A^d ist ein echtes atomares Intervall mit $A^d \subseteq Q^d$ (a ist ja ein Quader aus der Anfragediskretisierung), dessen Mittelpunkt zu $\mathcal{M}^d$ hinzugenommen wird, siehe die Definition 5.1 von $\mathcal{M}$ in Abschnitt 5.2.2. Dieser Fall ist also unkritisch. Ebenso unkritisch sind Dimensionen d, in denen das Anfrageintervall Q^d nur ein Punkt ist; in diesem Fall folgt $A^d = Q^d = \{L(Q^d)\}$ und $L(Q^d) \in \mathcal{M}^d$. Kritisch ist die Situation nur, wenn beide Fälle für d nicht erfüllt sind. Dann kann es sein, daß der Mittelpunkt (bzw. der einzige Punkt) von A^d nicht in $\mathcal{M}^d$ enthalten ist; dann werden aber auch die *Nachbarn* von a in dieser Dimension nicht überdeckt. Genauer: es gelte $A^d = \{P^d_i\}$ für $P^d_i \in \mathcal{P}^d$. Weil A^d von Q^d überdeckt wird, haben wir $P^d_i \in Q^d$. Da das Anfrageintervall kein einzelner Punkt ist, hat P^d_i einen Nachbarn in $\mathcal{P}^d$, P^d_{i-1} oder P^d_{i+1} nämlich, der ebenfalls in Q^d enthalten ist. Es sei ohne Einschränkung P^d_{i+1} dieser Nachbar. Der Quader $a' := A^1 \times \dots \times A^{d-1} \times [P^d_i, P^d_{i+1}] \times A^{d+1} \times \dots \times A^n$ ist offensichtlich Element der Anfragediskretisierung und wird nicht überdeckt; sonst wäre ja auch a überdeckt. Außerdem ist sein Intervall $[P^d_i, P^d_{i+1}]$ in Dimension d ein echtes atomares Intervall, also unkritisch. Wir ersetzen nun einfach a durch a'. Tut man dies für jede kritische Dimension d, so erhält man einen nicht überdeckten Quader in der Anfragediskretisierung, für den Aussage 5.3 erfüllt ist. Damit ist der Beweis abgeschlossen. ■

Mit Satz 4 ist natürlich auch gezeigt, daß Dimension Scanning genau dann mit $\top$ antwortet, wenn $Q \sqsubseteq \mathcal{H}$ gilt.

5.5 Laufzeit

Wir schließen dieses Kapitel mit der, vergleichsweise leicht durchführbaren, Laufzeitanalyse ab. Dazu untersuchen wir zunächst den Aufwand an jedem Haltepunkt und geben dann eine Rekursionsformel für den Gesamtaufwand im worst-case an. Das Preprocessing behandeln wir in Abschnitt 5.5.2.

5.5.1 Mengen, Scan-Listen und Rekursion

Betrachten wir die Definition des Dimension Scanning Algorithmus in Abbildung 5.8. Zunächst geht es uns darum, festzustellen, wieviel Zeit für die Berechnungen an jeweils einem Haltepunkt benötigt wird. Die einzig nicht-triviale Aktion ist (abgesehen vom rekursiven Aufruf natürlich, dazu später mehr) die Ausführung des Mengenschnitts von ACT_i^d mit $\mathcal{H}set$.

Es sollte klar sein, daß die Evidenzquader in den zur Frage stehenden Mengen durch ihre *Nummern* repräsentiert werden können; zu behandeln sind also Mengen von natürlichen Zahlen. Weiterhin steht ja die maximale Anzahl der Elemente in diesen Mengen von vornherein fest, es sind nämlich immer höchstens $|\mathcal{H}| = k$ Elemente vorhanden.[1] Man kann also in einer Implementierung die geforderten Mengenoperationen sehr bequem auf der Basis von *Bitvektoren* realisieren: Ein Mengenschnitt entspricht dann etwa der AND Operation auf zwei Bitvektoren der Länge k. Die Dauer einer solchen logischen Operation hängt nun von k und der Wortgröße der verwendeten Maschine ab; auf einer 32-Bit Maschine können offenbar Mengenoperationen mit bis zu $k \leq 32$ Elementen mit einem einzigen elementaren Prozessorbefehl durchgeführt werden. Allgemein ist die Dauer für eine so durchgeführte Mengenoperation auf einer Maschine mit Wortgröße W gleich $\lceil \frac{1}{W} * k \rceil = O(k)$; man erreicht also *asymptotisch* keine bessere Laufzeit, als man auch mit anderen geschickten Implementierungen erreichen könnte. Wir halten fest: Die Mengenoperation, und damit die lokalen Berechnungen an jedem Haltepunkt, sind in Zeit $O(k)$ ausführbar.

Die Anzahl der Haltepunkte in jeder Dimension hat die offensichtliche Obergrenze $2k+1$: Es gibt höchstens $2k+2$ Endpunkte, also höchstens $2k+1$ Diskretisierungsintervalle, deren Mittelpunkte ja als Haltepunkte benutzt werden. Andererseits werden nicht alle $2k+1$ Punkte abgearbeitet, wie man sich anhand der möglichen Struktur von $\mathcal{P}$ klar machen kann. Betrachten wir dazu den Anfang von $\mathcal{P}$.[2] Alle Punkte, die echt *vor* $L(Q)$ liegen, werden später gar nicht beachtet; liegen dagegen alle Punkte echt *hinter* $L(Q)$, so bemerkt der Algorithmus bereits am ersten Haltepunkt, daß Q nicht überdeckt wird und bricht ab. Das ist sogar der best-case für das Verfahren. Der worst-case tritt ein, wenn $L(Q)$ und die untere Begrenzung eines Evidenzquaders am Anfang der Liste zusammenfallen. Dann enthält $\mathcal{P}$ aber auch einen Punkt weniger, womit sich ein worst-case von $2k$ Haltepunkten ergibt.

Man kann die Anzahl der rekursiven Aufrufe einschränken, indem man sich folgendes überlegt: wenn der vorherige Haltepunkt überdeckt wurde, und nur neue Quader *hinzugekommen* sind (oder sich die Situation gar nicht geändert hat), so braucht man an dieser Stelle keine Rekursion zu starten; man kann durch Mengenvergleiche in Zeit $O(k)$ feststellen, ob eine solche Situation vorliegt. Mit ähnlichen Betrachtungen wie oben ergibt sich dadurch, daß im schlimmsten Fall tatsächlich nur k Aufrufe getätigt werden; dieser Fall tritt (unter anderem) ein, wenn an jedem Punkt gleichzeitig ein Quader aufhört und genau ein anderer anfängt. Abbildung 5.9 illustriert diese Situation.

Wir erstellen nun eine Rekursionsgleichung für den Gesamtaufwand des Algorithmus. In jeder Dimension werden höchstens $2k$ Haltepunkte betrachtet; das kostet jedesmal (an

[1] Bei einer *dynamischen* Anwendung, d.h., wenn etwa neue Evidenzquader hinzukommen können,trifft man hier auf Schwierigkeiten. Diese können aber umgangen werden, siehe Abschnitt 7.1.

[2] Der Einfachheit halber lassen wir den Index für die Dimension hier weg; die folgenden Betrachtungen sind für alle Dimensionen gültig.

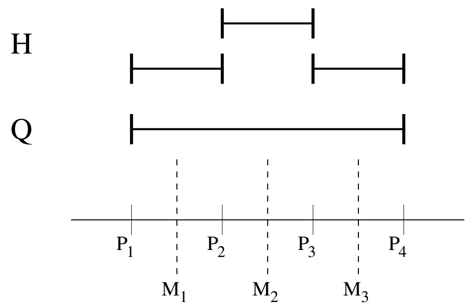

Abbildung 5.9: Ein Beispiel, in dem der worst-case für die Anzahl der rekursiven Aufrufe in einer Dimension eintritt.

jedem Punkt) den gleichen Aufwand. Wir bezeichnen den an jedem Haltepunkt zu erbringenden Aufwand mit c_k und halten fest (siehe oben): $c_k = O(k)$. Es werden höchstens k rekursive Aufrufe getätigt; diese Aufrufe müssen jeweils nur noch eine Dimension weniger betrachten. Offensichtlich erhalten wir so folgende Rekursionsformel.

$$T_{DS}(n,k) = 2kc_k + kT_{DS}(n-1,k) \tag{5.4}$$

Der Aufwand, falls gar keine Dimension mehr zu betrachten ist, sprich $T_{DS}(0,k)$, entspricht dem Fall $d = n+1$ im Algorithmus. Daher ist $T_{DS}(0,k) = 1$. Durch Auflösung dieser Rekursionsformel erhält man:

$$T_{DS}(n,k) = 2kc_k \sum_{i=0}^{n-1} k^i + k^n = 2kc_k \frac{k^n - 1}{k-1} + k^n \tag{5.5}$$

