Diplomarbeit

Skyline-Anfragen für die multidimensionale Suche
in einer Aktordatenbank

Skyline von Sydney - © Ulla Trampert / PIXELIO

Bearbeiter: Samuel Plentz

Abgabedatum: 25.05.2009

Verantwortliche Professoren: Prof. Dr. Kai-Uwe Sattler
Jun.-Prof. Dr.-Ing. Tom Ströhla

Hochschulbetreuer: Dipl.-Inf. Katja Hose

TECHNISCHE UNIVERSITÄT ILMENAU

Fakultät für Informatik und Automatisierung
Fachgebiet Datenbanken und Informationssysteme

URN: urn:nbn:de:gbv:ilm1-2009200119

Eidesstattliche Erklärung:

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel verfasst habe. Wörtliche und sinngemäße Zitate sind entsprechend gekennzeichnet.

Ilmenau, den 25. Mai 2009 _____

Samuel Plentz

Danksagung

An dieser Stelle möchte ich allen Personen danken, die mich beim Schreiben der Diplomarbeit durch Hinweise, Ratschläge, Lob, Kritik oder auf andere Weise unterstützt haben.

Ich bedanke mich bei Professor Kai-Uwe Sattler für die fachliche Unterstützung und die konstruktive Kritik. Bei meiner Betreuerin Katja Hose bedanke ich mich für die Einführung in die Welt der Skylines und die Änderungsvorschläge an der Diplomarbeit. Weiterhin danke ich Professor Ströhla und Torsten Erbe von der FAKULTÄT FÜR MASCHINENBAU für die gute Zusammenarbeit. Besonderer Dank gilt auch Markus Porfert, durch den zahlreiche sprachliche Korrekturen in die Diplomarbeit eingeflossen sind.

Diese Diplomarbeit widme ich meiner Frau Michaela, die viel Zeit damit verbracht hat, diese Arbeit mit mir zu verbessern. Danke, dass du mir den Rücken freigehalten und auf vielfältige Art deine Liebe bewiesen hast.

Zuletzt will ich meinen Eltern danken. Ich danke meinen leiblichen Eltern für die Ermöglichung des Studiums und alle Liebe, die sie mir in meinem Leben gegeben haben. Ich danke meinem himmlischen Vater für die Hoffnung, die er in mein Leben gegeben hat und für seine bedingungslose Annahme.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Für die Entwicklung mechatronischer Systeme stehen zahlreiche Aktoren mit unterschiedlichsten Eigenschaften zur Auswahl. Aktoren sind technische Systeme zur Erzeugung von zeitlich und räumlich definierten Bewegungen eines Körpers bei Überwindung von Bewegungswiderständen [KEQS+08]. Eigenschaften der Aktoren können beispielsweise Drehzahl, Wirkungsgrad, Preis oder Leistung sein. Die Entwickler können jedoch die Vielzahl der Aktoren nicht überblicken. Daher wurde an der TECHNISCHEN UNIVERSITÄT ILMENAU in der FAKULTÄT FÜR MASCHINENBAU eine Aktordatenbank angelegt. Mit dieser Datenbank soll die derzeitige Auswahl an Aktoren bestmöglich repräsentiert werden. Alle wichtigen Eigenschaften der Aktoren sind dabei in der Datenbank gespeichert.

Bislang gab es für Entwickler keinerlei Möglichkeit, Aktoren in dieser Aktordatenbank zu suchen, da sie nicht öffentlich zugänglich war. Die Entwickler bezogen Informationen zu nützlichen Aktoren über Tabellen in Büchern oder von den Websites der Aktorhersteller. Ansonsten konnten sie nur auf die Aktoren zurückgreifen, die sie bereits kannten.

Im letzten Jahr gab es Bemühungen, die Aktordatenbank im Internet zu veröffentlichen und mit einem Suchalgorithmus auszustatten. Dieser Suchalgorithmus soll im Rahmen dieser Diplomarbeit entstehen. Ein Webfrontend für den Suchalgorithmus wird von Torsten Mess im Rahmen seiner Diplomarbeit erstellt.

Mit dem Suchalgorithmus soll es möglich sein, nach beliebig vielen verschiedenen Eigenschaften der Aktoren zu suchen. Beispielsweise könnte nach

einem Aktor mit einer Leistung von bis zu 10 Kilowatt gesucht sein, der eine Drehzahl im Intervall von 20 bis 50 Umdrehungen pro Sekunde hat. Dabei soll der Wirkungsgrad 75% nicht unterschreiten. Auch wenn keiner der Aktoren alle gesuchten Eigenschaften erfüllt, soll der Suchalgorithmus ein brauchbares Ergebnis liefern. Dabei ist es schwierig, die Relevanz der einzelnen Eigenschaften zu bestimmen, da verschiedene Entwickler unterschiedliche Präferenzen diesbezüglich haben können. Jedoch können herkömmliche Suchalgorithmen nicht die Präferenzen aller Entwickler berücksichtigen. Der in dieser Diplomarbeit zur Anwendung kommende Skyline-Ansatz ist dagegen präferenzunabhängig.

Von dem Suchalgorithmus wird weiterhin ein möglichst breites Spektrum an Ergebnissen erwartet. Mit herkömmlichen Suchalgorithmen würden, in bestimmten Fällen, viele Aktoren mit fast identischen Eigenschaften die Ergebnisliste anführen. Die Ergebnisse anderer Baugruppen würden dabei zwar auch in der Ergebnisliste angezeigt werden, aber vom Entwickler seltener beachtet, da sie nicht am Anfang der Ergebnisliste aufgeführt werden. Beispielsweise könnten einige Aktoren mit sehr ähnlichen Eigenschaften von einem Hersteller in der Datenbank sein. Wenn einer dieser Aktoren zu Beginn der Ergebnisliste aufgeführt wird, gilt das Gleiche wahrscheinlich auch für die anderen Aktoren.

Durch den Skyline-Ansatz wird ein breiteres Spektrum unter den Suchergebnissen erzielt. Um dieses Spektrum weiter zu vergrößern, wurden im Rahmen dieser Diplomarbeit weitere Skyline-Varianten entwickelt.

Aufbau der Diplomarbeit

Im folgenden Kapitel wird die Struktur der Aktordatenbank vorgestellt. Dann wird der Stand der Technik betrachtet und der Skyline-Ansatz erklärt. Dabei wird auf bereits veröffentlichte Varianten und zwei später zur Anwendung kommende Algorithmen des Skyline-Ansatzes eingegangen. Um den Skyline-Ansatz an die Aufgabenstellung anzupassen, werden anschließend drei Skyline-Varianten entwickelt. Dabei wird besonderer Wert auf eine breite Vielfalt unter den zuerst in der Ergebnisliste aufgeführten Aktoren gelegt. Ebenso werden die Algorithmen angepasst, damit sie die neu entwickelten Skyline-Varianten durchführen können. Es folgen weitere Anforderungen an die Algorithmen, bei denen der Schwerpunkt auf Zeiteffizienz, Skalierbarkeit und der Umsetzung der Skyline-Varianten liegt.

Als nächstes wird dokumentiert, wie die Programme, die im Rahmen der Diplomarbeit entstanden, zu verwenden sind. Abschließend werden die Ergebnisse der Laufzeitmessungen dargestellt und ausgewertet. Dazu werden Testläufe mit Datenbanken unterschiedlicher Größe durchgeführt, wobei die Anzahl der Eigenschaften, an die Anfragen gestellt werden, variiert.

Auch wenn in dieser Diplomarbeit nur Bezug auf die Aktordatenbank genommen wird, sind die beschriebenen Skyline-Varianten und Algorithmen allgemein gehalten, damit sie auf beliebige andere Datenbanken angewandt werden können.

2. Die Aktordatenbank

Aufbau der Aktordatenbank

In der Aktordatenbank existieren zwei verschiedene Aktortypen: Die rotatorischen und translatorischen Aktoren. Die Aktoren dieser Aktortypen haben sowohl gemeinsame als auch unterschiedliche Eigenschaften. Deshalb besteht die Aktordatenbank hauptsächlich aus zwei Tabellen. In der einen Tabelle werden die rotatorischen, in der anderen die translatorischen Aktoren gespeichert. Dabei repräsentiert jede Zeile in solch einer Tabelle einen Aktor. Für alle Eigenschaften der Aktoren existiert in den jeweiligen Tabellen eine Spalte. Die Eigenschaften der Aktoren werden in zwei Klassen eingeteilt. Die Eigenschaften, die sich durch Zahlen beschreiben lassen, werden durch Gleitkommazahlen in den numerischen Spalten der Datenbank gespeichert. Die Eigenschaften der anderen Klasse sind beschreibend und werden durch Zeichenketten in den kategorischen Spalten der Datenbank abgelegt. Da die Quellen der Aktordatenbank nicht immer alle Eigenschaften angeben, fehlen bei manchen Aktoren einzelne Eigenschaftseinträge.

index	typ	herst	bew_rast	kraft	p_ab	masse
1	Gleichstrommagnet	EKS	mit Rast	0,033	4,6	0,0375
2	Gleichstrommagnet	EKS	mit Rast	6,12	9,77	0,24
8	Gleichstrommagnet	Kuhse	mit Rast	35	19	1
12	Gleichstrommagnet	Kuhse	mit Rast	340	82	13,6
48	Synchron-Linearmotor	A-drive	mit Rast	144	98,8	0,91
49	Synchron-Linearmotor	A-drive	mit Rast	98	26,1	0,72
62	Tauchspulenmotoren	A-drive	mit Rast	24,34	112,4	0,8648
63	Tauchspulenmotoren	A-drive	mit Rast	64,68	284,2	2,5076
200	Pneumatik-Zylinder	SMC	ohne Rast	211	*NULL*	1,295
201	Pneumatik-Zylinder	SMC	ohne Rast	1056	*NULL*	1,295
202	Pneumatik-Zylinder	SMC	ohne Rast	211	*NULL*	1,414
262	Pneumatik-Zylinder	SMC	mit Rast	907	*NULL*	3,705
263	Pneumatik-Zylinder	SMC	mit Rast	4536	*NULL*	3,705

Abbildung 2.1: Ausgewählte Zeilen und Spalten der translatorischen Tabelle

Zur Identifikation eines Aktors wird eine Indexspalte verwendet. Die Abbildung 2.1 zeigt ausgewählte Spalten von einzelnen Aktoren aus der translatorischen Tabelle der Aktordatenbank. Die Tabellen haben aktuell jeweils etwa 20 Spalten und um die 350 Einträge.

Die Infotabellen

Für beide Tabellen existiert zusätzlich jeweils eine Infotabelle. Die Infotabellen übernehmen den Namen der ursprünglichen Tabelle und haben den Suffix „_info“, zum Beispiel: „rotatorisch_info“. In den Infotabellen sind Informationen zu den einzelnen Spalten der jeweils zugehörigen Tabelle gespeichert. Die Abbildung 2.2 zeigt die zur Abbildung 2.1 gehörige Infotabelle.

index	typ	herst	bew_rast	kraft	p_ab	masse
Beschreibung	Typ	Hersteller	Bewegungsrast	Kraft	Leistung	Masse
Einheit	*NULL*	*NULL*	*NULL*	N	W	kg
Typ	a	a	a	z	z	z

Abbildung 2.2: Ausgewählte Zeilen und Spalten der translatorischen Infotabelle

Die Informationen der Infotabelle sind in drei Zeilen mit den Indizes „Beschreibung“, „Einheit“ und „Typ“ gespeichert. In der Zeile mit dem Index „Typ“ ist für jede Spalte angegeben, ob es sich um eine numerische oder eine kategorische Spalte handelt. Numerische Spalten haben in dieser Zeile den Wert „z“, kategorische Spalten den Wert „a“. Andere Werte sind in dieser Spalte unzulässig. Die Zeilen mit dem Indizes „Beschreibung“ und „Einheit“ werden nur von dem Webfrontend für die Anzeige verwendet und sind im Rahmen dieser Diplomarbeit unbedeutend. In diesen Zeilen werden die Einheit und der vollständige Name der jeweiligen Spalte gespeichert.

Die Suchanforderungen

Bei der Aktorsuche haben die Entwickler Vorstellungen von einer oder mehreren Eigenschaften der Aktoren. Aus diesem Grund soll der Suchalgorithmus der Aktordatenbank auf Spaltenanfragen basieren. Eine Suchanfrage an die Datenbank besteht aus der Kombination von beliebig vielen Anfragen an die einzelnen Spalten der Datenbank. Dabei unterscheiden sich die Anfragemöglichkeiten bei numerischen und kategorischen Spalten.

An numerische Spalten können vier Arten von Anfragen gestellt werden:

- Entspricht der Datensatz in der Spalte einem Referenzwert?
- Ist der Datensatz in der Spalte größer als ein Referenzwert?
- Ist der Datensatz in der Spalte kleiner als ein Referenzwert?
- Liegt der Datensatz in der Spalte in einem Intervall?

An kategorische Spalten können zwei Arten von Anfragen gestellt werden:

- Entspricht der Datensatz in der Spalte einer bestimmten Zeichenkette?
- Unterscheidet der Datensatz sich in der Spalte von einer Zeichenkette?

Einige Beispiele von Suchanfragen an die Aktordatenbank:

```
Suche 1: Der Wert in Spalte 7 entspricht 87,
         der Wert in Spalte 9 ist kleiner als 500,
         der Wert in Spalte 6 liegt zwischen 30 und 70;

Suche 2: Der Wert in Spalte 2 entspricht „Synchron-Linearmotor“,
         der Wert in Spalte 3 unterscheidet sich von „A-drive“;

Suche 3: Der Wert in Spalte 9 ist größer als 70,
         der Wert in Spalte 2 entspricht „Pneumatik-Zylinder“,
         der Wert in Spalte 8 liegt zwischen 175 und 250;
```

3. Stand der Technik

3.1 Standardsuchalgorithmen

Der einfachste Suchalgorithmus für Datenbanken ist die exakte Suche. Dabei ist ein Datensatz nur dann Bestandteil der Ergebnismenge, wenn alle Suchanforderungen vollständig erfüllt sind. Dieser Ansatz bietet lineare Laufzeit, da jeder Datensatz nur einzeln an den Suchanforderungen geprüft werden muss.

Beim Verfahren der exakten Suche wird nicht zwischen Datensätzen, die den Suchanforderungen sehr ähnlich sind, und Datensätzen, die den Suchanforderungen gar nicht entsprechen, differenziert. Genügt kein Datensatz den Suchanforderungen, ist eine Aussage darüber, welche der anderen Datensätze dann zu bevorzugen wären, nicht möglich.

Die in [PM05] vorgestellte Nearest Neighbor-Suche zeigt eine Möglichkeit auf dieses Problem zu lösen, indem die Suchergebnisse entsprechend ihrer Distanz zum Anfragebereich bewertet werden. Es wird vorgeschlagen, die k Datensätze mit geringster Distanz zu einem Anfragepunkt P zu bestimmen. Für die Entfernungsbestimmung wird dabei das euklidische Distanzmaß verwendet.

Die Bewertung der Suchergebnisse nach Distanz ist allerdings problematisch, wenn die Werte in den verschiedenen Spalten nicht miteinander vergleichbar sind, zum Beispiel bei den Dimensionen Drehzahl und Preis. Um in diesem Fall trotzdem Dimensionen miteinander vergleichen zu können, eignet sich die Abstandssuche. Wie bei der Nearest Neighbor-Suche werden alle Datensätze auf Basis der Distanz zum Anfragebereich bewertet und zusätzlich sortiert. Desweiteren kommt ein Gewichtsvektor zum Einsatz, der für die

Dimensionen jeweils angibt, durch welchen Faktor der Abstand zum Anfragebereich vergrößert wird.

Formal ist der Gewichtsvektor G ein Vektor mit den Elementen $g_1, g_2, \ldots, g_m$. Dabei beschreibt m die Anzahl der Dimensionen, an welche die Anfrage gestellt wurde. Für alle g_i mit $i \in \{1, 2, \ldots, m\}$ gilt $g_i \in R^+$. Der Abstandsvektor A mit den Elementen $a_1, a_2, \ldots, a_m$ beinhaltet die Abstände der einzelnen Dimensionen zum Anfragebereich. Auch für alle a_i mit $i \in \{1, 2, \ldots, m\}$ gilt $a_i \in R^+$. Der Gesamtabstand wird unter Einbeziehung des Gewichtsvektors wie folgt berechnet:

$$Abstand = \sqrt{\sum_{i \in \{1, 2, \ldots, m\}} (g_i \cdot a_i)^2}$$

Das Problem bei dem Gewichtsvektor besteht darin, dass sein Inhalt im Allgemeinen nicht hergeleitet werden kann, weil er von den Präferenzen der Anwender und nicht von den Inhalten der Datenbank abhängt. Der Gewichtsvektor muss in Abhängigkeit von diesen Präferenzen sinnvoll gewählt werden. Statt eines Gewichtsvektors könnte auch eine Funktion, in Abhängigkeit von den Werten der einzelnen Dimensionen, als Abstandsmesser zwischen dem Anfragebereich und den Datensätzen verwendet werden. In den meisten Fällen jedoch ist ein Gewichtsvektor ausreichend, um den Abstand zu bestimmen.