Man beweist diesen Sachverhalt leicht durch Induktion über n. Damit erhalten wir folgende obere Schranke für die Zeitkomplexität von Dimension Scanning.

$$T_{DS}(n,k) = 2kc_k \frac{k^n - 1}{k-1} + k^n = c_k O(k^n) = O(k^{n+1}) \tag{5.6}$$

Wir erhalten also bereits eine wesentlich bessere Abschätzung als für den Continuous Subtraction Algorithmus in Abschnitt 4.4.2. Man überlegt sich außerdem, wie auch in Abschnitt 4.4 schon, leicht, daß dieser worst-case tatsächlich niemals eintritt; er kann zwar in *einer Dimension* eintreten, dies schließt dann aber die worst-case Position in den anderen Dimensionen aus: In der worst-case Situation aus Abbildung 5.9 (und auch in allen anderen worst-case Situationen) wird der nächsten Dimension jeweils nur *ein einziger* Evidenzquader als Grundmenge übergeben! Es ist eine offene Frage, wie die Laufzeit hier enger abgeschätzt werden kann. Man beachte allerdings, daß die (Bitvektor) Operationen an einem Haltepunkt bei kleinem k in nahezu konstanter Zeit ausgeführt werden können. Wir erhalten dann eine asymptotische Laufzeit von $O(k^n)$.

5.5.2 Preprocessing

Wie bereits erwähnt, wird die Zeit, die für das Preprocessing benötigt wird, asymptotisch von der Laufzeit des eigentlichen Algorithmus dominiert. Man kann sich das wie folgt klar machen.

Zunächst müssen n Listen sortiert werden, nämlich die Listen der Intervallendpunkte. Diese Listen müssen unter Umständen überhaupt erst erzeugt werden; dies kann offenbar in Zeit $O(nk)$ geschehen: Gehe einfach über alle Dimensionen und alle Quader und sammle die entsprechenden Punkte ein. Anschließend kann man diese Listen mit einem beliebigen optimalen Sortierverfahren jeweils in Zeit $O(k \log k)$ sortieren; jede Liste muß einzeln sortiert werden, also haben wir hier insgesamt einen Aufwand von $O(nk + nk \log k) = O(nk \log k)$.

Um nun die IN/OUT Information zu erstellen, kann man wie folgt vorgehen: Gehe über jede Liste und trage für aufeinanderfolgende gleiche Punkte jeweils die Nummer des Evidenzquaders in der entsprechenden Menge ein. Das Eintragen in eine Menge entspricht dem Setzen eines Bits in einem Bitvektor (siehe den vorangehenden Abschnitt 5.5.1) und kann in konstanter Zeit durchgeführt werden. Damit erhält man hier insgesamt einen Aufwand in Höhe von $O(nk)$.

Das Erstellen der ACT Information ist der aufwendigste Schritt im Preprocessing: In jeder Dimension sind an höchstens $2k + 1$ Punkten konstant viele Mengenoperationen durchzuführen. Jede Mengenoperation benötigt Zeit $O(k)$, was für diesen Arbeitsvorgang zu einer Gesamtlaufzeit von $O(nk^2)$ führt.

Die Zeit für das gesamte Preprocessing wird somit von der Zeit für das Erstellen der ACT Information dominiert, wir erhalten eine Gesamtabschätzung von $O(nk^2)$. Dies ist, außer im (Ausnahme-) Fall $k = 1$, asymptotisch weniger als die Laufzeit für den eigentlichen Algorithmus, sprich $O(k^{n+1})$.

Kapitel 6

Empirischer Vergleich

Wir kommen nun zu einer empirischen Analyse des Laufzeitverhaltens von Continuous Subtraction und Dimension Scanning auf verschiedenartigen QCOVER Instanzen. Überraschenderweise stellt sich heraus, daß, trotz seiner schlechteren worst-case Laufzeitabschätzung (siehe Abschnitte 4.4 und 5.5), der Continuous Subtraction Algorithmus in den meisten Fällen das praktikablere Verfahren ist.

6.1 Implementierung

Die Implementierung der besprochenen Algorithmen wurde in C vorgenommen. Der Code gliedert sich im wesentlichen in die drei Teile *Testumgebung*, *Continuous Subtraction* und *Dimension Scanning*.

6.1.1 Die Testumgebung

Um die Algorithmen systematisch testen zu können, wurde zunächst eine *Testumgebung* geschaffen. Die Aufgabe dieser Umgebung ist es, QCOVER Instanzen zufällig zu erzeugen und den Algorithmen zur Bearbeitung zu übergeben. Folgende Parameter müssen beim Aufruf der Umgebung eingestellt werden:

1. t: Die Gesamtzahl der Testläufe, d.h., der erzeugten QCOVER Instanzen.
2. n: Die Anzahl der Dimensionen aller Quader.
3. k: Die Anzahl der Evidenzquader in jeder Instanz.
4. u: Die *Obergrenze* (upper limit) für die Erzeugung von Zufallszahlen; u muß als natürliche Zahl gewählt werden.

Die Bedeutung der ersten drei Parameter sollte klar sein: Beim Aufruf der Testumgebung werden nacheinander t QCOVER Instanzen erzeugt; jede dieser Instanzen ist n-dimensional (d.h., alle Quader der Instanz sind n-dimensional) und hat genau k Evidenzquader. Jeweils nach Erzeugung einer Instanz werden die beiden Algorithmen aufgerufen; nach Abschluß der Berechnungen wird eine neue Instanz erzeugt, solange, bis t Testläufe durchgeführt wurden.

Etwas komplexer ist die Rolle des letzten Parameters u: er bestimmt, *aus welcher Grundmenge* die Endpunkte der Intervalle beim zufälligen Erzeugen einer Instanz gewählt werden. Im bisherigen Verlauf der Arbeit wurde diese Grundmenge nicht weiter beachtet; aus theoretischer Sichtweise spielt sie auch keine große Rolle.[1] Wir werden aber sehen, daß die Grundmenge der Intervallendpunkte für die Laufzeit der Algorithmen von Bedeutung ist, zumindest beim Dimension Scanning.

Bisher waren die Intervallendpunkte immer einfach als Elemente von $\mathbb{Q}$ definiert, insbesondere enthielt die Grundmenge also *unendlich* viele Elemente. In einer realen Anwendung kann man davon ausgehen, daß die Intervallendpunkte aus einer *endlichen* Menge ausgewählt werden, und sei es nur deswegen, weil intern eben nur endlich viele rationale Zahlen dargestellt werden können. Es sind aber auch Anwendungen vorstellbar, siehe dazu die Abschnitte 7.1 und 7.2, in denen in jeder Dimension nur einige wenige Punkte für die Intervallgrenzen zur Verfügung stehen.

Um verschieden große Grundmengen modellieren zu können, wurde deshalb die Menge $\{0, \ldots, u-1\}$ für die Intervallendpunkte zugrunde gelegt. Offenbar ist die Größe dieser Menge über den Parameter u frei wählbar. Ein beliebiges Intervall wird dann wie folgt erzeugt.

- Wähle die untere Grenze L gleichverteilt aus der Grundmenge $\{0, \ldots, u-1\}$.
- Wähle die obere Grenze H gleichverteilt aus der Menge $\{L, \ldots, u-1\}$, also aus allen Elementen der Grundmenge, die größer oder gleich der unteren Grenze sind.

6.1.2 Continuous Subtraction

Zur Implementierung des Continuous Subtraction Verfahrens wurde der Pseudocode aus Kapitel 4 direkt in C übertragen. Die einzigen nicht-elementaren Operationen sind das Hinzufügen und Wegnehmen eines Elementes aus der Menge der aktuellen Anfragequader. Diese Operationen wurden mit Hilfe einer doppelt verketteten linearen Liste implementiert. Da keine Suche erforderlich ist, erhält man so für jede solche Operation konstante Laufzeit.

Zusätzlich ist ein einfaches *Preprocessing* implementiert, das vor Beginn des eigentlichen Prozesses

1. Evidenzquader, deren Schnitt mit dem Anfragequader leer ist, entfernt;
2. feststellt, ob der Anfragequader bereits ganz in einem Evidenzquader enthalten ist.

Dazu wird lediglich eine doppelt geschachtelte Schleife über alle Evidenzquader und alle Dimensionen ausgeführt.