Sowohl die Nearest Neighbor-Suche als auch die Abstandssuche erfordern Anpassungen an die Suchanforderungen der Aktordatenbank, insbesondere an die Anfragen mit Intervallen und kategorischen Spalten.

3.2 Der Skyline-Ansatz als Alternative

Der Skyline-Ansatz [BKS01] (auch bekannt als Maximalvektorproblem [KLP75]) ist im Basisfall nur auf die Maximierung (bzw. Minimierung) von Dimensionen ausgerichtet. Wie der Skyline-Ansatz angepasst werden muss, damit er die Suchanforderungen der Aktordatenbank erfüllt, wird in Kapitel 4.1 näher beschrieben.

Die Skyline ist die Menge der interessanten Datensätze einer Datenbank bezüglich der Maximierung (bzw. Minimierung) bestimmter Dimensionen. Wird beispielsweise ein Aktor gesucht, dessen Wirkungsgrad *(η)* und Leistung *(P)* maximal ist, dann ist der Aktor P_1 *(*$\eta = 55\%$ und $P = 180W$*)* im Vergleich zum Aktor P_2 *(*$\eta = 82\%$ und $P = 200W$*)* uninteressant, da P_2 sowohl mehr Leistung als auch einen besseren Wirkungsgrad als P_1 hat. Vergleicht man dagegen die Aktoren P_2 und P_3 *(*$\eta = 73\%$ und $P = 240W$*)* sind beide Aktoren interessant, da zwar P_2 einen besseren Wirkungsgrad aber P_3 eine höhere Leistung hat.

Die Ergebnismenge des Skyline-Ansatzes ist definiert als Menge der Datensätze, die von keinem anderen Datensatz dominiert werden. Ein Datensatz dominiert einen anderen, wenn er in allen Dimensionen entweder den gleichen oder einen besseren Wert hat. Dabei muss er mindestens in einer dieser Dimensionen besser bewertet sein. Die Ergebnismenge des Skyline-Ansatzes wird Skyline genannt. Abbildung 3.1 zeigt eine Skyline für den zweidimensionalen Fall und veranschaulicht, welcher Bereich von einem Datensatz dominiert wird.

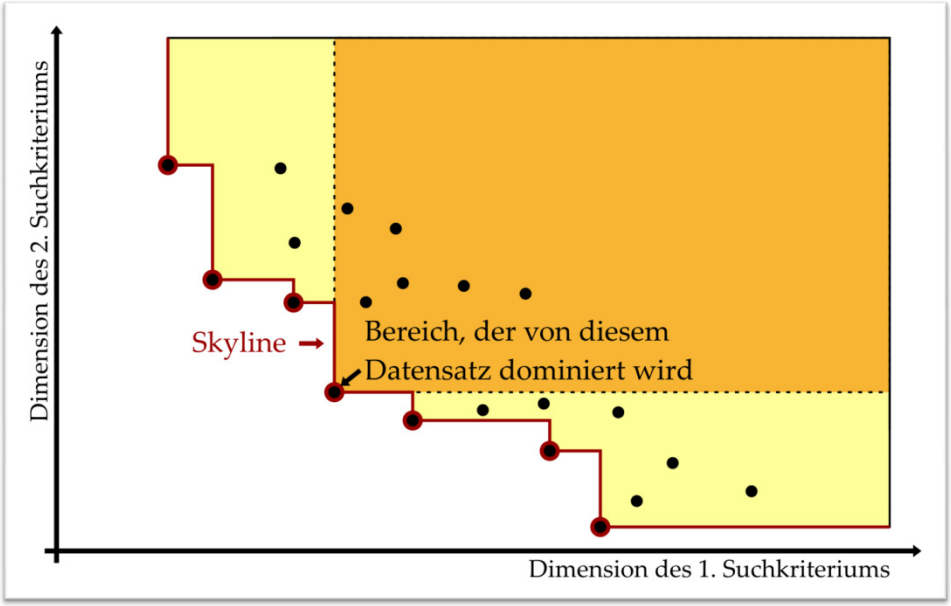

Abbildung 3.1: Beispiel einer Skyline mit zwei zu minimierenden Dimensionen

Formal definiert liegt dem Skyline-Ansatz ein m-dimensionaler Raum $D = (d_1, d_2, \dots, d_m)$ und eine Menge von n Datensätzen $P = (P_1, P_2, \dots, P_n)$ zugrunde. Jeder Datensatz $P_i \in P$ kann dabei als $P_i = (p_{i,1}, p_{i,2}, \dots, p_{i,m})$ dargestellt werden, wobei dann für alle $j \in \{1, 2, \dots, m\}$ folgendes gilt: $p_{i,j} \in d_j$. Weiterhin existiert für jede Dimension $d_i \in D$ die Ordnungsrelation $\succcurlyeq_i$, die je nach Präferenz entweder „≥“ oder „≤“ repräsentiert.

Ein Datensatz $P_x \in P$ dominiert den Datensatz $P_y \in P$ genau dann, wenn $p_{x,i} \succcurlyeq_i p_{y,i}$ für jedes $i \in \{1, 2, \dots, m\}$ gilt und mindestens ein $j \in \{1, 2, \dots, m\}$ existiert, für das $p_{x,j} \neq p_{y,j}$ ist. Für die Dominanzrelation wird der Operator $\prec$ eingeführt. $P_x \prec P_y$ bedeutet: Der Datensatz P_x dominiert den Datensatz P_y. Die Skyline ist folgendermaßen definiert:

$S(P) = \{ P_x \in P \mid \nexists\, P_y \in P : P_y \prec P_x \}$

Jeder Datensatz, den eine Abstandssuche mit geringstem Abstand bewertet, ist auch Teil der Skyline. Denn würde ein solcher Datensatz P_a nicht Element

der Skyline sein und von einem anderen Datensatz P_b dominiert werden, dann hätte der dominierende Datensatz P_b per Definition in jeder Dimension den gleichen oder einen besseren Wert als der Datensatz P_a. Das ist jedoch ein Widerspruch, da dann der dominierende Datensatz P_b bei der Abstandssuche mit geringstem Abstand bewertet sein müsste. Der Gewichtsvektor der jeweiligen Abstandssuche kann bei dieser Argumentation frei gewählt werden. Die daraus resultierende Präferenzunabhängigkeit zeichnet den Skyline-Ansatz aus.

3.3 Verschiedene Skyline-Varianten

Das Maximalvektorproblem ist schon seit Langem in der Informatik bekannt. Als es dann im Jahr 2001 in [BKS01] unter dem Namen Skyline als eine Operation für Datenbanksysteme vorgeschlagen wurde, erhielt es größere Aufmerksamkeit. Seit diesem Vorschlag von Stephan Börzsönyi, Donald Kossmann und Konrad Stocker wurden zahlreiche Skyline-Varianten entwickelt.

Varianten des Skyline-Basisansatzes wurden hauptsächlich aus drei Gründen entwickelt:

- Die Varianten sollen bei niedriger Dimensionszahl mehr Datensätze in die Skyline aufnehmen
- Die Varianten sollen bei hoher Dimensionszahl weniger Datensätze in die Skyline aufnehmen
- Die Varianten sollen ein dem Basisansatz ähnliches Ergebnis liefern, aber dabei eine bessere Laufzeit bieten

An dieser Stelle wird eine Auswahl von Skyline-Varianten vorgestellt. Dabei geht es primär um das Aufzeigen der verschiedenen Möglichkeiten des Skyline-Ansatzes und erst sekundär um die Algorithmen und Laufzeiten der veränderten Ansätze.

Elastische Skylines

Beim Basisansatz werden Datensätze, die in jeder Dimension nur einen geringen Abstand von den Skyline-Datensätzen haben, nicht in die Skyline aufgenommen. Da diese Datensätze jedoch in anderen Eigenschaften Vorzüge gegenüber den Skyline-Datensätzen haben können, werden sie beim elastischen Skyline-Ansatz mit in die Skyline einbezogen.

In [JHE04] wird eine Möglichkeit vorgestellt, bei der die Definition des Dominierens verändert wird. Hier muss der Wert eines Datensatzes pro Dimension um einen ε-Schwellwert besser sein, um einen anderen Datensatz dominieren zu können. Diese Variation wird in Abbildung 3.2 dargestellt.

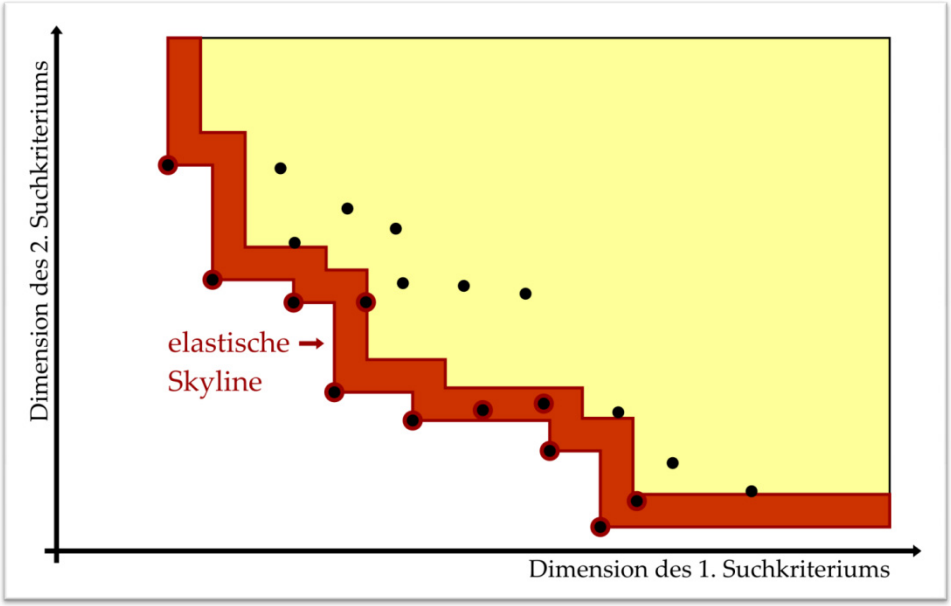

Abbildung 3.2: Beispiel einer elastischen Skyline mit ε-Schwellwerten

Formal wird bei dieser Variante die Dominanzrelation $\prec$ bei der Skyline-Berechnung durch die Dominanzrelation $\prec^1$ ersetzt. Für zwei beliebige Datensätze $P_x \in P$ und $P_y \in P$ ist diese wie folgt definiert: $P_x \prec^1 P_y$ genau dann, wenn $p_{x,i} \succcurlyeq_i p_{y,i} + \varepsilon$ für jedes $i \in \{1, 2, \ldots, m\}$ gilt. Wenn die Ordnungsrelation $\succcurlyeq_i$ dabei „≥“ repräsentiert, dann hat ε einen positiven Wert, im anderen Fall einen negativen.

Eine weitere Möglichkeit ist es, alle Datensätze zur Skyline hinzuzufügen, die sich in einem ε-Abstand zu einem beliebigen Skyline-Datensatz befinden. Dabei entstehen ε-Kugeln um die Skyline-Datensätze, wie sie in Abbildung 3.3 für den zweidimensionalen Fall dargestellt sind.

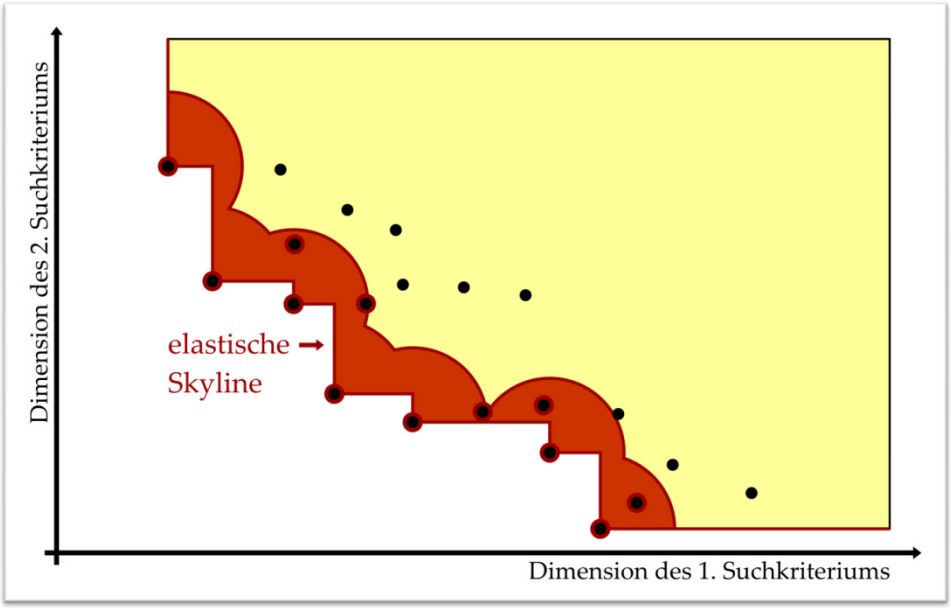

Abbildung 3.3: Beispiel einer elastischen Skyline mit ε-Kugeln

Formal wird die Skyline $S^1(P)$ dieser Variante wie folgt definiert:

$$S^1(P) = S(P) \cup \{ P_x \in P/S(P) \mid \exists P_y \in S(P) : \sqrt{\sum_{i \in \{1, 2, \ldots, m\}} (p_{x,i} - p_{y,i})^2} < \varepsilon \}$$

Für die elastischen Skyline-Ansätze ist wieder ein Gewichtsvektor notwendig, da Abstände innerhalb der einzelnen Dimensionen der Datensätze berechnet werden müssen.

k-Band Skyline

Bei diesem in [PTFS05] vorgestellten Ansatz wird, wie bei den elastischen Skylines, die Skyline vergrößert. Die *k*-Band Skyline besteht aus der Vereinigung aller bei *k* Durchläufen berechneten Skylines. Nach jedem Durchlauf werden die Skyline-Datensätze des letzten Durchlaufs entfernt und fortan nicht mehr beachtet. Dadurch werden im nächsten Durchlauf andere Datensätze in die Skyline aufgenommen. Abbildung 3.4 zeigt eine 2-Band Skyline.

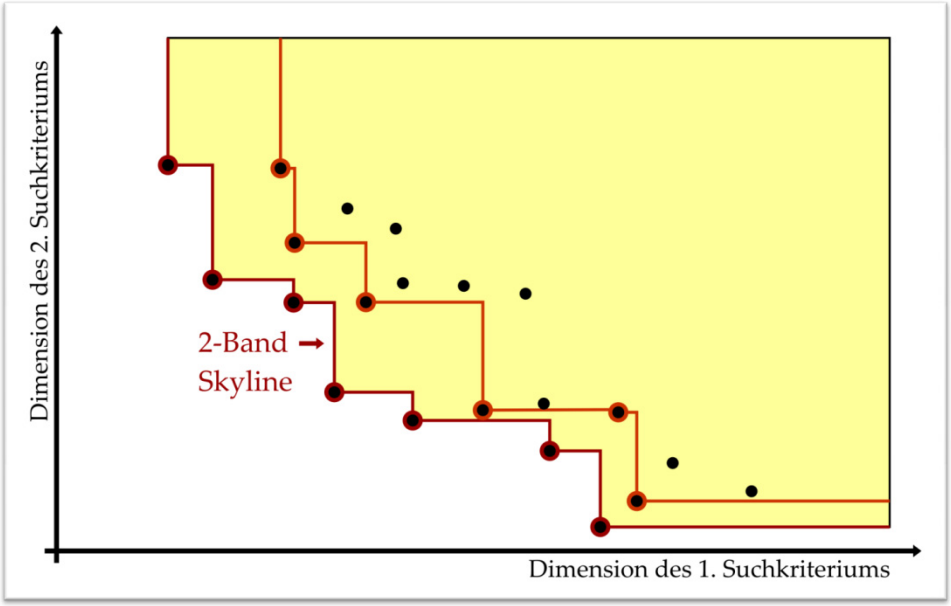

Abbildung 3.4: Beispiel einer *k*-Band Skyline

Formal definiert sei S_k die *k*-Band-Skyline, welche rekursiv berechnet wird. Für $k = 1$ ergibt sich: $S_1 = S(P)$. Für $k \in N$, $k > 1$ ergibt sich:

$S_k = S(P/S_{k-1}) \cup S_{k-1}$.