Diese Vorverarbeitung kann *wahlweise* zugeschaltet werden, d.h., bei der Parameterübergabe an die Testumgebung wird festgelegt, ob die Algorithmen davon Gebrauch machen dürfen oder nicht.

[1] Das ist nicht ganz richtig: Der Beweis aus Abschnitt 2.2 benutzt eine Grundmenge mit drei Punkten. Tatsächlich ist das Problem für Grundmengen $\mathcal{U}$ mit weniger als drei Punkten nicht mehr NP-hart: Man überlegt sich leicht, daß der Fall $|\mathcal{U}| = 2$ in Zeit $O(n * k)$ gelöst werden kann, der Fall $|\mathcal{U}| = 1$ ist trivial (Es kann nur ein einziger Quader gebildet werden, also wird Q genau dann überdeckt, wenn $\mathcal{H}$ nicht leer ist).

6.1.3 Dimension Scanning

Auch das Dimension Scanning kann anhand der Algorithmen aus Kapitel 5 direkt implementiert werden. Wiederum sind die einzigen nicht-elementaren Anweisungen Mengenoperationen; allerdings müssen diesmal Mengen geschnitten und vereinigt werden. Die Größe der zu betrachtenden Mengen ist statisch: wie in Abschnitt 5.5 bereits bemerkt, enthält jede Menge immer höchstens k Elemente. Die Mengenoperationen wurden deshalb mit Hilfe von Bitvektoren der Länge k implementiert; zum Vereinigen von zwei Mengen verknüpft man etwa die entsprechenden Vektoren mit OR.

Zur Ausführung von Dimension Scanning ist, wie in Kapitel 5 beschrieben, ein gewisses Preprocessing zwingend erforderlich: Bevor der Algorithmus mit dem Scannen beginnen kann, müssen erst an jedem Haltepunkt M_i^d die benötigten Mengen ACT_i^d der potentiell aktiven Evidenzquader vorliegen. Dazu werden, genau wie in Abschnitt 5.5.2 beschrieben, zunächst die Listen aller Intervallendpunkte sortiert, anschließend die Scan-Listen erstellt und an jedem Punkt dieser Scan-Listen ein Bitvektor erzeugt, in dem genau die Indizes der potentiell aktiven Quader auf 1 gesetzt sind.

Zusätzlich zu diesem zwingend erforderlichen Preprocessing kann, genau wie bei Continuous Subtraction, wahlweise das Preprocessing aus dem vorangehenden Abschnitt zugeschaltet werden. Dieser Prozeß wird dann ausgeführt noch bevor die Listen aller Intervallendpunkte erstellt werden: Evidenzquader, deren Schnitt mit dem Anfragequader leer ist, werden entfernt, ihre Endpunkte tauchen in den Listen gar nicht erst auf. Ist der Anfragequader vollständig in einem Evidenzquader enthalten, so gibt schon die Vorverarbeitung die korrekte Antwort $\top$ zurück.

6.2 Parameterabhängigkeiten und Hypothesen

Vor Beginn der Experimente ist es sinnvoll, einige Überlegungen anzustellen, etwa zur Wahl der Parameter und zum wahrscheinlichen Verhalten der Algorithmen. Wir tun dies in diesem Abschnitt und gelangen zu einer Sammlung von *Hypothesen*, die später empirisch überprüft werden können.

6.2.1 Positive und Negative Instanzen

Eine zu erwartende Folge aus dem Verhältnis der Parameter k, n und u zueinander ist die unterschiedliche Verteilung der Testläufe auf positive und negative Instanzen. Die Unterscheidung zwischen einer positiven und einer negativen Instanz ist dabei wie folgt definiert:

Es sei eine QCOVER Instanz gegeben durch $\mathcal{H}$ und Q. Die Instanz heißt *positiv* gdw. $Q \sqsubset \mathcal{H}$ erfüllt ist. Andernfalls heißt die Instanz *negativ*.

Wie man sich leicht klar machen kann, hängt die Anzahl der positiven (bzw. negativen) Instanzen, die man bei den t ausgeführten Testläufen erhält, in folgender Weise von den Parametern k, n und u ab:

- Je mehr Evidenzquader vorhanden sind, je größer also k ist, um so mehr positive Instanzen sind zu erwarten.

- Je höher n ist, um so mehr negative Instanzen sind zu erwarten. Dies liegt daran, daß ein Evidenzquader schon dann mit Q einen leeren Schnitt hat, wenn dies in einer Dimension für die jeweiligen Intervalle der Fall ist. Je mehr Dimensionen man nun hat, um so höher ist also die Wahrscheinlichkeit für jeden Evidenzquader, daß sein Schnitt mit Q leer ist.

- Je größer die Grundmenge der Intervallendpunkte ist, um so größer ist auch die Wahrscheinlichkeit, daß zwei zufällig gewählte Intervalle einen leeren Schnitt haben. Es ist daher zu erwarten, daß mit der Größe der Grundmenge u auch der Anteil der negativen Instanzen zunimmt.

Man kann diesen Sachverhalt auch wie folgt formulieren: Je höher die Dimension, und je größer die Grundmenge, um so mehr Evidenzquader sind nötig, damit eine zufällig gewählte Instanz mit gleicher Wahrscheinlichkeit positiv wie negativ ausfällt. Man kann also die Anzahl k der Evidenzquader, die nötig sind, damit von t Instanzen jeweils im Durchschnitt $t/2$ positiv und $t/2$ negativ ausfallen, als Funktion von n und u auffassen. Wir bezeichnen diese Funktion künftig mit $EQ(n, u)$.

Die Wahrscheinlichkeit, daß ein Evidenzquader nicht-leeren Schnitt mit Q hat, nimmt in n exponentiell ab; diese Wahrscheinlichkeit ist ja gerade das Produkt der Wahrscheinlichkeiten in jeder Dimension: wenn bei festem u mit Wahrscheinlichkeit p_u der Schnitt zweier Intervalle in einer Dimension nicht leer ist, so hat der Evidenzquader mit Wahrscheinlichkeit p_u^n nicht-leeren Schnitt mit Q. Wir kommen so zur ersten Hypothese.

Hypothese 1: Die Anzahl $EQ(n, u)$ der Evidenzquader, die nötig sind, um im Durchschnitt gleich viele positive wie negative Instanzen zu erhalten, nimmt in n exponentiell zu.

Die Auswirkung von u ist etwas schwieriger abzuschätzen; wir begnügen uns hier mit der Vermutung, daß $EQ(n, u)$ auch in u monoton wächst.

Eine weitere plausible Vermutung ist, daß der durchschnittliche Anteil positiver und negativer Instanzen im wesentlichen davon abhängt, in welchem Verhältnis die verwendete Anzahl k der Evidenzquader zum Wert von $EQ(n, u)$ steht. Anders ausgedrückt:

Hypothese 2: Wählt man für k gleiche Bruchteile von $EQ(n, u)$, so ergibt sich, für alle Werte von n und u, im Durchschnitt ein ähnlicher Anteil positiver Instanzen an den Gesamttestläufen.

Etwas genauer formuliert besagt diese Hypothese: Ist $k/EQ(n, u) = k'/EQ(n', u')$, und erzeugt man zufällig zwei Instanzen mit den jeweiligen Parametern, so sind beide Instanzen mit gleicher Wahrscheinlichkeit positiv. Trifft die Hypothese zu, und kennt man für ein n_0 und u_0 den Verlauf des Anteils positiver Instanzen bei wachsendem k_0, so kann man anhand von $EQ(n, u)$ und k den wahrscheinlichen Anteil positiver Instanzen berechnen: Er entspricht dem Wert für n_0 und u_0 an der Stelle $k_0 = k * EQ(n_0, u_0)/EQ(n, u)$.

6.2.2 Continuous Subtraction vers. Dimension Scanning

Ein wesentlicher Unterschied zwischen Continuous Subtraction und Dimension Scanning ist das verschiedene Verhalten bei der Bearbeitung von positiven bzw. negativen QCOVER

Instanzen. Während nämlich Continuous Subtraction auf positiven Instanzen unter Umständen vorzeitig abbrechen kann, weiß Dimension Scanning evtl. auf negativen Instanzen vorzeitig das Ergebnis. Etwas ausführlicher: Wird im Continuous Subtraction Algorithmus der Anfragequader bereits von den ersten Evidenzquadern vollständig überdeckt, so bricht das Verfahren sofort ab und gibt $\top$ zurück. Dem Dimension Scanning Verfahren hingegen bleibt nichts anderes übrig, als *alle* Haltepunkte vollständig abzusuchen, um dann am Schluß festzustellen, daß jeder Punkt überdeckt war.