Top-*k* Skyline

Im Gegensatz zu den letzten beiden Varianten, die die Skyline des Basisansatzes erweitert haben, schränkt der Top-*k* Ansatz die Skyline ein. Diese wird auf die besten *k* Datensätze beschränkt, wobei es verschiedene Möglichkeiten gibt, mit denen die besten Datensätze bestimmt werden können.

Bei dem in [PTFS05] erläuterten Ansatz wird ein Rang innerhalb der Skyline bestimmt. Dieser Rang wird - ähnlich der Abstandssuche - durch eine Bewertungsfunktion auf Basis der Werte der einzelnen Dimensionen erstellt. Die besten *k* Datensätze sind dann diejenigen mit dem höchsten Rang bzw. der höchsten Bewertung. Ein weiterer Ansatz aus [PTFS05] bewertet einen Datensatz auf Basis der Anzahl der durch ihn dominierten Datensätze.

Formal definiert sei $Rang(P_i, M)$ eine beliebige Rangfunktion, die für den Datensatz $P_i \in M$ bezüglich der Datensätze in $M \subseteq P$ den Rang bestimmt. Die Skyline $S^2(P)$ dieser Varianten berechnet sich folgenderweise:

$S^2(P) = \{ P_i \in S(P) \mid Rang(P_i, S(P)) \leq k \}$

Ein allgemein bekannter Top-*k* Ansatz [LYZZ05] bestimmt eine *k*-elementige Teilmenge der Skyline des Basisansatzes. Von allen möglichen *k*-elementigen Teilmengen der Skyline wird die Teilmenge bestimmt, die insgesamt die meisten anderen Datensätze dominiert.

Formal definiert sei $dom(M, P) = |\{ P_i \in P \mid \exists\, m \in M : m \prec P_i \}|$. Das bedeutet $dom(M, P)$ berechnet die Anzahl der Datensätze in P, die durch die Datensätze in M dominiert werden. Die Skyline $S^3(P)$ dieser Variante wird wie folgt berechnet:

$S^3(P) = s \subseteq S(P) \mid k \geq |s|, \nexists\, s' \subseteq S(P) : k \geq |s'|, dom(s', P) > dom(s, P)$

Eine Top-3 Skyline mit diesem Ansatz wird in Abbildung 3.5 dargestellt.

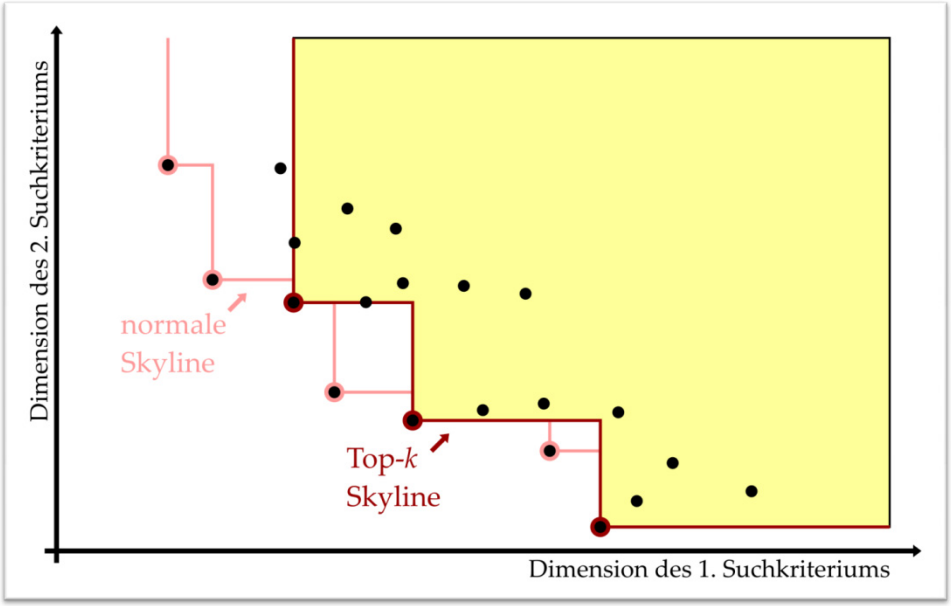

Abbildung 3.5: Beispiel einer Top-*k* Skyline

Ein letzter Top-*k* Ansatz aus [CJTTZ06] bewertet die Datensätze aufgrund der Häufigkeit ihres Vorkommens in Skylines von Unterräumen. Dabei können die Skylines dieser Unterräume entweder approximiert oder vorberechnet werden.

k-dominierende Skyline

Beim Ansatz der *k*-dominierenden Skyline aus [CJTTZ'06] wird die Definition des Dominierens geändert. Ein Datensatz dominiert einen anderen schon dann, wenn er in *k* statt in allen Dimensionen den gleichen oder einen besseren Wert hat. Auch hier gilt, dass der Datensatz in mindestens einer dieser *k* Dimensionen einen besseren Wert haben muss.

Formal wird bei dieser Variante die Dominanzrelation $\prec$ bei der Skyline-Berechnung durch die Dominanzrelation $\prec^2$ ersetzt. Für zwei beliebige

Datensätze $P_x \in P$ und $P_y \in P$ ist diese wie folgt definiert: $P_x \prec^2 P_y$ genau dann, wenn folgendes gilt:

$\exists\, t \subseteq \{1, 2, \ldots, m\} \mid k = |t|, \forall\, i \in t : p_{x,i} \succcurlyeq_i p_{y,i}, \exists\, i \in t : p_{x,i} \neq p_{y,i}$

Das k wird nah an der Anzahl der Dimensionen gewählt. So werden Datensätze, die nur in sehr wenigen Dimensionen gute Werte haben, nicht mehr in die Skyline aufgenommen.

Skyline unter Nebenbedingungen

Die Skyline unter Nebenbedingungen wird in [PTFS03] vorgestellt. Meist werden dabei eine oder mehrere Dimensionen auf bestimmte Intervalle eingegrenzt. So kann zum Beispiel die Drehzahl auf einen Wert zwischen 100 und 200 Umdrehungen pro Sekunde beschränkt werden. Es sind aber auch komplexere Nebenbedingungen in Form von Funktionen denkbar. Beispielsweise könnten nur die Datensätze für die Skyline-Berechnung weiterhin in Betracht gezogen werden, die eine bestimmte Formel erfüllen.

Die Skyline unter Nebenbedingungen wird berechnet, indem zunächst die Datensätze durch die Nebenbedingungen gefiltert werden und anschließend der Skyline-Basisansatz mit den verbleibenden Datensätzen abläuft.

Formal definiert sei $NB(P)$ die Funktion, welche die Datensätze filtert. Dann wird die Skyline $S^4(P)$ unter Nebenbedingungen wie folgt berechnet:

$S^4(P) = S(NB(P))$

Dynamische Skyline

Der dynamische Skyline-Ansatz aus [DS07] entfernt sich von dem Konzept, dass mit einer Skyline immer maximale (bzw. minimale) Werte gesucht werden. Der Skyline-Algorithmus selbst basiert zwar immer noch auf der Maximierung (bzw. der Minimierung), er arbeitet jedoch auf transformierten Datensätzen. Dabei kann sich die Dimensionsanzahl zwischen den transformierten und den nicht transformierten Datensätzen unterscheiden.

Jede neue Dimension der transformierten Datensätze wird durch eine Funktion definiert. Diese Funktionen können zur Berechnung eines transformierten Datensatzes prinzipiell die Werte aller Dimensionen des nicht transformierten Datensatzes verwenden. Das bedeutet, dass die neuen Werte aus den alten Werten des jeweiligen Datensatzes berechnet werden können. Nach dieser Vorverarbeitung läuft der Skyline-Basisansatz auf den transformierten Datensätzen ab.

Formal definiert liegt den transformierten Datensätzen $P^* = (P_1^*, P_2^*, \ \dots \ , P_n^*)$ ein q-dimensionaler Raum $D^* = (d_1^*, d_2^*, \ \dots \ , d_q^*)$ zugrunde. Jeder transformierte Datensatz $P_i^* \in P^*$ wird in den einzelnen Dimensionen $d_j^* \in D^*$ durch die Funktionen F_j^* aus den nicht transformierten Datensätzen $P_i \in P$ berechnet:

$$P_i^* = (p_{i,1}^* = F_1^*(P_i) , \ p_{i,2}^* = F_2^*(P_i) , \ \dots \ , p_{i,q}^* = F_q^*(P_i))$$

In der Praxis handelt es sich bei diesen Funktionen fast ausschließlich um einfache Abstandsfunktionen. Dabei ändert sich die Anzahl der Dimensionen nicht und die Funktionen bestimmen jeweils nur den Abstand zu einem bevorzugten Datensatz. Die dynamische Skyline sucht daher in den meisten Fällen Datensätze, die einem bevorzugten Datensatz gleichen.

Ein Beispiel einer solchen Abstandsfunktion, die den Abstand zum Datensatz $P^+ = (p_1^+, p_2^+, \ \dots \ , p_m^+)$ bestimmt, wird für die Dimensionen $j \in \{1, 2, \dots , m\}$ wie folgt definiert:

$$F_j^*(P_i = (p_{i,1}, p_{i,2} , \dots , p_{i,m})) = \left| p_{i,j} - p_j^+ \right|$$

Für die Anzahl der Dimensionen der transformierten und nicht transformierten Datensätze gilt bei dieser Abstandsberechnung: $m = q$.

Soll der dynamische Skyline-Ansatz nur zu einer solchen Abstandsberechnung verwendet werden, kann die Berechnung der einzelnen Datensätze vereinfacht ausgedrückt werden:

$$P_i^* = (p_{i,1}^* = \left| p_{i,1} - p_1^+ \right| , p_{i,2}^* = \left| p_{i,2} - p_2^+ \right| , \ \dots \ , p_{i,m}^* = \left| p_{i,m} - p_m^+ \right|)$$

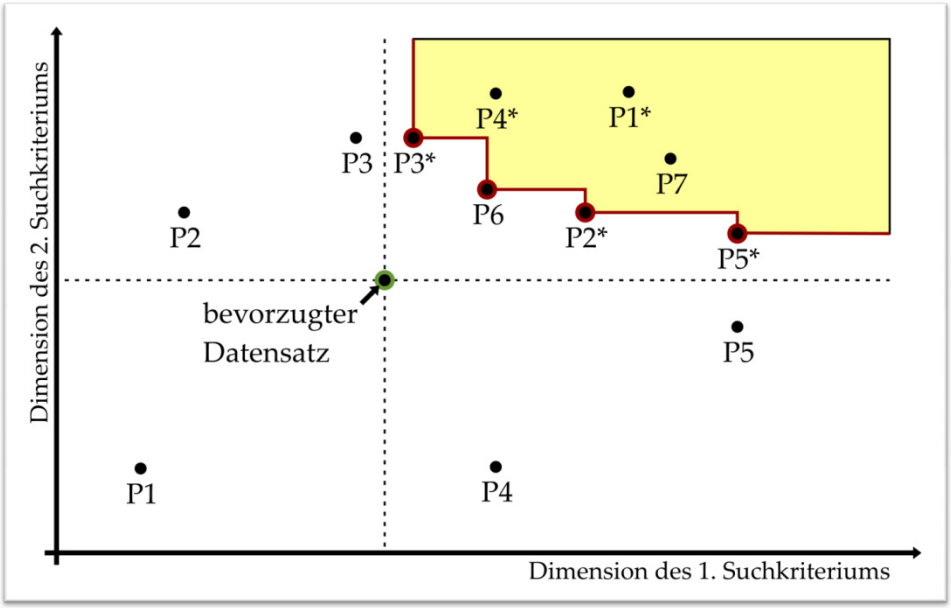

Abbildung 3.6: Beispiel einer dynamischen Skyline

In Abbildung 3.6 sieht man ein Beispiel einer solchen dynamischen Skyline. Für das visuelle Verständnis wurden sowohl die transformierten als auch die nicht transformierten Datensätze eingezeichnet. Die transformierten

Datensätze wurden dabei in den ersten Quadranten verschoben. Hat ein transformierter Datensatz dieselbe Position wie der zugehörige nicht transformierte, wurde nur der nicht transformierte Datensatz eingezeichnet.

3.4 Der selbstorganisierte BNL-Algorithmus

Der BNL-Algorithmus

Der Block Nested Loop (BNL) Algorithmus ist ein oft referenzierter [BKS01, PTFS05, CJTTZ'06, PTFS03] Basisalgorithmus zur Berechnung von Skylines. Die Eingabedatensätze werden dabei nacheinander mit allen Datensätzen einer temporären Skyline verglichen. Diese temporäre Skyline ist auf eine bestimmte Anzahl von Datensätzen beschränkt und kann kurzzeitig auch Datensätze enthalten, welche nicht in die endgültige Skyline aufgenommen werden.

Für einen Eingabedatensatz, der mit der temporären Skyline verglichen wird, gibt es drei Möglichkeiten:

- Dominiert der Eingabedatensatz einen oder mehrere Datensätze aus der temporären Skyline, werden die dominierten Datensätze aus der temporären Skyline entfernt und dafür der Eingabedatensatz in die temporäre Skyline eingefügt.
- Dominiert ein Datensatz aus der temporären Skyline den Eingabedatensatz, wird dieser nicht weiter als Skyline-Element in Betracht gezogen und muss nicht mehr mit den anderen Datensätzen der temporären Skyline verglichen werden.
- Tritt keiner der beiden Fälle auf, wird der Eingabedatensatz in die temporäre Skyline aufgenommen. Würde damit die maximale Anzahl von Datensätzen in der temporären Skyline überschritten, wird der Eingabedatensatz in eine temporäre Datei geschrieben und im nächsten Durchlauf verarbeitet. Alle Datensätze, die in einem Durchlauf in eine temporäre Datei geschrieben wurden, bilden im nächsten Durchlauf die Eingabedatensätze.

Sobald ein Datensatz der temporären Skyline mit jeweils allen Eingabedatensätzen einmal verglichen wurde und er sich immer noch in der temporären Skyline befindet, wird er in die Skyline-Ergebnismenge aufgenommen und aus der temporären Skyline entfernt. Wurde nach einem Durchlauf kein Datensatz in eine temporäre Datei geschrieben, werden alle Datensätze der temporären Skyline in die Skyline-Ergebnismenge aufgenommen und der Algorithmus terminiert.

Pseudocode des BNL-Algorithmus:

```
Skyline={};
while(Exist(Eingabedatei)){
  while(ReadFromFile(Eingabedatei, Eingabe)){
    Ergebnis=Dominate(Eingabe, Skyline);
    if(Ergebnis==„Eingabe wurde dominiert“) continue;
    if(Ergebnis==„Eingabe hat dominiert“) RemoveDominated(Skyline);
    if(Skyline.size()<SkylineLimit) AddToSkyline(Eingabe, Skyline);
    else                            WriteToFile(Tempdatei, Eingabe);
  }
  AddToFinalSkyline(Skyline);
  Skyline={};
  Eingabedatei=Tempdatei;
  Tempdatei=CreateNewFile();
}
```

Um sicher zu stellen, dass ein temporärer Skyline-Datensatz nicht doppelt mit einem Eingabedatensatz verglichen wird, werden Zeitstempel für jeden temporären Skyline-Datensatz eingeführt. Zum Zeitpunkt des Einfügens eines Datensatzes in die temporäre Skyline wird sein Zeitstempel durch die Anzahl der Datensätze in der temporären Datei definiert. Beim nächsten Durchlauf muss genau diese Anzahl von Eingabedatensätzen aus der temporären Datei

gelesen und mit der temporären Skyline verglichen werden, bevor der temporäre Skyline-Datensatz als Element der Skyline bestätigt werden kann.

Der selbstorganisierte BNL-Algorithmus

Das Ziel des selbstorganisierten BNL-Algorithmus aus [BKS01] besteht darin, die Eingabedatensätze mit möglichst wenigen Datensätzen zu vergleichen. Dabei wird angenommen, dass die Datensätze der temporären Skyline die Eingabedatensätze mit unterschiedlich hohen Wahrscheinlichkeiten dominieren. Datensätze, die mit einer verhältnismäßig hohen Wahrscheinlichkeit Eingabedatensätze dominieren, sollten deswegen zuerst verglichen werden. Der aktuelle Eingabedatensatz kann dadurch früher als Skyline-Element ausgeschlossen werden, wenn er bereits durch einen dieser Datensätze dominiert wird. Somit kann der Algorithmus sofort den nächsten Eingabedatensatz bearbeiten.