Liegt andererseits eine *negative* Instanz vor, so kann es sein, daß bereits einer der ersten Haltepunkte fürs Dimension Scanning nicht überdeckt ist. Der Algorithmus bemerkt dies und kann sofort abbrechen. Für Continuous Subtraction wiederum sind negative Instanzen eher ungünstig: Alle Evidenzquader werden der Reihe nach von der Anfragemenge subtrahiert, nur um dann am Ende festzustellen, daß immer noch etwas übrig ist, Q also nicht überdeckt wird.

Wir fassen diese Beobachtungen zur folgenden Hypothese zusammen:

Hypothese 3: Auf positiven Instanzen terminiert Continuous Subtraction schneller, auf negativen Dimension Scanning.

Die Größe der Grundmenge für die Intervallendpunkte, also der Wert von u, ist für die Effizienz des Dimension Scanning von entscheidender Bedeutung: Je weniger Punkte die Grundmenge enthält, um so weniger Haltepunkte kann das Verfahren ja in jeder Dimension haben, nämlich maximal u. Man macht sich leicht klar, daß, wenn m die maximale Anzahl von Haltepunkten in einer Dimension (d.h., in einer Scan-Liste) bezeichnet, Dimension Scanning nie mehr als m^n Haltepunkte betrachtet. Hat man nun viele Evidenzquader, also einen hohen Wert von k, so gibt es bei kleinem u trotzdem nur wenige Haltepunkte in jeder Dimension; erhöht man jedoch u, so ist zu erwarten, daß die Laufzeit des Verfahrens explodiert. Das Continuous Subtraction Verfahren hingegen sollte relativ unempfindlich gegenüber dem Wert von u sein.

Hypothese 4: Bei hohen Werten von k steigt die Laufzeit von Dimension Scanning mit der Größe u der Grundmenge stark an. Die Laufzeit von Continuous Subtraction hingegen bleibt vergleichsweise unverändert.

6.2.3 Preprocessing

Wie beschrieben, kann wahlweise ein Preprocessing zugeschaltet werden, das vor Ausführung des jeweiligen Algorithmus überflüssige Evidenzquader (deren Schnitt mit Q leer ist) entfernt und feststellt, ob Q bereits ganz in einem einzigen Evidenzquader enthalten ist. Als Folge der Entfernung von überflüssigen Evidenzquadern gelangen wir zur folgenden Hypothese.

Hypothese 5: Behält man k und u bei zugeschaltetem Preprocessing bei und erhöht n, so wird die Laufzeit beider Algorithmen geringer.

Das ist eine plausible Vermutung, denn je höher die Dimension ist, um so höher ist die Wahrscheinlichkeit, daß ein Evidenzquader leeren Schnitt mit Q hat; es können also viele Evidenzquader ganz zu Anfang einfach entfernt werden. Bei sehr hohem n ist sogar zu erwarten, daß *alle* Evidenzquader entfernt werden, was zu einer trivialen Instanz führt.

Die zweite Aufgabe des Preprocessing, festzustellen, ob Q ganz in einem Evidenzquader enthalten ist, führt zur folgenden Hypothese.

Hypothese 6: Erhöht man für feste Werte von n und u die Anzahl k der Evidenzquader, so steigt die Laufzeitkurve beider Algorithmen zunächst stark an; bei zugeschaltetem Preprocessing wird die Steigung der Kurven für sehr hohe Werte von k deutlich flacher.

Der erste Teil dieser Hypothese sollte unmittelbar einleuchten: Je mehr Evidenzquader man bei festem n und u hat, um so schwieriger ist die Instanz. Ist das Preprocessing zugeschaltet, so ist aber für sehr hohe Werte von k die Wahrscheinlichkeit hoch, daß einer der Evidenzquader Q ganz enthält und das jeweilige Verfahren gar nicht erst gestartet werden muß. Im *Durchschnitt* sollte deshalb die Laufzeit der Algorithmen für hohe Werte von k mäßiger ansteigen.

6.3 Experimente

Wir führen nun systematisch Experimente durch, um festzustellen, welches der beiden Verfahren das effizientere ist, und um die Gültigkeit der Hypothesen aus dem vorangehenden Abschnitt zu überprüfen. Wir beginnen mit den Vermutungen zu $EQ(n,u)$, d.h., zu der Anzahl Evidenzquader, die nötig sind, um bei gegebenem n und u gleich viele positive wie negative Instanzen zu erhalten.

6.3.1 Die richtige Wahl von k

Um bei gegebenem n und u den Wert von $EQ(n,u)$ zu bestimmen, wurde wie folgt vorgegangen:

- Wähle einen geschätzten Startwert für k;
- Starte die Testumgebung mit n, u und k auf 1000 Testläufen;
- Wurden gleich viele positive wie negative Instanzen erzeugt, so breche ab. Anderenfalls erhöhe oder verringere k, je nachdem ob weniger oder mehr als die Hälfte der Instanzen positiv war; wiederhole den Testlauf.

Abbildung 6.1 zeigt den Verlauf von $EQ(n,100)$ für steigende Dimension n.

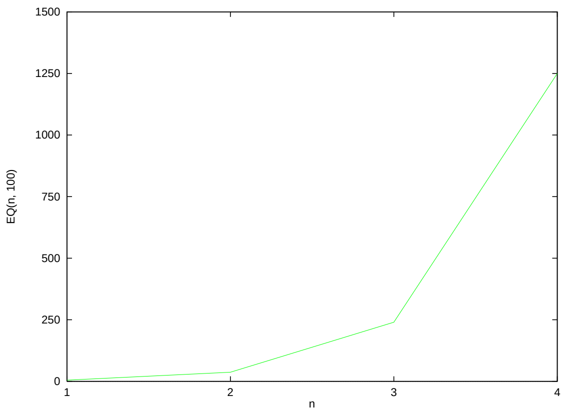

Abbildung 6.1: Ermittelte Werte von $EQ(n,100)$ für wachsendes n.

Der exponentielle Anstieg von $EQ(n, 100)$ in n ist in Abbildung 6.1 deutlich erkennbar; für andere feste Werte von u ergibt sich ein ähnliches Bild. Damit ist Hypothese 1 bestätigt. Abbildung 6.2 zeigt den Kurvenverlauf von $EQ(3, u)$ bei fester Dimension 3 und steigendem u.

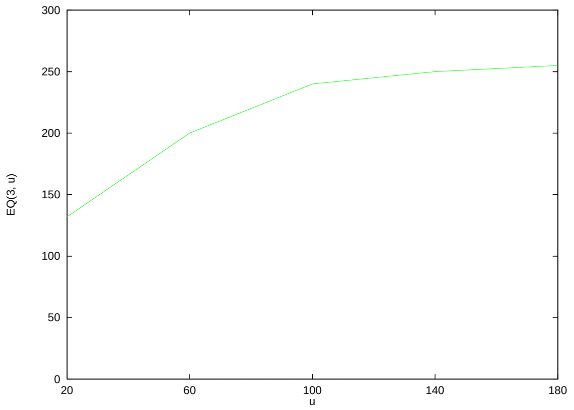

Abbildung 6.2: Ermittelte Werte von $EQ(3, u)$ für wachsendes u.

Offenbar wachsen die Werte von $EQ(3, u)$ monoton in u, womit auch diese Vermutung bestätigt ist.

Wir versuchen nun, Unterstützung für Hypothese 2 zu finden, also aufzuzeigen, daß der Anteil positiver Instanzen konstant bleibt, wenn man für unterschiedliche Werte von n und u jeweils gleiche Bruchteile von $EQ(n, u)$ für k wählt. Abbildung 6.3 zeigt das Ergebnis der Experimente für den festen Wert $u = 100$, wachsendes n und laufendes k von $20\% EQ(n, 100)$ bis $180\% EQ(n, 100)$.

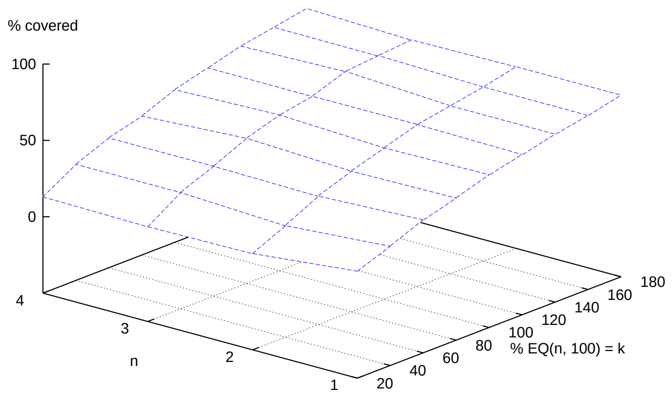

Abbildung 6.3: Der Anteil positiver Instanzen bei wachsender Dimension und gleichen Prozentsätzen von $EQ(n, 100)$.