Dieses Verhalten wird durch eine Heuristik umgesetzt. Dabei wird jeder Datensatz, der einen anderen dominiert, an den Anfang der temporären Skyline verschoben. Eingabedatensätze werden zunächst mit diesen verschobenen Datensätzen verglichen. Im Allgemeinen kann durch diese Heuristik eine Leistungssteigerung erreicht werden. Im schlechtesten Fall wird der Algorithmus durch das Verschieben der Datensätze etwas langsamer.

3.5 Der D&C-Algorithmus

Der in [PS85] beschriebene Divide and Conquer (D&C) Algorithmus funktioniert wie folgt:

- Zuerst wird in einer beliebigen Dimension d_1 aus den Werten der Eingabedatensätze der Median berechnet. Anhand des Medians werden die Eingabedatensätze in die beiden Partitionen P_1 und P_2 unterteilt. Für die Partition P_1 gilt, dass die Werte der Datensätze in der Dimension d_1 kleiner als der Median oder ihm gleich sind. Für die Partition P_2 gilt, dass die Werte der Datensätze in der Dimension d_1 größer als der Median sind.
- Beide Partitionen werden analog zum vorherigen Schritt in jeweils zwei weitere Partitionen unterteilt. Der Median wird jedoch für die Dimension d_2 berechnet. Bei der Dimension d_2 darf es sich aber nicht um die Dimension d_1 handeln $(d_1 \neq d_2)$. Existieren keine zwei verschiedenen Dimensionen, kann der D&C-Algorithmus nicht verwendet werden. Die Partition P_1 unterteilt sich in die beiden Partitionen $P_{1,1}$ und $P_{1,2}$. Für die Partition $P_{1,1}$ gilt, dass die Werte der Datensätze in der Dimension d_2 kleiner als der Median oder ihm gleich sind. Für die Partition $P_{1,2}$ gilt, dass die Werte der Datensätze in der Dimension d_2 größer als der Median sind. Die Partition P_2 unterteilt sich analog in die beiden Partitionen $P_{2,1}$ und $P_{2,2}$.
- Für alle vier Partitionen $P_{1,1}$, $P_{1,2}$, $P_{2,1}$ und $P_{2,2}$ werden jeweils die Skylines $S_{1,1}$, $S_{1,2}$, $S_{2,1}$ und $S_{2,2}$ berechnet. Das geschieht durch die rekursive Anwendung des gesamten D&C-Algorithmus auf die jeweiligen Partitionen. Das rekursive Unterteilen wird beendet, sobald eine Partition nur noch aus einer kleinen Anzahl Datensätze besteht. Die Skyline dieser kleinen Partition kann dann schnell berechnet werden.

- Abschließend werden die Skylines von allen vier Partitionen zusammengefügt. Dabei wird überprüft, ob sich Datensätze der vier Skylines untereinander dominieren. Die dominierten Datensätze werden entfernt und alle anderen Datensätze bilden die Ergebnis-Skyline.

Die Effizienz des D&C-Algorithmus beruht auf dem letzten Schritt. Beim Skyline-Basisansatz können die Datensätze der Skylines $S_{1,2}$ und $S_{2,1}$ sich gegenseitig nicht dominieren und müssen deswegen beim Zusammenfügen der Skylines nicht untereinander verglichen werden. Der Grund dafür ist, dass per Definition alle Datensätze in $S_{1,2}$ in der Dimension d_2 jeweils schlechter sind als jeder einzelne Datensatz aus $S_{2,1}$. Andersherum gilt, dass alle Datensätze in $S_{2,1}$ in der Dimension d_1 jeweils schlechter sind als jeder einzelne Datensatz aus $S_{1,2}$. Die Pfeile in Abbildung 3.7 verdeutlichen, welche Skylines beim ersten Rekursionsschritt verglichen werden müssen.

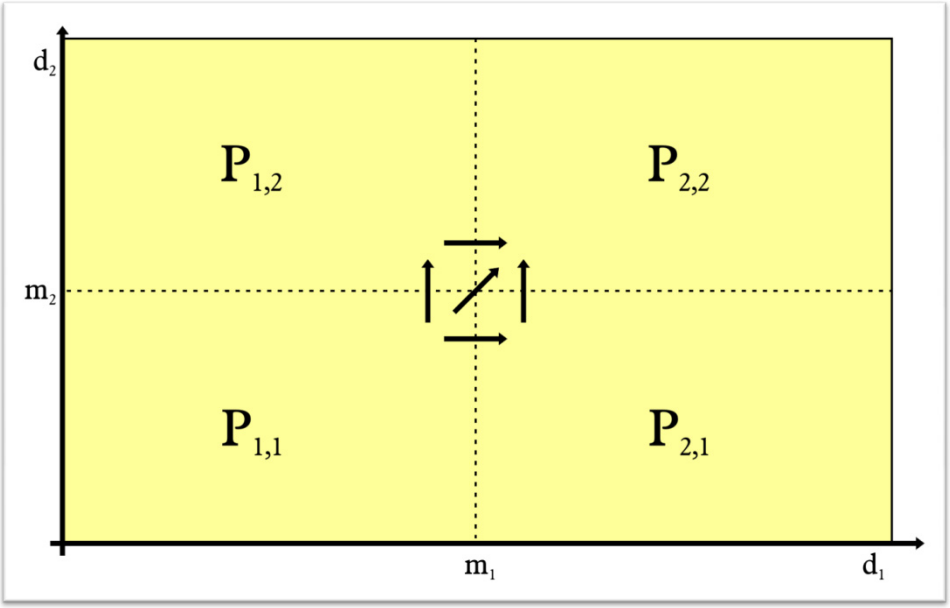

Abbildung 3.7: D&C-Algorithmus beim Skyline-Basisansatz

Pseudocode des D&C-Algorithmus:

```
Skyline DnC_Algorithmus(Eingabe)
{
  if(Eingabe.count()<Schwelle) return Skyline(Eingabe);
  Median1=ComputeMedian(Eingabe, Dimension1);
  Median2=ComputeMedian(Eingabe, Dimension2);
  Divide(Eingabe, Median1, Dimension1, P_1, P_2);
  Divide(P_1, Median2, Dimension2, P_11, P_12);
  Divide(P_2, Median2, Dimension2, P_21, P_22);
  S_11=DnC_Algorithmus(P_11);   S_12=DnC_Algorithmus(P_12);
  S_21=DnC_Algorithmus(P_21);   S_22=DnC_Algorithmus(P_22);
  return Merge(S_11, S_12, S_21, S_22);
}
```

Pseudocode der Divide-Funktion:

```
void Divide(Eingabe, Median, Dim, &Partition_1, &Partition_2)
{
  Partition_1=Partition_2={};
  for each Datensatz in Eingabe{
    if(Datensatz[Dim]<=Median) AddToSet(Datensatz, Partition_1);
    else                       AddToSet(Datensatz, Partition_2);
  }
}
```

Pseudocode der ersten Merge-Funktion:

```
Skyline Merge(S_11, S_12, S_21, S_22)
{
  Skyline={};
  Merge2(S_11, S_12);   Merge2(S_11, S_21);   Merge2(S_11, S_22);
  Merge2(S_12, S_22);   Merge2(S_21, S_22);
  AddToSet(S_11, Skyline);   AddToSet(S_12, Skyline);
  AddToSet(S_21, Skyline);   AddToSet(S_22, Skyline);
  return Skyline;
}
```

Pseudocode der zweiten Merge-Funktion:

```
void Skyline Merge2(Skyline1, &Skyline2)
{
  for each Datensatz1 in Skyline1{
    for each Datensatz2 in Skyline2{
      Ergebnis=Dominate(Datensatz1, Datensatz2);
      if(Ergebnis=„Datensatz1 hat dominiert“){
        Remove(Skyline2, Datensatz2)
      }
    }
  }
}
```

4. Skyline-Varianten für die Aktordatenbank

Ziel dieses Kapitels ist es, für die Aktordatenbank geeignete Skyline-Varianten vorzustellen, die im Rahmen dieser Diplomarbeit entstanden sind. Dazu wird der in Kapitel 3.3 beschriebene dynamische Skyline-Ansatz auf Anfragebereiche erweitert. Anschließend werden weitere im Rahmen dieser Diplomarbeit entstandene Skyline-Varianten vorgestellt. Zuletzt wird sowohl der BNL-Algorithmus als auch der D&C-Algorithmus an die neu entwickelten Varianten angepasst.

Die kategorischen Spalten könnten im Rahmen der in Kapitel 3.3 beschriebenen Skyline unter Nebenbedingungen berücksichtigt werden. Vor der Ermittlung der Skyline würden dann die Datensätze durch die Suchanforderungen, welche an die kategorischen Spalten gestellt wurden, gefiltert werden.

Um das Skyline-Prinzip jedoch auch auf die kategorischen Spalten auszuweiten, werden alle kategorischen Spalten zur Laufzeit wie folgt in numerische Spalten umgewandelt. Wenn Datensätze, in einer kategorischen Spalte der Anfrage entsprechen, bekommen sie in der entsprechenden numerischen Spalte den Wert „0“, ansonsten bekommen sie dort den Wert „1“. In den so umgewandelten numerischen Spalten wird jeweils nach dem Wert „0“ gesucht.

Um die Algorithmen nicht unnötig kompliziert zu gestalten, werden zusätzlich alle Anfragearten an numerische Spalten durch Intervallanfragen realisiert. Die Anfrage, ob ein Datensatz in numerischen Spalten einem Wert entspricht, wird durch eine Intervallanfrage ersetzt, deren Intervall einzig

diesen Wert beinhaltet. Die Anfragen, ob ein Datensatz in numerischen Spalten kleiner oder größer als ein Wert ist, werden durch Intervallanfragen mit einem Intervall vom gesuchten Wert bis zum Maximalwert (bzw. Minimalwert) umgesetzt.

4.1 Dynamische Bereichs-Skyline

Die im Kapitel 2 vorgestellten Suchanforderungen sind komplexer als die Maximierung (bzw. Minimierung) beim Skyline-Basisansatz. Durch die am Ende von Kapitel 3.3 beschriebene dynamische Skyline-Variante ist es zwar möglich, nach bestimmten numerischen Werten in den Spalten zu suchen, für Bereichsanfragen ist diese Variante jedoch nicht geeignet. Daher wurde im Rahmen dieser Diplomarbeit die dynamische Bereichs-Skyline entwickelt.

Dazu werden folgende Annahmen getroffen: Ein Datensatz hat einen besseren Wert in einer Dimension, wenn er einen kleineren Abstand zum jeweiligen Anfrageintervall besitzt als ein anderer Datensatz. Der beste Wert, den ein Datensatz in einer Dimension haben kann, ist ein Wert innerhalb des gesuchten Intervalls. Das bedeutet, es gibt keinen Unterschied zwischen zwei Datensätzen bezüglich einer Dimension, wenn ihre Werte in dieser Dimension jeweils im gesuchten Intervall liegen. Mit diesen Annahmen und der allgemein gehaltenen Definition des Skyline-Basisansatzes wird der dynamische Skyline-Ansatz um Anfragebereiche erweitert.

Formal definiert sei $B = ([b_{1,1}, b_{1,2}], [b_{2,1}, b_{2,2}], \ldots , [b_{m,1}, b_{m,2}])$ der gesuchte Anfragebereich. Für jede Dimension $d_i \in D$ definiert $[b_{i,1}, b_{i,2}]$ das jeweilige Anfrageintervall. Die Formel für F_j^* lautet:

$$F_j^*(P_i = (p_{i,1}, p_{i,2} , \ldots , p_{i,m})) = \begin{cases} b_{j,1} - p_{i,j}, \text{ wenn } p_{i,j} \leq b_{j,1} \\ p_{i,j} - b_{j,2}, \text{ wenn } p_{i,j} \geq b_{j,2} \\ 0, \text{ wenn } b_{j,1} \leq p_{i,j} \leq b_{j,2} \end{cases}$$

Die so entwickelte dynamische Bereichs-Skyline wird in Abbildung 4.1 dargestellt. Für das visuelle Verständnis wurden sowohl die transformierten als auch die nicht transformierten Datensätze eingezeichnet. Zum besseren Vergleich wurden die transformierten Datensätze in beiden Dimensionen um die zugehörige obere Intervallgrenze des Anfragebereichs verschoben. Hat ein transformierter Datensatz dieselbe Position wie der zugehörige nicht transformierte, wurde nur der nicht transformierte Datensatz eingezeichnet.

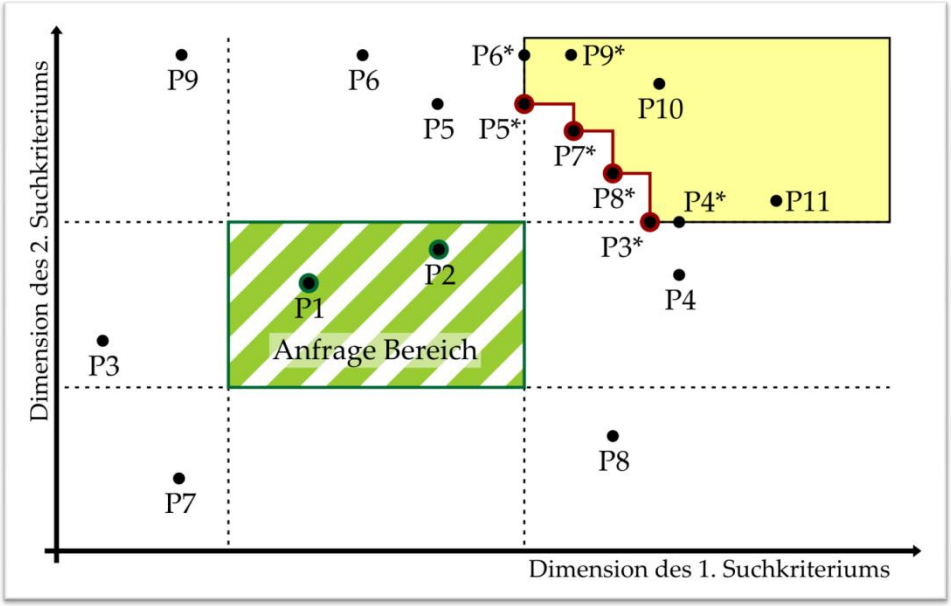

Abbildung 4.1: Beispiel einer dynamischen Bereichs-Skyline

Wenn Datensätze wie P_1 und P_2 in allen Dimensionen im Intervall des Anfragebereichs liegen, würden sie nach der Definition alle anderen Datensätze dominieren. Da die Skyline im Normalfall dadurch sehr klein wird, werden solche Datensätze bei der Berechnung der Skyline ignoriert und direkt als Datensätze erster Klasse in das Ergebnis aufgenommen. Die Ergebnisse zweiter Klasse sind die Datensätze der Skyline.

Diese dynamische Bereichs-Skyline kann nicht durch eine dynamische (Punkt)-Skyline ersetzt werden, indem beispielsweise der Mittelpunkt des Bereichs als Referenzdatensatz genutzt wird. Im Beispiel von Abbildung 4.1 würde sonst der Datensatz P_6 zur Skyline hinzugefügt werden, weil er den kleinsten x-Abstand zum Mittelpunkt hat. Der Datensatz P_5 sollte jedoch den Datensatz P_6 dominieren, da beide Datensätze in der x-Dimension jeweils Werte im gesuchten Intervall des Anfragebereichs haben und der Datensatz P_5 zusätzlich einen kleineren y-Abstand zum Anfragebereich hat.

Erweiterung auf beidseitigen Abstand

Bei der dynamischen Bereichs-Skyline zählt innerhalb einer Dimension nur der Abstand zum Anfrageintervall. Dabei ist es unwichtig, ob ein Wert in der Dimension größer oder kleiner als das Anfrageintervall ist. In manchen Anwendungen macht es aber einen Unterschied, ob eine Eigenschaft schwächer oder stärker als gewünscht ausgeprägt ist. So könnte bei einer gewünschten Drehzahl *(n)* von 30 Umdrehungen pro Sekunde der Aktor P_1 *(n = 25 1/s)* den Aktor P_2 *(n = 37 1/s)* dominieren. Da eine Drehzahl herunter geregelt, aber nicht ohne Probleme erhöht werden kann, ist der Aktor P_2 im Gegensatz zum Aktor P_1 für die gewünschte Aufgabe besser geeignet.

Es ergibt sich folgende Zusatzregel: Wenn zwei nicht transformierte Datensätze in einer Dimension auf verschiedenen Seiten des Anfrageintervalls dieser Dimension liegen, kann keiner der beiden entsprechend transformierten Datensätze den anderen dominieren.