Man beachte beim Betrachten von Abbildung 6.3, daß nur der *Prozentsatz* von $EQ(n, 100)$ in jeder Dimension gleich ist; da $EQ(n, 100)$ exponentiell in n wächst, tun dies auch die Werte von k, die verwendet werden: In Dimension 2 ist etwa $40\% EQ(2, 100) = 15$, in Dimension 4 ist das bereits $40\% EQ(4, 100) = 500$.

Tatsächlich ist in Abbildung 6.3 recht deutlich zu erkennen, daß der Verlauf des Anteils positiver Instanzen, wenn für k wachsende Prozentsätze von $EQ(n, 100)$ gewählt werden, bei wachsendem n keine relevante Veränderung erfährt. Abbildung 6.4 zeigt die Kurvenverläufe bei *fester Dimension* $n = 2$ und wachsendem u.

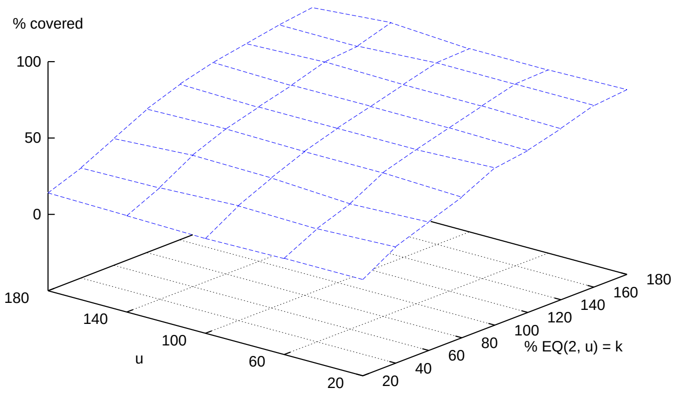

Abbildung 6.4: Der Anteil positiver Instanzen bei wachsender Größe u der Grundmenge und gleichen Prozentsätzen von $EQ(2, u)$.

Wie man in Abbildung 6.4 sieht, ändert sich der Kurvenverlauf auch bei wachsendem u nicht. Damit ist Hypothese 2 bestätigt.

Wir werden uns in den folgenden Experimenten bei der Wahl von k grundsätzlich an $EQ(n, u)$ orientieren. Das hat folgende Gründe:

- Das Verhältnis von positiven zu negativen Instanzen bei den Testläufen ist, wie wir in Abschnitt 6.2.2 gesehen haben, von Bedeutung für das Verhalten der beiden Algorithmen.

- $EQ(n, u)$ ist eine Art Maßzahl für eine bei den gegebenen Parametern vernünftige Wahl von k: bleibt man sehr weit unter diesem Wert, so erhält man eine triviale Instanz, insbesondere können wahrscheinlich bei zugeschaltetem Preprocessing alle Evidenzquader entfernt werden. Wählt man dagegen für k einen sehr hohen Wert, so ist die Wahrscheinlichkeit groß, daß ein Evidenzquader Q bereits ganz enthält, was wiederum bei zugeschaltetem Preprocessing zu einer trivialen Instanz führt.

6.3.2 Laufzeitmessungen

In Tabelle 6.5 sind einige Laufzeiten der beiden Algorithmen bei fest gewähltem $u = 100$ zu sehen. Die Laufzeiten von Continuous Subtraction (C.S.) und Dimension Scanning (D.S.) sind dabei aufgeschlüsselt nach der durchschnittlichen Zeit auf positiven Instanzen (pos), der durchschnittlichen Zeit auf negativen Instanzen (neg) und der durchschnittlichen Zeit, gemittelt über alle Instanzen (ges). Für Dimension Scanning ist zusätzlich der durchschnittliche Aufwand (pre) für das erforderliche Preprocessing angegeben. Alle Zeitangaben sind in *tausendstel* Sekunden; der Wert von k ist absolut und in $\%EQ(n, 100)$ für die jeweilige Dimension n angegeben.

		C.S.			D.S.			
n	k	pos	neg	ges	pos	neg	pre	ges
3	40% (=96)	0.1	1.8	1.4	3.7	1.4	1.9	4.1
3	100% (=240)	0.9	5.3	3.2	20.8	7.2	5.2	19.8
3	160% (=388)	1.2	7.5	3.6	43.7	23.7	10.0	38.7
4	40% (=500)	10.0	40.5	36.4	1875.0	51.4	20.6	363.8
4	100% (=1250)	30.0	96.6	75.2	2084.1	166.3	83.2	901.6
4	160% (=2000)	45.0	123.7	82.3	3112.0	511.6	101.3	1913.4

Abbildung 6.5: Einige Laufzeiten der beiden Algorithmen bei festem $u = 100$. Alle Zeitangaben sind in tausendstel Sekunden, k ist absolut und in $\%EQ(n, 100)$ für die jeweilige Dimension n angegeben.

Mehrere Beobachtungen fallen anhand von Tabelle 6.5 sofort ins Auge:

1. Auf allen Testläufen ist Continuous Subtraction das schnellere Verfahren. Insofern ist also, zumindest bei Verwendung einer Grundmenge mit $u = 100$ Elementen, die Frage nach dem effizienteren Verfahren bereits beantwortet.

2. Continuous Subtraction benötigt für positive Instanzen weniger Zeit als für negative Instanzen, Dimension Scanning verhält sich genau umgekehrt.

3. Das für Dimension Scanning benötigte Preprocessing wird zwar von der Laufzeit des eigentlichen Algorithmus dominiert, trägt aber einen nicht unerheblichen Anteil zur Gesamtlaufzeit bei. In manchen Fällen benötigt das Preprocessing bereits mehr Zeit als der Continuous Subtraction Algorithmus für die Gesamtlösung.

Daß Continuous Subtraction stets schneller terminiert als Dimension Scanning scheint Hypothese 3 zu widerlegen, deren Aussage es ja ist, daß Continuous Subtraction auf positiven Instanzen, Dimension Scanning aber auf negativen Instanzen im Vorteil ist; Beobachtung 2 hingegen unterstützt diese Hypothese, zumindest in dem Sinn, welche Instanzen für die jeweiligen Algorithmen günstiger sind.

Eine genauere Erklärung für das Laufzeitverhalten der Algorithmen erhält man, wenn man nicht direkt die Laufzeiten mißt, sondern die *Anzahl der betrachteten Objekte* zählt. Im Fall von Dimension Scanning sind das alle im Verlauf der Rekursion besuchten Haltepunkte; im Fall von Continuous Subtraction sind das alle betrachteten Schnitte von Evidenzquadern und Anfragequadern, mit der Terminologie aus Abschnitt 4.4 also gerade $\sum_{i=0}^{k-1} m_i$, die Aufsummierung des Betrags der Anfragemenge über die Anzahl der Evidenzquader. In Tabelle 6.6 ist die durchschnittliche Anzahl der betrachteten Objekte für die beiden Algorithmen bei $n = 3$ aufgelistet.

Wie die gemessenen Werte in Tabelle 6.6 deutlich demonstrieren, verhält sich die Größe des Suchraums der beiden Verfahren tatsächlich wie in Hypothese 3 vermutet: Auf positiven Instanzen betrachtet Dimension Scanning erheblich mehr Objekte als Continuous Subtraction, auf negativen Instanzen jedoch verhält es sich genau umgekehrt.

Daß Continuous Subtraction trotzdem auch auf negativen Instanzen schneller terminiert als Dimension Scanning hat folgende Erklärung: Wie wir in Abschnitt 5.5 gesehen haben, benötigt Dimension Scanning an jedem Haltepunkt Zeit $O(k)$; aus Abschnitt 4.4 wissen

	C.S.		D.S.	
k	pos	neg	pos	neg
40% (=96)	133	1347	1023	332
100% (=240)	747	5644	4539	1454
160% (=388)	1210	7365	7067	3854

Abbildung 6.6: Die durchschnittliche Anzahl der betrachteten Objekte für die beiden Algorithmen auf positiven und negativen Instanzen. Die Parameter $u = 100$ und $n = 3$ sind fest gewählt, k ist absolut und in $\%EQ(3, 100)$ angegeben.

wir, daß Continuous Subtraction für jeden betrachteten Anfragequader Zeit $O(n)$ benötigt. Da die Werte von k erheblich größer sind als die Werte von n, hat Dimension Scanning trotz geringerem Suchraum eine höhere Laufzeit. Prinzipiell ist natürlich auch vorstellbar, daß n größer als k ist, daß man also mehr Dimensionen als Evidenzquader hat; dies scheint aber für eine Anwendung unrealistisch zu sein. Außerdem erhält man in so einem Fall aufgrund des exponentiellen Wachstums von $EQ(n, u)$ in n fast ausschließlich negative Instanzen.

Eher realitätsnah ist wohl eine Grundmenge mit weniger als 100 Elementen; in Abbildung 6.7 sind die Laufzeitkurven der beiden Algorithmen für festes $n = 3$ und $k = 240$ sowie wachsende Größe u der Grundmenge zu sehen.

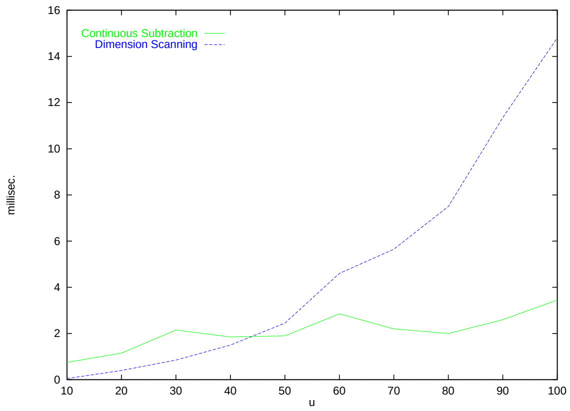

Abbildung 6.7: Laufzeitkurven für festes $n = 3$ und $k = 240$ bei wachsendem u.

Wie in Abbildung 6.7 sehr schön zu erkennen ist, steigt die Laufzeit von Dimension Scanning mit der Größe der Grundmenge stark an, während die Laufzeit von Continuous Subtraction vergleichsweise unverändert bleibt. Damit ist Hypothese 4 bestätigt. Wie man auch sehen kann, zeigt Dimension Scanning bei kleinen Werten von u sogar ein besseres Laufzeitverhalten als Continuous Subtraction; für sehr kleine Grundmengen kann also auch Dimension Scanning das bessere Verfahren sein.

6.3.3 Preprocessing

Das in Abschnitt 6.1.2 beschriebene Preprocessing kann in Zeit $O(nk)$ durchgeführt werden; danach sind unnötige Evidenzquader entfernt und ein erster Test auf Überdeckung von Q durch einen einzelnen Evidenzquader ist durchgeführt. Es ist zu erwarten, daß dieses Preprocessing den Laufzeiten beider Algorithmen sehr zu gute kommt, in der Praxis ist die Verwendung des Preprocessing also wünschenswert. Wir schließen deshalb die Experimente mit einigen Betrachtungen zu den Auswirkungen dieser Vorverarbeitung ab. Zunächst überzeugen wir uns anhand von Tabelle 6.8 davon, daß das Preprocessing tatsächlich große Einsparungen bei der Laufzeit bringt.

		C.S.		D.S.			
k		ges		pre		ges	
40%	(=96)	0.9	(1.4)	0.4	(1.9)	0.7	(4.1)
100%	(=240)	2.1	(3.2)	1.6	(5.2)	7.3	(19.8)
160%	(=388)	2.2	(3.6)	3.2	(10.0)	19.2	(38.7)

Abbildung 6.8: Laufzeiten der beiden Algorithmen (in tausendstel Sekunden) bei zugeschaltetem Preprocessing, mit fest gewählten Parametern $u = 100$ und $n = 3$; k ist absolut und in $\%EQ(3, 100)$ angegeben. Die Laufzeiten *ohne* Preprocessing stehen in Klammern.

Man beachte bei Tabelle 6.8, daß es für Dimension Scanning *zwei* Arten von Preprocessing gibt: Das zwingend erforderliche (Die Erstellung der Scan-Listen), sowie das in diesen Versuchen zugeschaltete. Es zeigt sich, daß auch die Zeit für das zwingend erforderliche Preprocessing durch Zuschalten des fakultativen Preprocessing deutlich geringer wird.

In Abbildung 6.9 sind die Laufzeitkurven zu sehen, die sich ergeben, wenn man bei zugeschaltetem Preprocessing k und u konstant hält und die Dimension erhöht.

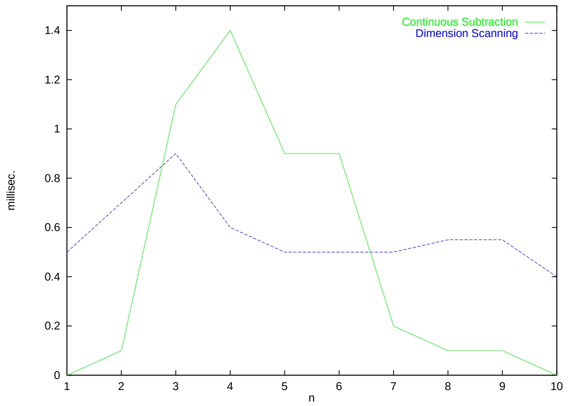

Abbildung 6.9: Laufzeitkurven für $k = 100$ und $u = 100$ bei wachsendem n.

Wie gemäß Hypothese 5 erwartet, steigen die Laufzeitkurven in Abbildung 6.9 zunächst an, um dann mit wachsendem n wieder stark abzufallen, weil fast alle Evidenzquader vom Preprocessing entfernt werden. Die Laufzeit des Dimension Scanning wird auf dieser Testreihe fast ausschließlich im Preprocessing verbraucht; insbesondere sind auf den höheren Dimensionen alle Instanzen negativ, der Algorithmus bricht fast sofort ab, nachdem die Vorverarbeitung abgeschlossen ist.

Abbildung 6.10 zeigt die Laufzeitkurven bei steigendem k auf drei Dimensionen mit $u = 100$ und zugeschaltetem Preprocessing.

Wie in Abbildung 6.10 gut zu erkennen ist, steigt die Laufzeit beider Algorithmen zunächst mit k stark an und erreicht dann, genau wie in Hypothese 6 vorausgesehen, irgendwann einen Punkt, an dem die Steigung deutlich flacher wird. Dies erklärt sich durch

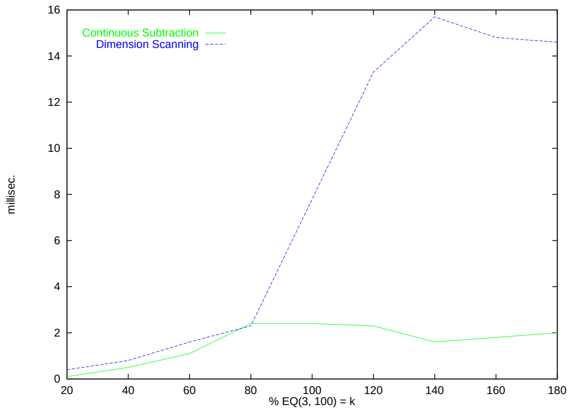

Abbildung 6.10: Laufzeitkurven für steigendes k bei drei Dimensionen und $u = 100$ mit zugeschaltetem Preprocessing.

den Umstand, daß mit steigendem k auch die Chance steigt, daß einer der Evidenzquader Q bereits ganz überdeckt.

6.4 Schlußfolgerungen

Aufgrund der empirischen Untersuchung kommen wir zu dem Schluß, daß Continuous Subtraction in den meisten Fällen das vorzuziehende Verfahren ist. Die Gründe dafür sind, noch einmal zusammengefaßt:

1. Die beiden Algorithmen haben zwar für gewöhnlich einen ähnlich großen Suchraum, was die betrachteten Haltepunkte bzw. Anfragequader angeht; zur Betrachtung eines Haltepunktes braucht Dimension Scanning aber Zeit $O(k)$, zur Betrachtung eines Anfragequaders benötigt Continuous Subtraction Zeit $O(n)$. Im Normalfall dürfte k erheblich größer als n sein, was (trotz Bitvektor-Implementierung) zu wesentlich längeren Laufzeiten für Dimension Scanning führt.

2. Das zwingend erforderliche Preprocessing für Dimension Scanning wird zwar asymptotisch von der Laufzeit des eigentlichen Verfahrens dominiert, nimmt aber in der Praxis erhebliche Zeitspannen in Anspruch; insbesondere benötigt die Vorverarbeitung der Instanz in manchen Fällen bereits mehr Zeit als der Continuous Subtraction Algorithmus für die Gesamtlösung.

3. Ist die Grundmenge, aus der die Intervallendpunkte ausgewählt werden, sehr groß, so fallen in den Scan-Listen für Dimension Scanning nur wenige Punkte zusammen, was zu langen Listen führt. Ist n die Anzahl der Dimensionen und m die Länge der größten Scan-Liste (also das Maximum der Längen der Scan-Listen über alle Dimensionen), so betrachtet der Algorithmus $O(m^n)$ Haltepunkte. Deshalb steigt die Laufzeit des Verfahrens mit der Größe der Grundmenge stark an.