Im Folgenden wird davon ausgegangen, dass die Unterscheidung des beidseitigen Abstands gewünscht ist. In der Abbildung 4.2 wird, im Gegensatz zu Abbildung 4.1, diese Erweiterung verwendet. Dabei müssen die Datensätze auch nicht mehr für das visuelle Verständnis transformiert werden.

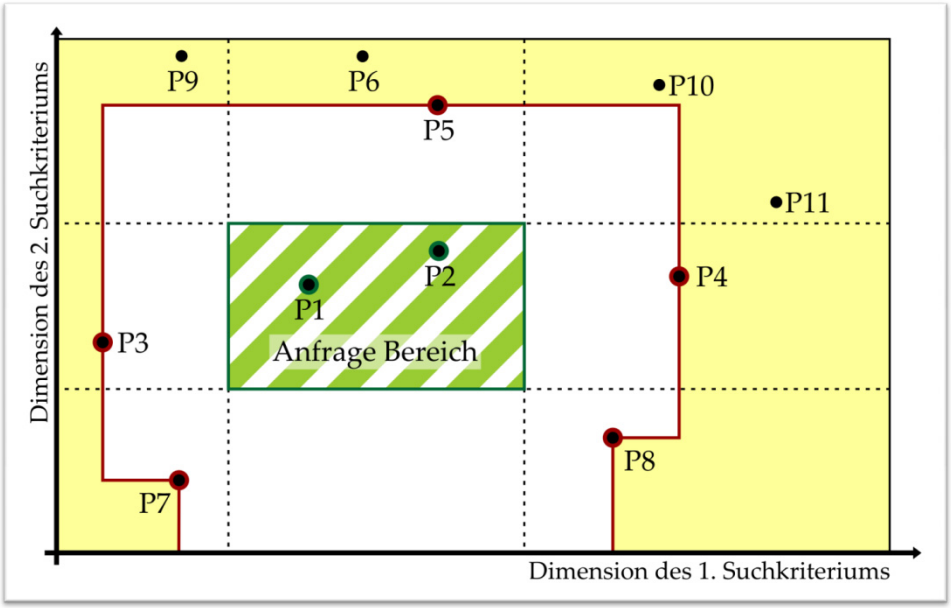

Abbildung 4.2: Vorheriges Beispiel unter Berücksichtigung des zweiseitigen Abstands

4.2 Winkeldominanzansatz

Ein Nachteil der dynamischen Bereichs-Skyline besteht darin, dass an den teilweise langen Rändern des Anfragebereichs (im zweidimensionalen Fall) nur jeweils ein einziger Datensatz in die Skyline aufgenommen wird. Dies ergibt sich, da entlang eines Randes eine der beiden Dimensionen unwichtig wird und dadurch nur der Abstand der anderen Dimension zählt. Das lässt sich auf beliebige Dimensionen erweitern. Im dreidimensionalen Fall gibt es beispielsweise analog dazu Randflächen, an denen nur ein einziger Datensatz in die Skyline aufgenommen wird, weil zwei Dimensionen unwichtig werden.

Der während der Diplomarbeit entstandene Winkeldominanzansatz liefert an den Rändern (bzw. Randflächen) des Anfragebereichs ein breiteres Spektrum an Ergebnissen. Dieser Erweiterung liegt der Gedanke zugrunde, dass ein Datensatz nicht die Datensätze dominiert, die entlang der Ränder weit von ihm entfernt sind. Realisiert wird dieses Verhalten durch die Auswertung eines Winkels zwischen jeweils zwei Datensätzen.

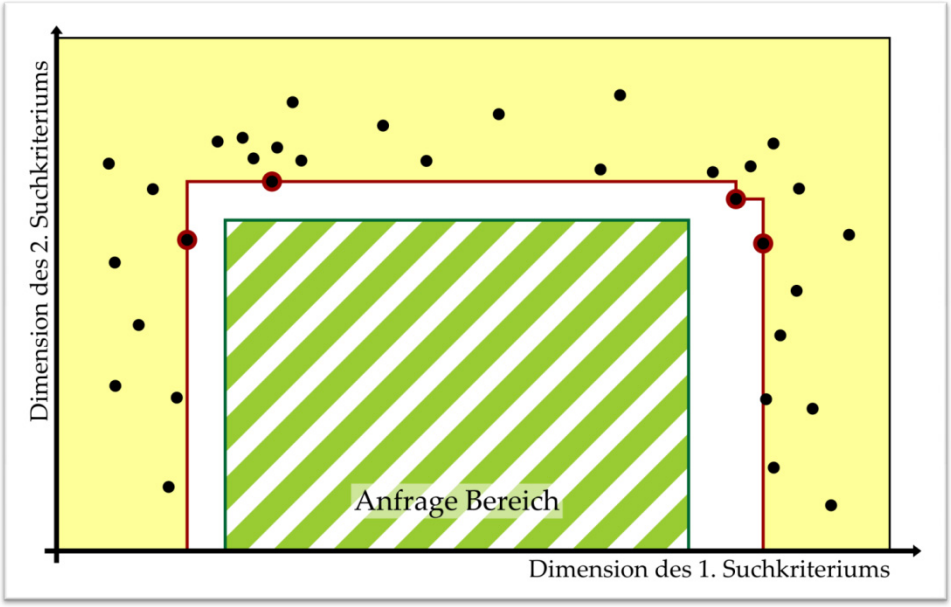

Abbildung 4.3: Beispiel ohne den Winkeldominanzansatz

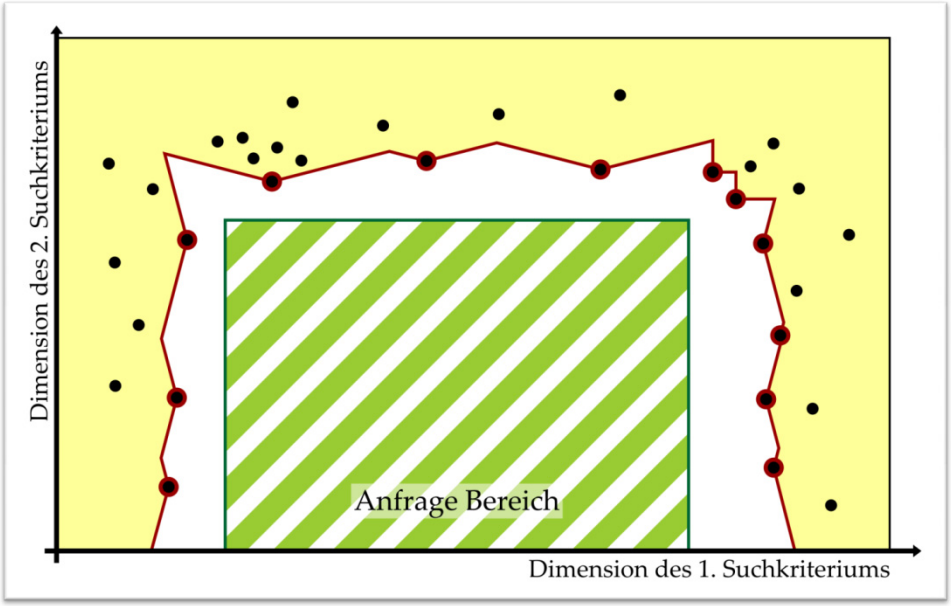

Abbildung 4.4: Beispiel mit dem Winkeldominanzansatz

Den Unterschied zwischen normaler Dominanz und der Winkeldominanz sieht man in Abbildung 4.3 und 4.4.

Um den Winkel zwischen zwei Datensätzen bestimmen zu können, muss wie bei den elastischen Skyline-Ansätzen ein Gewichtsvektor vorliegen, damit die Abstände zwischen den beiden Datensätzen in den einzelnen Dimensionen bestimmt werden können. Für den zweidimensionalen Fall müssen nur die Abstände zwischen den Datensätzen innerhalb der beiden Dimensionen in die folgende Formel (Tangens im rechtwinkligen Dreieck) eingesetzt werden. Die Dimension D_B ist dabei die Dimension, die entlang des Randes verläuft. Die Dimension D_A verläuft demnach senkrecht dazu.

$$tan\,\alpha = \frac{Abstand_{D_A}}{Abstand_{D_B}}$$

Die Formel kann auf eine beliebige Anzahl an Dimensionen erweitert werden. Dabei werden nicht mehr die Abstände einzelner Dimensionen verwendet, sondern der Abstand der Datensätze bezüglich mehrerer Dimensionen. Der

Abstand bezüglich mehrerer Dimensionen $D_i \subseteq \{1, 2, \dots, m\}$ kann durch den Satz des Pythagoras wie folgt berechnet werden:

$$Abstand_{D_i} = \sqrt{\sum_{j \epsilon D_i} Abstand_j^2}$$

Die Abstände von allen Dimensionen, für die gilt, dass der Wert des dominierenden Datensatzes in dieser Dimension im Intervall des Anfragebereichs liegt, bilden den $Abstand_{D_B}$. Alle anderen Dimensionen bilden den $Abstand_{D_A}$.

Der dreidimensionale Fall einer Winkeldominanz wird in Abbildung 4.5 exemplarisch dargestellt. Dabei wird der Winkel zwischen den beiden Datensätzen aus dem Abstand entlang der Seitenfläche und aus dem Abstand senkrecht zur Seitenfläche berechnet.

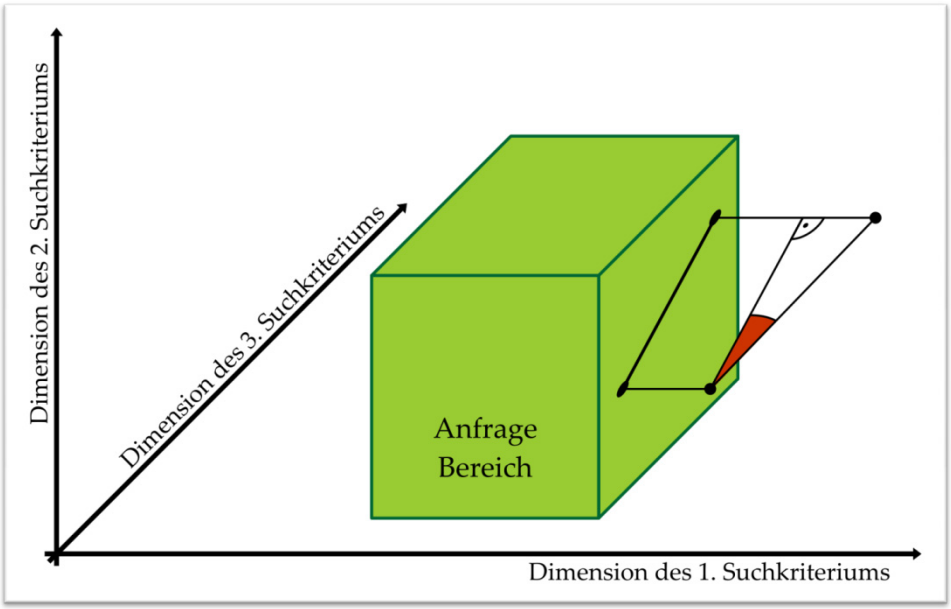

Abbildung 4.5: Berechnung des Winkels im dreidimensionalen Fall

Es ergibt sich folgende Zusatzregel: Wenn zwischen zwei Datensätzen sowohl der $Abstand_{D_A}$ als auch der $Abstand_{D_B}$ definiert ist und wenn der daraus

berechnete Winkel kleiner als ein bestimmter Schwellwert ist, kann keiner der Datensätze den anderen dominieren. Die Abstände sind dann nicht definiert, wenn keine Dimension die jeweiligen Kriterien erfüllt.

Formal wird bei dieser Variante die Dominanzrelation $\prec$ bei der Skyline-Berechnung durch die Dominanzrelation $\prec^3$ ersetzt. Die zwei Teilmengen der Dimensionen $D_B = \{ i \in \{1, 2, \dots, m\} \mid b_{i,1} \leq p_{x,i} \leq b_{i,2} \}$ und $D_A = \{1, 2, \dots, m\} / D_B$ sind für jeden beliebigen Datensatz $P_x \in P$ definiert. Für einen beliebigen Datensatz $P_y \in P$ gilt: $P_x \prec^3 P_y$ genau dann, wenn der folgende logische Ausdruck erfüllt ist:

$$P_x \prec P_y \quad \wedge \quad (|D_A| = 0 \quad \vee \quad |D_B| = 0 \quad \vee \quad tan\,\alpha \geq \frac{\sqrt{\sum_{i \in D_A} (p_{y,i} - p_{x,i})^2}}{\sqrt{\sum_{j \in D_B} (p_{y,j} - p_{x,j})^2}})$$

Zu mehr Ergebnissen an den Rändern des Anfragebereichs würde alternativ auch der in Kapitel 3.3 zuerst beschriebene elastische Skyline-Ansatz führen. Der Unterschied liegt darin, dass beim Winkeldominanzansatz redundante Datensätze ausgeschlossen werden. Das führt zu einer größeren Vielfalt unter den Ergebnissen.

Beide Ansätze können kombiniert werden. Dabei wird die in Kapitel 3.3 beschriebene Dominanzrelation $\prec^1$ verwendet. Weiterhin wird bei der Winkelberechnung ein ε-Schwellwert vom $Abstand_{D_A}$ abgezogen. Durch diese Anpassungen lautet der logische Ausdruck wie folgt:

$$P_x \prec^1 P_y \quad \wedge \quad (|D_A| = 0 \quad \vee \quad |D_B| = 0 \quad \vee \quad tan\,\alpha \geq \frac{\sqrt{\sum_{i \in D_A} (p_{y,i} - p_{x,i})^2} - \varepsilon}{\sqrt{\sum_{j \in D_B} (p_{y,j} - p_{x,j})^2}})$$

Diese Kombination der beiden Ansätze wird im Folgenden verwendet, wenn vom Winkeldominanzansatz gesprochen wird.

4.3 Doppelter Winkeldominanzansatz

In der Aktordatenbank gibt es eine kategorische Spalte, in der die Klasse der Datensätze gespeichert ist. Grundgedanke dieses Ansatzes ist es, dass ein Datensatz mehr Datensätze der eigenen Klasse und weniger Datensätze von anderen Klassen dominieren sollte. Das reduziert das Auftreten von sehr ähnlichen Datensätzen einer Klasse im Ergebnis.

Realisiert wird dieses Konzept durch den gerade vorgestellten Winkeldominanzansatz. Allerdings werden sowohl zwei verschiedene Referenzwinkel als auch zwei verschiedene ε-Schwellwerte genutzt. Bei der Überprüfung, ob ein Datensatz einen anderen dominiert, wird sowohl der Referenzwinkel als auch der ε-Schwellwert abhängig von den Klassen der beiden Datensätze gewählt. Haben beide Datensätze die gleiche Klasse, wird der erste Referenzwinkel und der erste ε-Schwellwert genutzt, sonst der jeweils andere. Sinnvollerweise ist sowohl der erste Referenzwinkel als auch der erste ε-Schwellwert dabei kleiner.

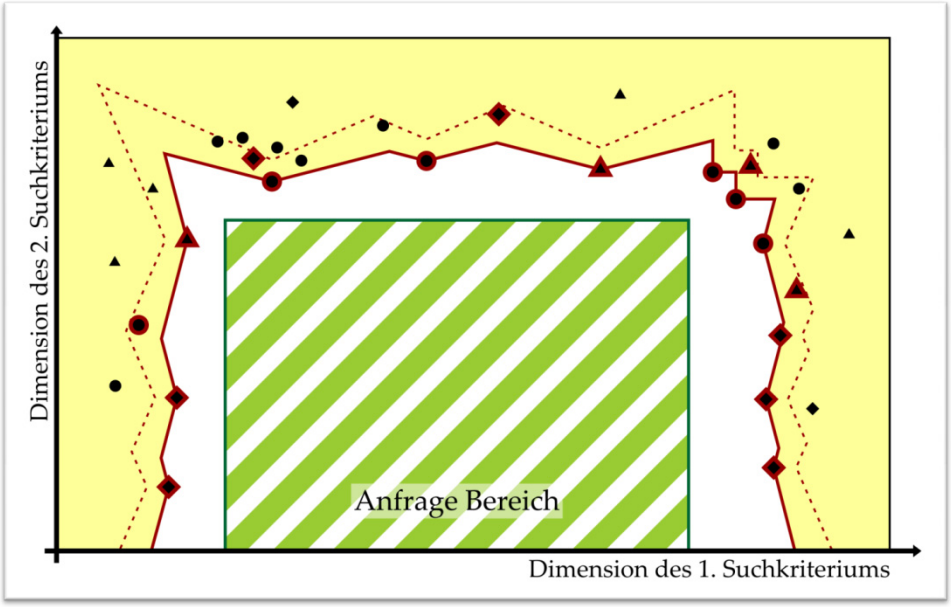

Abbildung 4.6: Beispiel mit dem doppelten Winkeldominanzansatz

Bei dem in Abbildung 4.6 veranschaulichten doppelten Winkeldominanzansatz symbolisieren die verschiedenen Formen die Klassen der Datensätze.

Der Gewichtsvektor

Durch die beiden Winkeldominanzansätze ist es wie bei der Abstandssuche nötig geworden, die einzelnen Dimensionen mit einem Gewichtsvektor zu skalieren. Bei einer Dimension mit einem verhältnismäßig kleinen Wertebereich könnten sich sonst zu viele Datensätze in der Winkeldominanzregion der Skyline befinden.

Um die einzelnen Werte des Gewichtsvektors nicht manuell wählen zu müssen, bietet sich eine Normierung an: Der Gewichtsvektor wird dabei so gewählt, dass nach der Skalierung durch den Gewichtsvektor in jeder Dimension zwischen dem Maximalwert und dem Minimalwert jeweils der gleiche Abstand besteht. Diese automatische Wahl des Gewichtsvektors liefert für die Winkeldominanzansätze gute Ergebnisse.

Sortierung von Skylines

Im Basisansatz ist die Skyline eine Ergebnismenge ohne jegliche Ordnung. Bei jeder Art von Suche spielt aber das Ranking eine sehr große Rolle. Ab einer bestimmten Menge an Suchergebnissen genügt es nicht mehr, dass ein wichtiges Ergebnis im Suchergebnis vorkommt. Wenn es beachtet werden soll, muss es auch zu Beginn der Ergebnisliste erscheinen. Ein geeignetes Beispiel für dieses Prinzip ist die Suchmaschine Google. Auch dort erwartet man die relevantesten Ergebnisse auf den ersten Seiten.

Zunächst werden daher alle Datensätze in drei Klassen eingeteilt. Datensätze erster Klasse erfüllen alle Suchanforderungen, Datensätze zweiter Klasse befinden sich in der Skyline und auf Datensätze dritter Klasse trifft keine der beiden Eigenschaften zu. Nach der Einteilung in Klassen werden die Datensätze zweiter und dritter Klasse entsprechend ihrer Entfernung zum Anfragebereich sortiert.

Die Entfernung eines Datensatzes zum Anfragebereich ist der Mittelwert der Abstände innerhalb der einzelnen Dimensionen. Alle Abstände werden dabei prozentual angegeben. Wenn die Suchanforderungen an eine Dimension vollständig erfüllt sind, ist der Abstand in dieser Dimension 0%. In numerischen Spalten wird der Datensatz mit maximalem Abstand vom Anfrageintervall bestimmt. Dieser Datensatz wird dann mit dem Abstand 100% gewertet. Die anderen Datensätze zwischen Maximalabstand und Anfrageintervall bekommen anteilige Abstandswerte. In kategorischen Spalten erhält ein Datensatz 100% Abstand, wenn die Suchanfrage an die Spalte nicht erfüllt ist, ansonsten 0% Abstand.

Anschließend wird jedem Datensatz eine Relevanz zugeordnet. Das gibt dem Ingenieur zusätzliche Informationen über die Klasse des Datensatzes. Alle Datensätze erster Klasse erhalten eine Relevanz von 100%, Datensätze zweiter Klasse Relevanzen zwischen 90% und 60% und Datensätze dritter Klasse Relevanzen unter 50%. Dabei werden die Datensätze zweiter und dritter Klasse so verteilt, dass der Datensatz mit dem kleinsten Gesamtabstand zum Anfragebereich die größte Relevanz und der Datensatz mit dem größten Gesamtabstand zum Anfragebereich die kleinste Relevanz bekommt. Die anderen Datensätze erhalten eine verhältnismäßige Relevanz.

Zur Anwendung gebrachte Skyline-Varianten

Zur Anwendung kommen letztendlich drei Skyline-Varianten, welche Kombinationen aus einigen der vorgestellten Skyline-Varianten sind. Folgende Ansätze werden genutzt:

- Die dynamische Bereichs-Skyline
- Die dynamische Bereichs-Skyline mit Winkeldominanz
- Die dynamische Bereichs-Skyline mit doppelter Winkeldominanz

Bei allen drei Ansätzen werden die Datensätze nach Bestimmung der Skyline zur besseren Interpretation des Ergebnisses sortiert. Wie beschrieben nutzen die beiden Winkeldominanzansätze jeweils zusätzlich einen ε-Rand vom elastischen Skyline-Ansatz.

4.4 Anpassung der Algorithmen

Anforderungen an die Algorithmen

Da die Algorithmen für eine öffentliche Datenbanksuche im Internet genutzt werden sollen, liegt der Schwerpunkt der Anforderungen auf der Performanz. Deswegen sollten Daten möglichst vorberechnet werden, anstatt sie während der Laufzeit zu bestimmen. Für den Skyline-Basisansatz existieren verschiedene Möglichkeiten zur Vorberechnung von Daten, die einen großen Zeitgewinn ermöglichen. Diese Ansätze können jedoch durch die ständig wechselnden Anfragebereiche der dynamischen Skyline-Ansätze nicht umgesetzt werden.

Eine weitere wichtige Anforderung ist die Externität der Algorithmen, da die Algorithmen auf beliebig großen Datenbanken ablaufen sollten. Es kann daher nicht davon ausgegangen werden, dass alle Datensätze einer

Datenbank gleichzeitig in den Hauptspeicher geladen werden können. Dies bedeutet, dass die Skyline möglicherweise in mehreren Durchläufen berechnet werden muss. Dabei müssen noch nicht bearbeitete Datensätze in temporären Dateien zwischengespeichert werden.

Eine grundlegende Voraussetzung ist, dass die Algorithmen die dynamische Bereichs-Skyline und die Winkeldominanzansätze umsetzten können. Viele herkömmliche Algorithmen für den Skyline-Basisansatz setzen aber Eigenschaften der Skyline voraus, die nicht auf die Winkeldominanzansätze übertragbar sind. Der im Rahmen der Diplomarbeit verwendete BNL-Algorithmus stellt keinerlei Bedingungen an die Dominanzrelation und muss diesbezüglich nicht angepasst werden. Beim ebenfalls verwendeten D&C-Algorithmus ist jedoch eine Anpassungen an die Winkeldominanz nötig.

Veränderungen am BNL-Algorithmus:

Da keine funktionalen Änderungen nötig sind, wird der selbstorganisierte BNL-Algorithmus im Folgenden nur angepasst, um eine bessere Laufzeit zu erreichen.

Wie dem selbstorganisierten BNL-Ansatz in Kapitel 3.4 liegt auch der folgenden Veränderung der Gedanke zugrunde, dass die Datensätze der temporären Skyline die Eingabedatensätze mit unterschiedlich hohen Wahrscheinlichkeiten dominieren. Insbesondere bei Bereichsanfragen an die Aktordatenbank kann das angenommen werden, da hier die Datensätze an den Bereichsrändern verhältnismäßig viele Datensätze dominieren.

Die Anzahl der temporären Skyline-Elemente ist ein interner Parameter des BNL-Algorithmus. Dieser Anzahlparameter verändert die Geschwindigkeit

des Algorithmus. Bei einem kleinen Wert für den Anzahlparameter sind die vielen Dateizugriffe nachteilig. Sie sind jedoch notwendig, da mit jedem Durchlauf nur diese festgelegte Anzahl an Skyline-Elementen zur Skyline hinzugefügt werden kann. Andererseits besitzen die Datensätze in der kleineren temporären Skyline zu Beginn durchschnittlich eine höhere Wahrscheinlichkeit Eingabedatensätze zu dominieren. Das ist vorteilhaft, da alle Datensätze, die nach einem Durchlauf mit einer kleinen temporären Skyline eliminiert wurden, nicht mehr mit den temporären Skyline-Elementen nachfolgender Durchläufe verglichen werden müssen.

In den ersten Durchläufen, in denen die Wahrscheinlichkeit, Datensätze zu dominieren, noch stark schwankt, ist ein kleiner Anzahlparameter günstig. In späteren Durchläufen ist hingegen eine große Anzahl von Datensätzen in der temporären Skyline vorteilhaft, damit keine zusätzliche Zeit für Dateizugriffe benötigt wird. Deswegen empfiehlt es sich, den Anzahlparameter dynamisch zu ändern, indem beispielsweise der Wert des Parameters nach jedem Durchlauf verdoppelt wird.

Veränderungen am D&C-Algorithmus

Wie in Kapitel 3.5 beschrieben, beruht die Effizienz des D&C-Algorithmus darauf, dass einzelne Skylines beim Zusammenfügen nicht untereinander verglichen werden müssen. Bei den dynamischen Bereichs-Skylines können beim Zusammenfügen der Skylines nicht immer zwei Skylines gefunden werden, deren Datensätze sich nicht dominieren.

Sobald der Median in einer oder mehreren Dimensionen im Anfrageintervall liegt, können sich auch die Datensätze auf beiden Seiten des entsprechenden Medians im Anfrageintervall befinden. Nach der Definition der dynamischen

Bereichs-Skylines sind solche Datensätze in der entsprechenden Dimension gleichwertig. Die Argumentation, dass die Datensätze einer Skyline jeweils in einer Dimension schlechter als die Datensätze der anderen Skyline sind, kann also nicht aufrecht erhalten werden. Demnach müssen in diesen Fällen alle Skylines untereinander verglichen werden.

In Abbildung 4.7 werden beim zweiten Rekursionsschritt links oben und rechts unten die Fälle dargestellt, in denen es nicht möglich ist, zwei Skylines zu finden, deren Datensätze sich nicht dominieren können. Rechts oben in der Abbildung ist der Fall dargestellt, bei dem dies möglich ist. Die Pfeile verdeutlichen, welche Skylines untereinander verglichen werden müssen, weil sich Datensätze beider Skylines untereinander dominieren könnten.

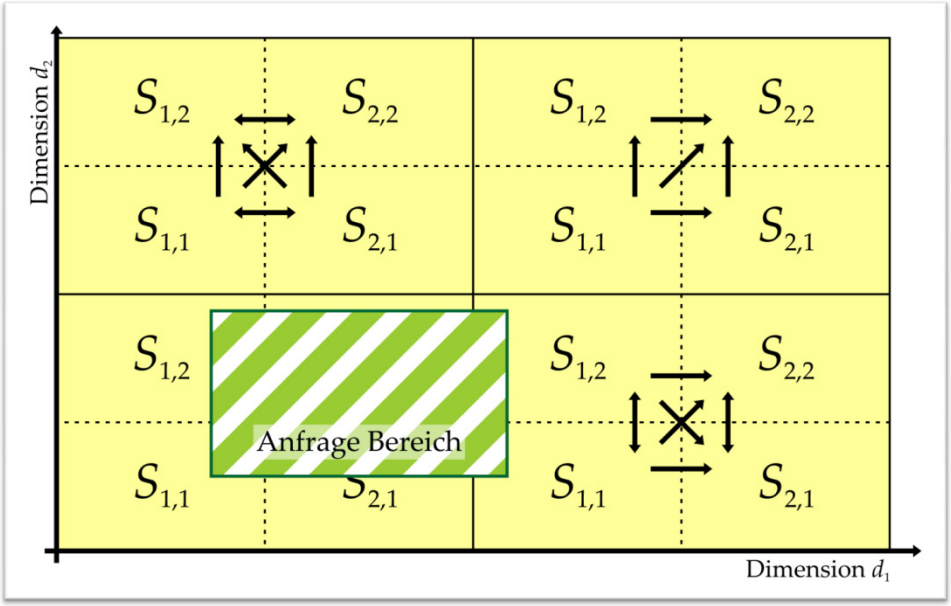

Abbildung 4.7: D&C-Algorithmus bei einer dynamischen Bereichs-Skyline

Große Anfragebereiche können die Skyline-Berechnung verlangsamen, weil es dadurch seltener möglich ist zwei Skylines zu finden, deren Datensätze sich nicht dominieren können. Aus diesem Grund werden für die Dimensionen d_1

und d_2 des D&C-Algorithmus die zwei Dimensionen mit dem kleinsten Anfrageintervall gewählt.

Verwendete Dateistrukturen

Von den Algorithmen werden aus drei Gründen Dateien benötigt:

- Damit die Algorithmen schneller auf die Werte der Datensätze zugreifen können, werden Indexdateien benötigt.
- Damit Datensätze in den Algorithmen ausgelagert werden können, werden temporäre Dateien benötigt.
- Bei der nach Relevanz sortierten Ausgabe der Datensätze werden temporäre Ausgabedateien benötigt.

Indexdateien der Algorithmen

Der BNL-Algorithmus kann beschleunigt werden, wenn die Datensätze nicht aus der Datenbank, sondern aus Indexdateien gelesen werden. Vorteilhaft ist dabei, dass die Daten in Blöcken in den Hauptspeicher eingelesen werden können. Die Dateistruktur dieser Indexdateien wird auch für den D&C-Algorithmus und die temporären Auslagerungsdateien verwendet.

Die Indexdateien sind in einen Header und einen Datenblock aufgeteilt. Am Anfang des Headers werden die Anzahl der Spalten, Anzahl der Partitionen und Anzahl der Datensätze gespeichert. Für jede Spalte werden im Header zwei Werte gespeichert. Der eine Wert enthält die zugehörige Spaltennummer in der Datenbank und der andere die Information, ob es sich bei der entsprechenden Spalte um eine kategorische oder numerische Spalte handelt. Ebenso wird für jede einzelne Partition der Index gespeichert, der den Beginn

der jeweiligen Partition kennzeichnet. Die Informationen zu den Partitionen werden nur beim D&C-Algorithmus benötigt.

Die Werte in den numerischen Spalten werden im Datentyp Double abgelegt und benötigen jeweils 8 Byte Speicherplatz. Die Zeichenketten in den kategorischen Spalten werden durch eine Hashfunktion in den Datentyp Integer umgewandelt und benötigen dadurch nur jeweils 4 Byte Speicherplatz. Wenn es bei den Hashwerten zu Kollisionen kommt, werden bei der Suche nach einer Zeichenkette auch alle Zeichenketten mit kollidierenden Hashwerten gefunden. Kollisionen bei der Berechnung der Hashwerte können aber im Allgemeinen ausgeschlossen werden.

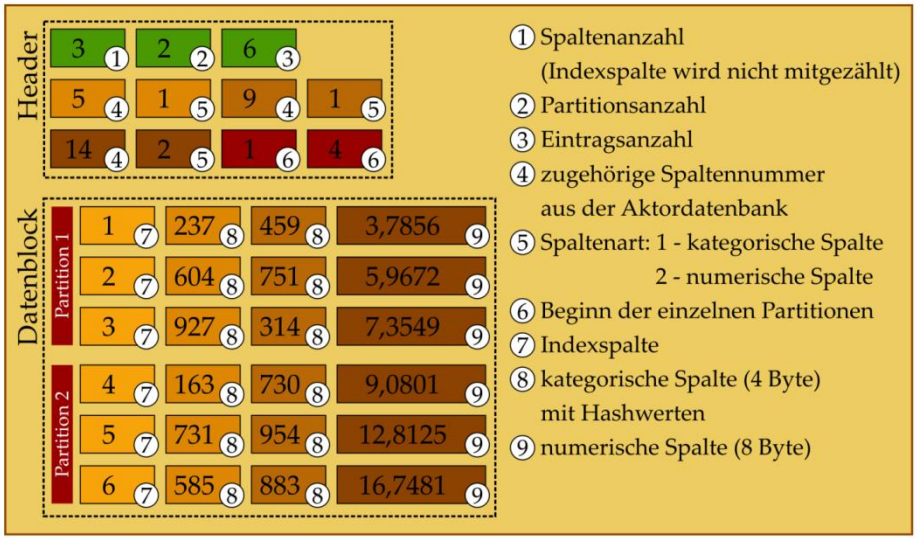

Abbildung 4.8: Dateistruktur der Indexdateien an einem Beispiel

Im Datenblock werden die Datensätze zeilenweise gespeichert. Dabei ist der Speicherplatz pro Zeile durch das Generieren der Hashwerte jeweils konstant. Dadurch können viele Datensätze mit einer einzigen Dateioperation in den Hauptspeicher eingelesen werden und es muss nicht für jeden Datensatz

separat Speicherplatz angelegt werden. Die Dateistruktur ist in Abbildung 4.8 dargestellt.

Um eine Indexdatei aus einer Tabelle der Aktordatenbank zu erstellen, werden alle Datensätze der entsprechenden Tabelle abgefragt und in die Indexdatei geschrieben. Sobald die Datensätze von den Algorithmen benötigt werden, können sie blockweise aus der Indexdatei gelesen werden.