In Anbetracht dieser drei Feststellungen ist zu vermuten, daß der Einsatz des Dimension Scanning nur bei solchen Anwendungen Sinn macht, für die die folgenden Bedingungen erfüllt sind:

1. Es gibt nicht wesentlich mehr Evidenzquader als Dimensionen.

2. Das wiederholte Preprocessing für jede einzelne Instanz kann umgangen werden.

3. Die Grundmenge, aus der die Intervallendpunkte stammen, ist sehr klein.

Im nachfolgenden Ausblick (Kapitel 7) werden die beiden einzigen dem Autor zur Zeit bekannten Anwendungen der QCOVER Algorithmen besprochen; die eine davon erfüllt Bedingung zwei und drei, die andere erfüllt Bedingung eins und drei. Es gibt also tatsächlich Anwendungen, in denen der Einsatz des Dimension Scanning Verfahrens sinnvoll sein kann.

Ein interessanter Aspekt der QCOVER Problematik ist das Verhältnis der Parameter k, n und u zueinander: kann man in einer Anwendung die möglichen Intervallendpunkte auf eine endliche Menge einschränken (was etwa einem festen Wert für u in der Testumgebung entspricht), so sind Rückschlüsse auf den wahrscheinlichen Ausgang der Untersuchungen möglich. Ist man mit k etwa weit oberhalb von $EQ(n, u)$ (d.h., von dem aufgrund der Endpunktverteilung berechneten $EQ(n, u)$), so ist die Instanz mit großer Wahrscheinlichkeit positiv, umgekehrt gibt es weit unterhalb von $EQ(n, u)$ fast nur negative Instanzen. Je nach Anwendung und Abstand von $EQ(n, u)$ könnte man dann sogar direkt, ohne weitere Berechnung, als einen *heuristischen* Wert das erwartete Ergebnis zurückgeben.

Falls eine Anwendung vorliegt, in der auch der Einsatz von Dimension Scanning sinnvoll erscheint, so kann man den Abstand von k zu $EQ(n, u)$ in jedem Fall als Heuristik verwenden, um einen der beiden Algorithmen auszusuchen: erwartet man eine positive Instanz, so ist Continuous Subtraction vorzuziehen, andernfalls Dimension Scanning. Die optimale Strategie wäre natürlich, beide Algorithmen auf parallel arbeitenden Prozessoren gleichzeitig auszuführen.

Kapitel 7

Ausblick

In dieser Arbeit wurde ein - dem Ermessen des Autors nach - neues algorithmisches Problem behandelt, das seinen Ursprung in der KI-Handlungsplanung hat [Koehler, 1999]. Die Untersuchung der theoretischen Komplexität des Problems ergab, daß es in seiner ursprünglichen Formulierung co-NP-vollständig ist, was die Suche nach guten Lösungsverfahren zu einer interessanten und schwierigen Aufgabe macht. Anhand der vorhandenen Literatur aus dem Bereich der algorithmischen Geometrie wurden zwei gängige algorithmische Prinzipien ausgewählt und auf ihrer Grundlage zwei Lösungsverfahren für das Problem entwickelt. Die bei der Anwendung der Prinzipien auftretenden Probleme und ihre Lösungen wurden ausführlich diskutiert; die Korrektheit der entstandenen Algorithmen Continuous Subtraction und Dimension Scanning wurde bewiesen und ihr worst-case Laufzeitverhalten wurde abgeschätzt. Anschließend wurden sie anhand der empirischen Untersuchung einer möglichst effizient gehaltenen Implementierung bezüglich ihrer durchschnittlichen Laufzeit verglichen. Es ergab sich, daß Dimension Scanning nur auf Anwendungen mit bestimmten Eigenschaften mit Continuous Subtraction mithalten kann; in diesem Fall arbeitet Continuous Subtraction auf positiven, Dimension Scanning auf negativen Instanzen schneller.

Abschließend wollen wir zwei Problemstellungen besprechen, auf die die entwickelte Maschinerie angewendet werden kann; das ist zum einen die Handlungsplanung unter Ressourcenconstraints, die ja die QCOVER Fragestellung überhaupt erst motivierte, zum anderen ist es das Erfüllbarkeitsproblem für aussagenlogische Formeln, SAT.

7.1 Eine Anwendung in der Handlungsplanung

Wie bereits in Abschnitt 1.1 beschrieben, tritt das Hyperquader-Überdeckungsproblem im Planungssystem IPP mit Ressourcen auf, wenn festgestellt werden muß, ob alle möglichen Zuweisungen an die Ressourcen-Variablen bereits in den abgespeicherten (als unlösbar nachgewiesenen) Hyperquadern enthalten sind. Dabei werden *dynamisch* QCOVER Instanzen erzeugt. Um genau zu sein, verläuft der Prozeß wie folgt:

- Anfangs ist $\mathcal{H} = \emptyset$.
- Jedesmal, wenn für die aktuellen Ressourcen-constraints kein Plan gefunden werden konnte, wird der entsprechende Hyperquader (das Kreuzprodukt der Constraint-Intervalle) zu $\mathcal{H}$ hinzugenommen.

- Für jede neue Kombination von Constraints wird zunächst geprüft, ob der entsprechende Quader Q bereits von den Quadern in $\mathcal{H}$ überdeckt wird. Genau dann ist bereits die Unlösbarkeit von Q nachgewiesen.

Die dynamische Entwicklung von $\mathcal{H}$ hat im wesentlichen zwei Auswirkungen auf die Anwendbarkeit von Dimension Scanning:

- Die Länge der zu verwendenden Bitvektoren bei den Mengenoperationen ist nicht von Anfang an festgelegt, sie wächst im fortschreitenden Prozeß. Man kann dieses Problem etwa lösen, indem man eine Höchstlänge festlegt und jeweils über die aktuelle Länge Buch führt. Eine andere Möglichkeit besteht darin, die Bitvektoren als lineare Listen zu implementieren, bei denen jedes Element einen Teilvektor und einen Zeiger auf die Fortsetzung enthält; bei Bedarf kann dann einfach ein weiteres Teilstück angehängt werden.

- Die Scan-Listen in jeder Dimension, d.h., für jede Ressource, müssen nicht für jede Instanz neu berechnet werden; die Bitvektoren an jedem Haltepunkt müssen nur angepaßt werden, wenn ein neuer Evidenzquader hinzukommt. Für jeden Anfragequader muß dann nur noch der jeweils relevante Teil der Scan-Listen bestimmt werden. Den größten Teil des Aufwands für das zum Dimension Scanning zwingend erforderliche Preprocessing kann man sich also sparen.

Neben den Einsparungen beim Preprocessing spricht noch ein weiterer Grund *für* die Anwendung von Dimension Scanning: Die Grundmenge der Haltepunkte in jeder Dimension entspricht genau den Werten, die eine Ressource annehmen kann; diese Werte ergeben sich lediglich aus einem Startwert und der Anwendung verschiedener Planungsoperatoren, wobei ein Operator jeweils einen konstanten Wert zur Ressource addiert oder von ihr subtrahiert. Da es im allgemeinen nur wenige Operatoren gibt, die eine bestimmte Ressource beeinflussen, ist also zu erwarten, daß die Grundmengen für die Intervallendpunkte relativ klein ausfallen.

Gegen die Anwendung von Dimension Scanning spricht, daß es wahrscheinlich erheblich mehr Evidenzquader als Dimensionen gibt: Während die Formalisierung eines interessierenden Weltausschnitts für gewöhnlich mit einigen wenigen Ressourcen, also Dimensionen, geschehen kann, werden beim Planen nicht selten Millionen von Lösungsmöglichkeiten, und damit auch Ressourcen-Constraints, durchprobiert; $k = |\mathcal{H}|$ kann also im Vergleich zu n sehr groß werden.

7.2 Eine Anwendung auf SAT

Betrachtet man den Beweis der NP-Härte von QNEGCOVER aus Abschnitt 2.2, so stellt man fest, daß die Reduktion von 3SAT auf QNEGCOVER sehr effektiv ist. Hat man allgemein eine aussagenlogische Formel ϕ in konjunktiver Normalform mit n Variablen und k Klauseln, so kann eine entsprechende QNEGCOVER Instanz in Zeit $O(kn)$ gebildet werden: Die Definition 2.2 der Reduktionsfunktion f kann leicht so erweitert werden, daß zu Klauseln C beliebiger Länge ($\leq n$) in Zeit $O(n)$ der entsprechende Klauselquader gebildet wird. Die so entstehende QNEGCOVER (bzw. QCOVER) Instanz ist negativ (d.h., der Anfragequader $[-1, 1]^n$ wird *nicht* überdeckt) genau dann, wenn ϕ erfüllbar ist.