Wenn diese Dateistruktur verwendet wird, um Datensätze in temporäre Dateien auszulagern, werden nur die Spalten in die Datei geschrieben, an die auch Anfragen gestellt werden. Um Namenskonflikte bei den temporären Dateien zu vermeiden, wird die aktuelle Uhrzeit in den Dateinamen übernommen. Alle temporären Dateien werden am Ende des Skyline-Programms gelöscht.

Verwendung der Indexdateien beim D&C-Algorithmus

Um beim D&C-Algorithmus nicht jedes Mal die Partitionen neu bestimmen zu müssen, werden die Datensätze nach Partitionen sortiert in die Indexdatei geschrieben. Das führt zu einer besseren Laufzeit, da weder die Mediane berechnet noch die Datensätze anschließend in die Partitionen sortiert werden müssen.

Da die zu partitionierenden Werte in kategorischen Spalten abhängig von der Anfrage an die Spalte sind, werden alle kategorischen Spalten bei der Generierung der Indexdateien ignoriert. Deswegen werden vom D&C-Algorithmus immer zwei numerische Dimensionen benötigt, für welche die Mediane berechnet werden. Alternativ wird der BNL-Algorithmus verwendet.

Weil die Anfragen jedoch an beliebige Dimensionen gestellt werden können, muss für jedes Dimensionspaar jeweils auch eine Indexdatei existieren. Bei einer Datenbank mit zehn numerischen Spalten ist der Speicherbedarf 45 Mal so groß wie bei den Indexdateien derselben Tabelle für den BNL-Algorithmus.

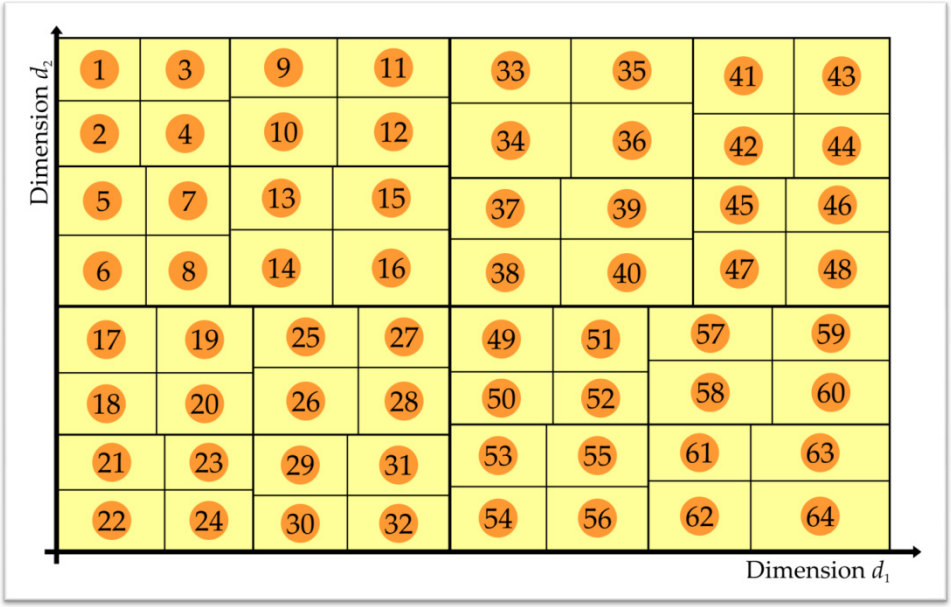

Abbildung 4.9: Reihenfolge der Partitionen in den Indexdateien des D&C-Algorithmus

Damit die Partitionen aus den Indexdateien in fortlaufender Reihenfolge für den D&C-Algorithmus genutzt werden können, werden die Partitionen mit der in Abbildung 4.9 durch die Nummern dargestellten Reihenfolge in der Indexdatei gespeichert.

Durch diese Struktur können jeweils die Skylines von vier fortlaufenden Partitionen in der untersten Ebene sofort zusammengefügt werden. Wurden die Skylines auf einer Ebene vier Mal zusammengefügt, können auf der nächsten Ebene ebenfalls vier Skylines zusammengefügt werden.

Ein Beispiel: Beim Einlesen können sofort die Skylines der Partitionen 1, 2, 3 und 4 berechnet und anschließend die Ergebnisse zusammengefügt werden. In gleicher Weise werden auch die Partitionen 5-8, 9-12 und 13-16 behandelt. Anschließend können die Ergebnis-Skylines dieser vier Schritte erneut zusammengefügt werden.

Um die Dateizugriffe zu reduzieren, werden die Ergebnisse des Zusammenfügens jeweils nur dann in temporäre Dateien ausgelagert, wenn der zulässige Speicherverbrauch überschritten wurde. Dabei werden zuerst die Ergebnisse in den oberen Ebenen ausgelagert, weil diese Ergebnisse bei den Berechnungen als letztes benötigt werden. Benötigte, aber ausgelagerte Ergebnisse, werden aus der entsprechenden temporären Datei geladen.

Sortierte Ausgabe

Um die Datensätze der Ausgabe effizient nach der Relevanz zu sortieren, wurde kein vergleichsbasierter Sortieralgorithmus, wie beispielsweise Quicksort, verwendet. Stattdessen kommt ein angepasster Bucketsort-Algorithmus zur Anwendung. Durch diesen Algorithmus ist das Sortieren der Ergebnisliste in Linearzeit möglich. Um den Bucketsort-Algorithmus anwenden zu können, darf nur eine endliche Anzahl von Schlüsselwerten vorhanden sein. Die Schlüsselwerte sind in diesem Fall die Relevanzen, nach denen die Datensätze der Ausgabe sortiert werden sollen. Wenn die Relevanzen auf eine Nachkommastelle gerundet werden, ist die Voraussetzung der endlichen Anzahl von Schlüsselwerten erfüllt.

Beim Bucketsort-Algorithmus wird eine Liste sortiert, indem zunächst die Anzahl der einzelnen Schlüsselwerte innerhalb der Liste bestimmt wird. Anschließend wird daraus für jeden Schlüsselwert die Einfügeposition in der

sortierten Liste bestimmt. In einem zweiten Durchlauf ist es durch die Einfügepositionen möglich, alle Elemente in einer zweiten Liste sofort an der richtigen Position einzutragen.

Die Datensätze stehen dem Bucketsort-Algorithmus jedoch nicht gleichzeitig zur Verfügung. Ein Datensatz muss immer dann bearbeitet werden, wenn dieser Datensatz durch die Algorithmen endgültig als Datensatz erster, zweiter oder dritter Klasse klassifiziert wurde. Da der Bucketsort-Algorithmus zwei lineare Durchläufe auf den Datensätzen benötigt, könnte dieser Algorithmus nur dann verwendet werden, wenn die Datensätze in Dateien zwischengespeichert würden.

Um beim Bucketsort-Algorithmus nicht auf zwei Durchläufe angewiesen zu sein, werden alle Datensätze mit der gleichen Relevanz gruppiert und in dynamischen Listen gespeichert. Wenn eine Gruppe eine bestimmte Anzahl von Datensätzen erreicht, wird sie in eine temporäre Datei ausgelagert. Bei der Ausgabe können dann die Gruppen nach absteigender Relevanz ausgegeben werden. Wurden Datensätze in Dateien ausgelagert, werden sie bei der Ausgabe genauso behandelt wie die anderen Datensätze der eigenen Gruppe.

5. Implementierung

Die Algorithmen wurden mit MICROSOFT VISUAL STUDIO 2008 in der Programmiersprache C++ implementiert. Die Programmiersprache C++ wurde wegen ihrer Zeiteffizienz gegenüber Programmiersprachen, die beispielsweise den Code nur interpretieren, genutzt. Das Skyline-Programm wurde unter MICROSOFT WINDOWS XP PROFESSIONAL entwickelt. Das Programm hat keine besonderen Systemvoraussetzungen. Für die Nutzung auf dem SUNOS-Server wurde es von der Firma IMMS[1] portiert.

Bei der Aktordatenbank handelt es sich um eine MySQL-Datenbank. Das Webfrontend, in das die Algorithmen integriert werden sollen, ist in der Skriptsprache PHP implementiert. Das Zusammenspiel der einzelnen Komponenten wird in Abbildung 5.1 und das Webfrontend in Abbildung 5.2 dargestellt.

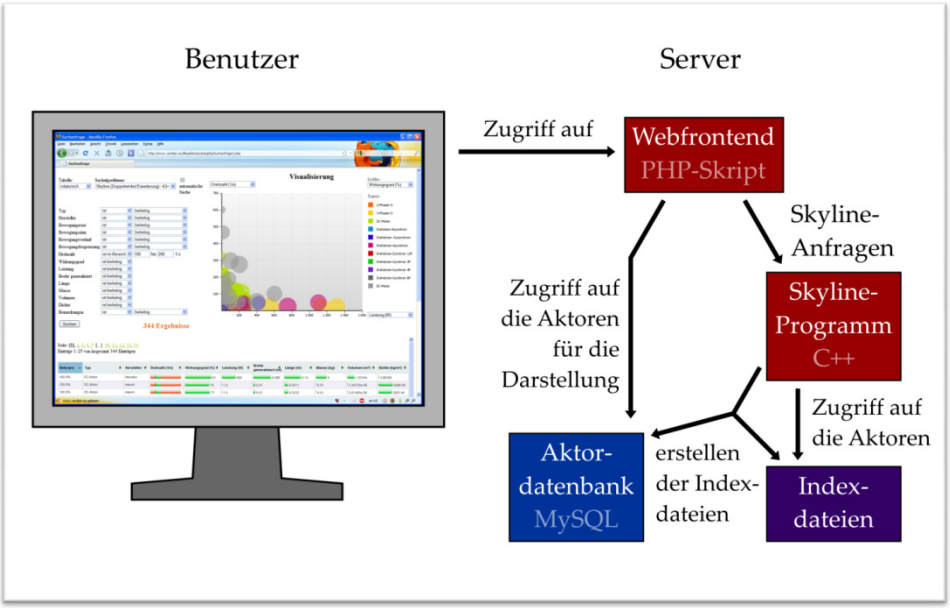

Abbildung 5.1: Zusammenspiel der einzelnen Komponenten

[1] http://www.imms.de/

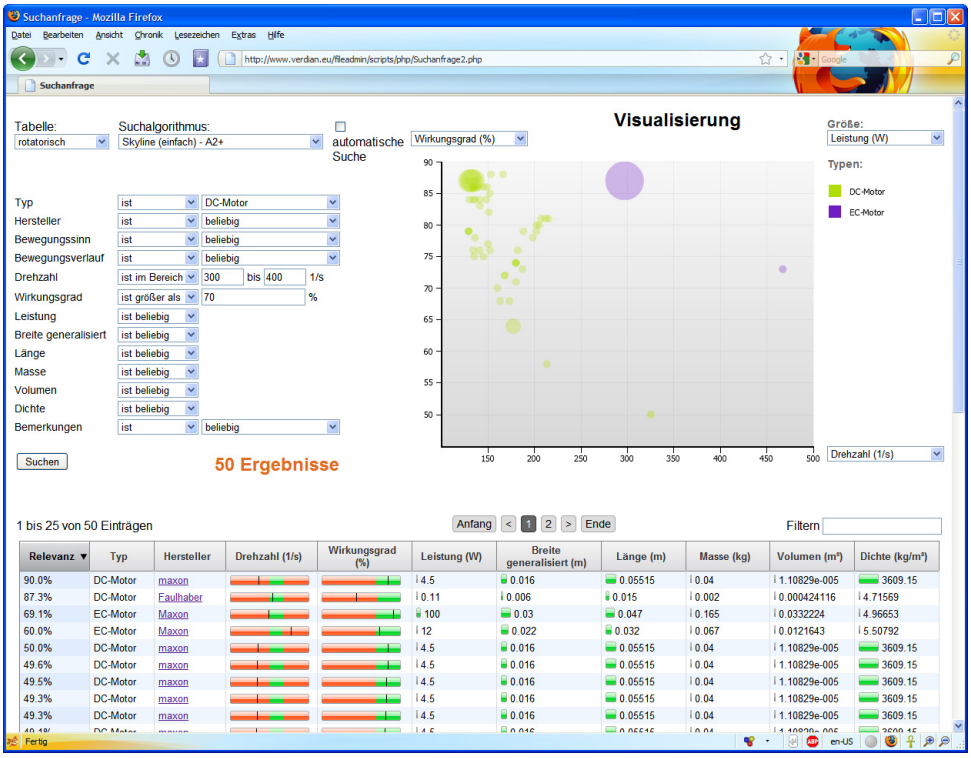

Abbildung 5.2: Webfrontend bei der Berechnung einer Skyline

In dem Skyline-Programm sind folgende Skyline-Varianten implementiert:

- Die dynamische Bereichs-Skyline
- Die dynamische Bereichs-Skyline mit Winkeldominanz
- Die dynamische Bereichs-Skyline mit doppelter Winkeldominanz

Alle drei Varianten sind jeweils für den BNL-Algorithmus und den D&C-Algorithmus implementiert. Die Abstandssuche wird zusätzlich als Referenzimplementierung zugefügt.

Um auch bei zukünftigen Änderungen an der Datenbank flexibel zu sein, wurden sehr wenige Annahmen an die Datenbank gestellt. Aufgrund dieser Flexibilität ist es möglich, das in diesem Kapitel beschriebene Programm auf beliebigen Datenbanken ablaufen zu lassen, solange auch an diese Datenbanken die in Kapitel 2 beschriebenen Suchanforderungen gestellt werden.

5.1 Ein- und Ausgabe

Die Algorithmen und Skyline-Varianten wurden nicht auf mehrere ausführbare Dateien aufgeteilt, da viele Funktionen von den Algorithmen gemeinsam genutzt werden. Sie können über geeignete Parameter durch das Skyline-Programm ausgeführt werden. In MICROSOFT WINDOWS wird das Skyline-Programm durch die ausführbare Datei „Skyline.exe“ gestartet.

Die Ausgabe wird nicht über temporäre Dateien geregelt, sondern um es einfach zu halten, werden die Ergebnisse der Algorithmen in die Standardausgabe geschrieben. Dadurch müssen vom Webfrontend keinerlei dynamische Dateien verwaltet werden. Bei Programmende wird der Rückgabewert gesetzt. Nach fehlerfreier Ausführung des Programms beträgt der Rückgabewert „0“. Trat ein Fehler auf, wird die entsprechende Fehlernummer im Rückgabewert gespeichert und eine Fehlerbeschreibung in die Standardausgabe geschrieben. Insgesamt können 43 verschiedene Fehler bei der Verwendung des Skyline-Programms auftreten. Sowohl die Standardausgabe als auch der Rückgabewert wird mit PHP durch folgenden Befehl bestimmt:

```
string exec(string $befehl [, array $ausgabe [, int $return_var]])
```

Folgender Aufruf des Skyline-Programms erzeugt eine Liste der vorhandenen Algorithmusnamen. Diese dynamische Bestimmung der Algorithmusnamen wird verwendet, um dem Skyline-Programm neue Algorithmen hinzufügen zu können, ohne das Webfrontend anpassen zu müssen. Auf dem gleichen Weg können auch nicht benötigte Algorithmen ausgeblendet werden.

```
Skyline.exe „0“
```

Die Liste der Algorithmusnamen wird nach diesem Aufruf wie folgt in die Standardausgabe geschrieben:

```
Name1|Name2|Name3|Name4|Name5|...
```

Jedem *Name* in der Liste wird eine Algorithmusnummer zugeordnet, die beim Standardaufruf des Skyline-Programms als zweiter Parameter benötigt wird. Die Position eines Namens in der Liste entspricht seiner Algorithmusnummer. Der $Name_2$ gehört beispielsweise zur Algorithmusnummer „2“. Die einzelnen Parametereinträge werden wie angegeben durch vertikale Separatoren „|“ getrennt. In späteren Ein- und Ausgaben werden auch Semikola „;“ zum Trennen der Parametereinträge genutzt.

Mit folgendem Aufruf kann die Skyline mit einem bestimmten Algorithmus berechnet werden:

```
Skyline.exe „Suchstring“ „Algorithmus-Nummer“
```

Der *Suchstring* bei diesem Standardaufruf ist wie folgt unterteilt:

```
Server|Port|Benutzer|Passwort|Datenbank|Tabelle;Anfrage
```

Der *Server* und der *Port* geben die Internetadresse des Datenbank-Servers an. *Benutzer* und *Passwort* ermöglichen das Einloggen in den Datenbank-Server. *Datenbank* und *Tabelle* bestimmen den Ort, an dem gesucht werden soll.