Der Gedanke liegt also nahe, zu einer gegebenen aussagenlogischen Formel in konjunktiver Normalform einfach die Reduktion auf QNEGCOVER zu berechnen und dann die beiden Algorithmen Continuous Subtraction und Dimension Scanning laufen zu lassen. Durch die besondere Struktur der so entstandenen QCOVER Instanzen ist auch hier, wie in Abschnitt 7.1, der Einsatz von Dimension Scanning nicht von vornherein auszuschließen:

- Man macht sich leicht klar, daß die Scan-Liste in jeder Dimension höchstens zwei Haltepunkte enthält, nämlich $-\frac{1}{2}$ und $\frac{1}{2}$. Insbesondere erhalten wir damit bereits eine obere Schranke von 2^n betrachteten Haltepunkten für Dimension Scanning, was zu einer Höchstlaufzeit von $O(2^n k)$ führt.

- Continuous Subtraction benötigt zwar für jeden Anfragequader nur Zeit $O(n)$, während Dimension Scanning an jedem Haltepunkt Zeit $O(k)$ braucht; aufgrund der Implementierung mit Bitvektoren kann aber (bei einer Standardwortgröße von 32 Bit) k bis zu 32 mal größer sein als n, ohne daß für Haltepunkte tatsächlich mehr Zeit gebraucht wird als für Anfragequader. Wenn nun für eine aussagenlogische Formel k mehr als 32 mal so groß wie n ist, so enthält die Formel wesentlich mehr Klauseln als Variablen; die Formel ist *over-constrained*. Man kann dann wohl davon ausgehen, daß die Formel nicht erfüllbar ist. In [Selman et al., 1992a] etwa ist zu sehen, daß die Erfüllbarkeitswahrscheinlichkeit für zufällig gewählte 3SAT Instanzen bereits bei 7 mal mehr Klauseln als Variablen nahezu bei 0 liegt.[1]

Interessant ist es auch, von der Reduktion auf QNEGCOVER zu abstrahieren und direkt zu betrachten, was für Algorithmen sich ergeben, wenn man Continuous Subtraction und Dimension Scanning auf SAT *spezialisiert*. Dabei entspricht dann ein Quader jeweils immer genau einer *Konjunktion* von Literalen (Für alle Kanten $[-1, 0]$ die entsprechende Variable negiert, für alle Kanten $[0, 1]$ die entsprechende Variable als positives Vorkommen); eine *Menge* von Quadern entspricht also einer *Disjunktion von Konjunktionen.*

- Das Continuous Subtraction Verfahren, angewandt auf SAT, kann als eine Art *Hypothesenraumsuche* aufgefaßt werden: Die aktuelle Anfragemenge entspricht immer genau der Menge der bis jetzt noch erfüllenden Variablenbelegungen in disjunktiver Normalform. Anfangs ist diese Menge der ganze Raum, also die Disjunktion, die nur das eine Element $\top$ enthält; mit jeder Klausel wird dann der entsprechende Raumausschnitt abgezogen, was der AND Operation mit einer *negierten Konjunktion* entspricht. Die Operation wird dabei mit jedem Element der Disjunktion einzeln durchgeführt. Falls sich ein Widerspruch ergibt, wird das aktuelle Disjunkt gestrichen; das entspricht dem Fall, daß der aktuelle Anfragequader ganz im Evidenzquader enthalten ist. So erhält man zum Schluß wieder eine disjunktive Normalform, in der alle Variablenbelegungen, die von der Klausel ausgeschlossen wurden, nicht mehr erfüllend sind.

- Ein originelles Verfahren ergibt sich, wenn man die Spezialisierung von Dimension Scanning auf SAT durchdenkt: Die Menge der *potentiell aktiven* Evidenzquader an

[1] Hier zeigt sich eine interessante Parallele zu $EQ(n, u)$: Auch bei Formeln in konjunktiver Normalform, die zufällig nach einem bestimmten Schema gebildet werden, gibt es einen festen Wert, bei dem die Erfüllbarkeitswahrscheinlichkeit nahe bei $\frac{1}{2}$ liegt. Weit oberhalb und weit unterhalb dieses Wertes erhält man fast nur triviale Instanzen.

den zwei Haltepunkten je Variable entspricht genau der Menge der Klauseln, die die Belegung dieser Variable mit dem entsprechenden Wert *potentiell ausschließen* können; die Klauseln, deren Quader etwa in Dimension d den Punkt $\frac{1}{2}$ enthält, sind ja aufgrund der Definition der Reduktionsfunktion f genau diejenigen, die das Literal v_d *nicht* enthalten. Das sind genau die Klauseln, die bei einer Belegung von v_d mit $\top$ unter Umständen nicht erfüllt sein können. Das Dimension Scanning Verfahren besteht nun darin, systematisch immer mehr Variablenbelegungen festzulegen und effektiv durch AND Operationen auf Bitvektoren festzustellen, ob die jeweilige partielle Belegung bereits von keiner Klausel mehr ausgeschlossen wird, mit anderen Worten: ob bereits jede Klausel erfüllt ist. Aus einer solchen partiellen Belegung kann dann eine Menge von erfüllenden Belegungen extrahiert werden (Jede Fortsetzung dieser Belegung auf die verbliebenen Variablen ist eine erfüllende Belegung), im Gegensatz zu Continuous Subtraction erhält man allerdings nicht *alle* erfüllenden Variablenbelegungen.

Inwiefern diese auf etwas ungewöhnliche Art und Weise entstandenen Lösungsalgorithmen für das Erfüllbarkeitsproblem mit existierenden Verfahren wie etwa der *Davis Putnam Prozedur* aus [Davis and Putnam, 1960] oder *GSAT* aus [Selman et al., 1992b] konkurrieren können, scheint eine lohnende Forschungsaufgabe zu sein.

Literaturverzeichnis

[Blum and Furst, 1997] Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1–2):279–298.

[Culberson and Reckhow, 1994] Culberson, J. and Reckhow, R. (1994). Covering polygons is hard. *Journal of Algorithms*, 2-44.

[Davis and Putnam, 1960] Davis, M. and Putnam, H. (1960). A computing procedure for quantification theory. *J. Assoc. Comput. Mach.*, pages 201 – 215.

[Edelsbrunner, 1987] Edelsbrunner, H. (1987). *Algorithms in Combinatorial Geometry.* Springer, Berlin.

[Fikes and Nilsson, 1971] Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *AI*, 2:189–208.

[Franzblau and Kleitman, 1984] Franzblau, D. and Kleitman, D. (1984). An algorithm for constructing regions with rectangles: Independence and minimum generating sets of collections of intervals. *Proceedings of the 16th Annual ACM Symposium on the Theory of Computation*, pages 167–174.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). *Computers and Intractability - A Guide to the Theory of NP - Completeness.* Freeman.

[Koehler, 1998] Koehler, J. (1998). Planning under resource constraints. In Prade, H., editor, *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 489–493. John Wiley & Sons, Chichester, New York.

[Koehler, 1999] Koehler, J. (1999). IPP - a planning system for ADL and resource-constrained planning problems. Habilitationsschrift, Albert-Ludwigs-Universität.

[Lee and Preparata, 1984] Lee, D. and Preparata, F. (1984). Computational geometry – a survey. *IEEE Transactions on Computers*, pages 1072–1102.

[Lifschitz, 1990] Lifschitz, V. (1990). On the semantics of STRIPS. In *Readings in Planning*, pages 1–9, kauf.

[Masek, 1979] Masek, W. (1979). Some np - complete set covering problems. unpublished manuskript.

[Mehlhorn, 1984] Mehlhorn, K. (1984). *Data structures and algorithms, Vol. 2: Multidimensional searching and computational geometry.* Springer, Berlin.

[Ottmann and Widmayer, 1997] Ottmann, T. and Widmayer, P. (1997). *Algorithmen und Datenstrukturen.* Spektrum.

[Selman et al., 1992a] Selman, B., Levesque, H., and Mitchell, D. (1992a). Hard and easy distributions of sat problems. In *Proc. of AAAI-92*, pages 459 – 465, San Jose, CA.

[Selman et al., 1992b] Selman, B., Levesque, H., and Mitchell, D. (1992b). A new method for solving hard satisfiability problems. In *Proc. of AAAI-92*, pages 459 – 465, San Jose, CA.