Eine *Anfrage* ist in beliebig viele der folgenden Spaltenanfragen unterteilbar:

```
Spaltennummer|Operator|Vergleichswert1[|Vergleichswert2];
```

Die *Spaltennummer* gibt an, an welche Spalte die Anfrage gestellt wird. Es gibt für numerische und kategorische Spalten jeweils verschiedene *Operatoren*. Bei vielen Operatoren wird nur der *Vergleichswert*$_1$ benötigt. Der *Vergleichswert*$_2$ wird nur bei einem einzigen Operator benötigt.

Für numerische Spalten gibt es vier verschiedene Operatoren. Mit dem Operator „=“ kann in der angegebenen Spalte nach dem Vergleichswert$_1$ gesucht werden. Die Operatoren „<“ und „>“ ermöglichen die Suche nach kleineren und größeren Werten als dem Vergleichswert$_1$. Der Operator „><“ ist der einzige Operator, bei dem beide Werte benötigt werden. Es wird überprüft, ob sich der Wert in dieser Spalte zwischen Vergleichswert$_1$ und Vergleichswert$_2$ befindet.

Für kategorische Spalten gibt es nur zwei Operatoren. Mit dem Operator „==“ wird abgefragt, ob in der angegebenen Spalte der Vergleichswert$_1$ steht. Der Operator „!=“ überprüft, ob etwas anderes als der Vergleichswert$_1$ in der Spalte steht.

Ein Beispiel für den kompletten Suchstring wäre:

```
localhost|7188|root||db1|rotatorisch;1|==|DC-Motor;9|>|230;4|=|400;
```

Bei diesem Beispiel werden in der rotatorischen Tabelle der Datenbank „db1“ die Aktoren gesucht, die in Spalte 1 die Zeichenkette „DC-Motor“, in Spalte 9 ein größeren Wert als 230 und in Spalte 4 den Wert 400 haben.

Das Erstellen der Indexdateien muss nach jedem Datenbankupdate manuell vorgenommen werden. Zu diesem Zweck kann ein separater Algorithmus,

wie die anderen Algorithmen, über das Skyline-Programm aufgerufen werden. Dabei werden Parameter, welche die Anfragen an die Spalten betreffen, ignoriert, ebenso der Parameter „Tabelle“ im Suchstring. Stattdessen werden für alle Tabellen, für die eine Infotabelle existiert, die Indexdateien generiert.

Die Indexdateien können beispielsweise wie folgt berechnet werden:

```
Skyline.exe „localhost|7188|root||db1|rotatorisch;“ „4“
```

Die Ausgabe bei einem Standardaufruf sieht wie folgt aus:

```
Index1|Relevanz1; Index2|Relevanz2; Index3|Relevanz3; ...
```

Bei der Ausgabe werden für die Datensätze jeweils der *Index* und die *Relevanz* ausgegeben. Die Datensätze werden dabei nach absteigender Relevanz sortiert. Datensätze, die eine Relevanz von „0“ haben, werden nicht mit ausgegeben.

Beispiel der Ausgabe bei einem Standardaufruf:

```
67|100.0;34|100.0;59|90.0;17|72.7;82|63.5;35|60.0;21|50.0;69|41.7;
```

In dem Beispiel sind die Aktoren mit den Indizes 67 und 34 Datensätze erster Klasse und haben eine Relevanz von 100%. Die Aktoren mit den Indizes 59, 17, 82 und 35 sind Datensätze zweiter Klasse und haben absteigende Relevanz. Die beiden letzten Aktoren sind Datensätze dritter Klasse.

Zeichenersetzungen in den Parametern

Durch die Texteinträge in den kategorischen Spalten und bei den anderen Textwerten im Suchstring kann es zu Auswertungsschwierigkeiten kommen. So würde beispielsweise bei einer Suche nach einem Spalteneintrag der Form „Text;Rest“ durch das Semikolon nur die Zeichenkette „Text“ gesucht. Das dahinterstehende „Rest“ würde als neue Spaltenanfrage interpretiert werden.

Um solche und ähnliche Doppeldeutigkeiten zu vermeiden, wurden innerhalb der Texteinträge die folgenden Zeichenersetzungen verwendet:

- Leerzeichen („ “) werden durch „\a“ ersetzt.
- Anführungszeichen („ " “) werden durch „\b“ ersetzt.
- Backslashs („\“) werden durch „\c“ ersetzt.
- Vertikale Separatoren („|“) werden durch „\d“ ersetzt.
- Semikola („;“) werden durch „\e“ ersetzt.

Das Webfrontend muss in den Texteinträgen der Parameter alle vorkommenden Doppeldeutigkeiten ersetzen. Nach der Übergabe an das Skyline-Programm werden die Zeichenersetzungen in den Texteinträgen wieder rückgängig gemacht.

Einstellungen für die Algorithmen

In der Initialisierungsdatei „Skyline.ini“ werden grundlegende Parameter festgelegt. Im Gegensatz zu den übergebenen Parametern werden diese aber kaum geändert. Hinter den folgenden Parametern stehen jeweils die Standardwerte, die verwendet werden, wenn auf die Datei „Skyline.ini“ nicht zugegriffen werden kann.

Parameter in der Initialisierungsdatei „Skyline.ini“:

```
Vorzeichen=1            ; soll ein Vorzeichen verwendet werden?
Hauptspeicher=512       ; maximaler Hauptspeicher Verbrauch
MaxSkylineProzent=90.0  ; größter Prozentwert für Skyline-Elemente
MinSkylineProzent=60.0  ; kleinster Prozentwert für Skyline-Elemente
MaxRestProzent=50.0     ; größter Prozentwert für andere Elemente

Winkel=15.0             ; Winkel der Winkel-Dominanz
Epsilon=1.0             ; Epsilon der Winkel-Dominanz
Winkel2=15.0            ; Winkel der doppelten Winkel-Dominanz
Epsilon2=5.0            ; Epsilon der doppelten Winkel-Dominanz
Klasse=herst            ; Klasse der doppelten Winkel-Dominanz
```

Der Parameter *Vorzeichen* bestimmt, ob bei der Abstandsberechnung die Erweiterung auf beidseitigen Abstand aktiviert ist oder nicht. Der Wert „1“ aktiviert die Erweiterung, der Wert „0“ deaktiviert sie. Der Parameter *Hauptspeicher* gibt an, wie viel Speicher (in Megabyte) die Algorithmen maximal verwenden dürfen.

Mit den Prozentparametern kann die Relevanz der Datensätze bei der Ausgabe beeinflusst werden. Die Skyline-Datensätze verteilen sich zwischen *MinSkylineProzent* und *MaxSkylineProzent*. Die restlichen Datensätze werden zwischen 0% und *MaxRestProzent* angeordnet. Wenn die Skyline nicht geordnet werden soll, kann man MinSkylineProzent und MaxSkylineProzent auf den gleichen Wert setzen. Sollen die restlichen Datensätze nicht geordnet werden, kann MaxRestProzent auf 0% gesetzt werden.

Die Parameter *Winkel* und *Epsilon* werden für den Winkeldominanzansatz und den doppelten Winkeldominanzansatz benötigt. Beim doppelten Winkeldominanzansatz kommen zusätzlich die drei Parameter *Winkel2*,

Epsilon2 und *Klasse* dazu. Der Parameter Klasse gibt dabei den Spaltennamen der Spalte an, in der die Klasse der Datensätze gespeichert wird.

Informationen zu den einzelnen Einträgen stehen auch in der Initialisierungsdatei selbst. Dort wird sowohl auf den Zweck eingegangen, als auch auf den jeweiligen Wertebereich und die Einheit. Die Datei „Skyline.ini“ muss sich im gleichen Verzeichnis wie die „Skyline.exe“ befinden, sonst werden Standardwerte verwendet.

5.2 Die Struktur des Skyline-Programms

Für die Entwicklung des Skyline-Programms werden die in Abbildung 5.3 dargestellten Dateien verwendet.

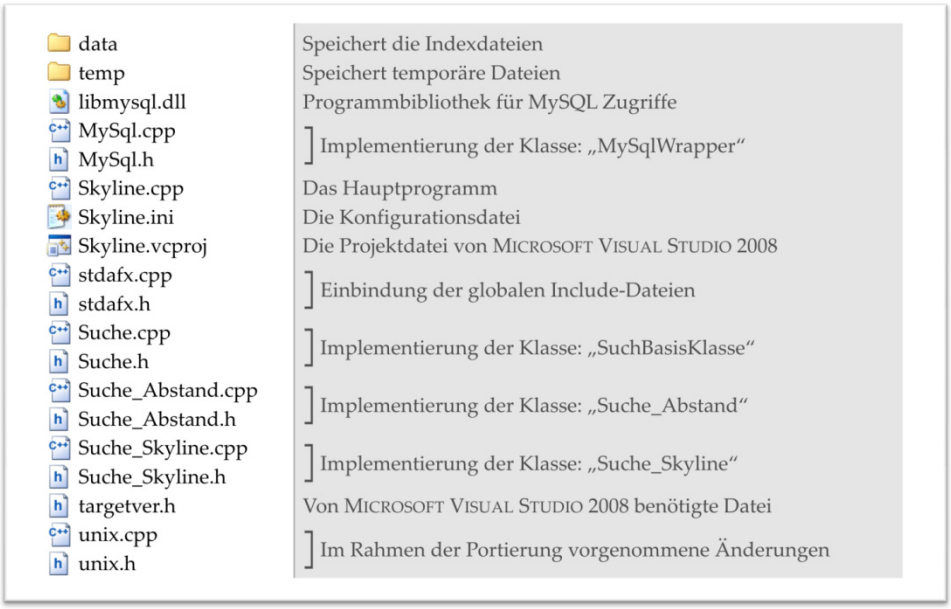

Abbildung 5.3: Verwendete Dateien bei der Entwicklung des Skyline-Programms

Die Klasse „SuchBasisKlasse“ ist das zentrale Element, in dem viele der Grundlagen des Skyline-Programms implementiert sind. Das betrifft zum Beispiel das Einlesen und Überprüfen der Parameter, die sortierte Ausgabe

der Datensätze und die Funktion, welche die Abstände zwischen den Datensätzen bestimmt.

Von der Klasse „SuchBasisKlasse“ sind die beiden Klassen „Suche_Skyline“ und „Suche_Abstand“ abgeleitet. Beide Klassen überschreiben die virtuelle Methode „Start()“ aus der Klasse „SuchBasisKlasse“. Durch diese Methode wird der jeweilige Suchalgorithmus durchgeführt. Das Klassendiagramm ist in Abbildung 5.4 dargestellt.

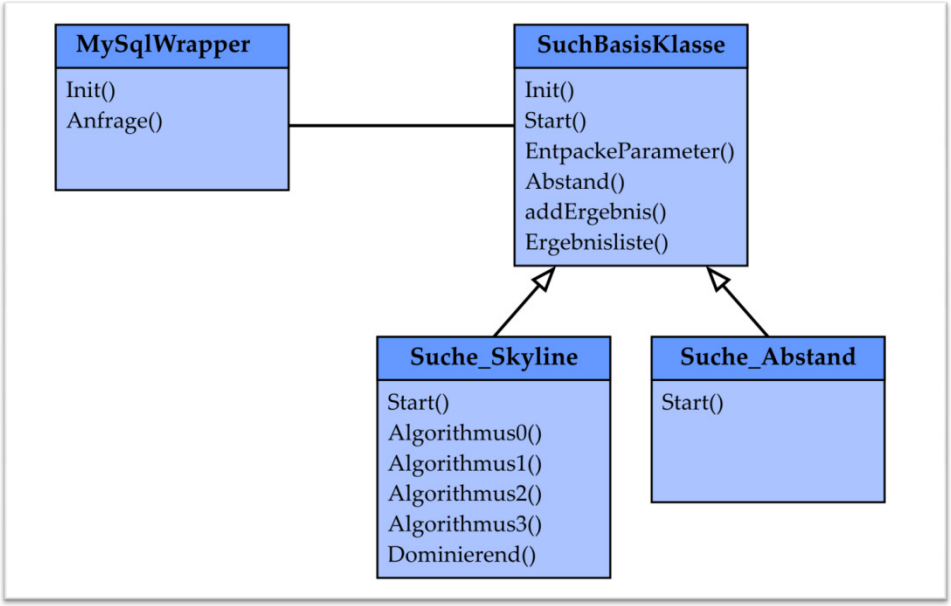

Abbildung 5.4: Klassendiagramm des Skyline-Programms

In der Klasse „Suche_Skyline“ sind die verwendeten Algorithmen und die verschiedenen Skyline-Varianten implementiert. Durch die Methode „Algorithmus0()“ werden die Indexdateien generiert. Zur Berechnung der Skyline verwendet die Methode „Algorithmus2()“ den BNL-Algorithmus und die Methode „Algorithmus3()“ den D&C-Algorithmus. Die Methode „Algorithmus1()“ ist veraltet und berechnet die Skyline mit dem BNL-Algorithmus. Dabei werden keine Speicherbegrenzungen beachtet. Innerhalb

dieser Methoden gibt es kleine Variationen der Algorithmen. Diese wurden im Rahmen der Entwicklung des Programms geschrieben und sind im Vergleich zu den endgültigen Algorithmen langsamer. In der Methode „Dominierend()“ sind alle drei Dominanzrelationen der verwendeten Skyline-Varianten implementiert.

Die Klasse „Suche_Abstand“ wurde als Referenzsuche implementiert. Zur leichteren Anbindung der Algorithmen an die MySQL-Datenbank gibt es die Klasse „MySqlWrapper“. Sie wird von den Suchklassen verwendet, um Anfragen an die Datenbank zu stellen oder um auf die einzelnen Datensätze zuzugreifen. Dies ist jedoch unnötig, wenn Indexdateien zur Anwendung kommen.

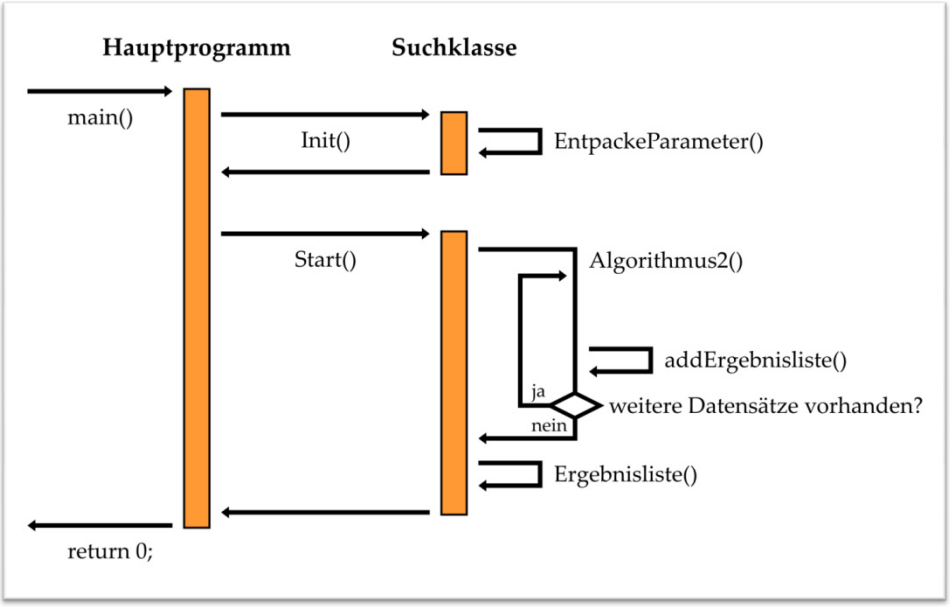

Abbildung 5.5: Sequenzdiagramm einer typischen Skyline-Berechnung

In Abbildung 5.5 wird ein typischer Programmablauf mit einigen der wichtigen Methoden dargestellt. Im Hauptprogramm wird anhand der Algorithmusnummer die entsprechende Suchklasse gewählt. Zur

Initialisierung werden dieser Suchklasse die Parameter und eine Referenz auf die initialisierte „MySqlWrapper“-Klasse übergeben. Anschließend wird die Methode „Start()“ aufgerufen, um den jeweiligen Algorithmus ablaufen zu lassen.

Erweiterbarkeit des Skyline-Programms

Durch die Trennung der Algorithmen von den Dominanzrelationen der Skyline-Varianten können sowohl weitere Algorithmen als auch weitere Dominanzrelationen problemlos hinzugefügt werden. Für den Umstieg auf andere Datenbanksysteme muss allein die „MySqlWrapper“-Klasse angepasst werden.

5.3 Das Programm „BatchIt“

Für die Auswertung der einzelnen Testserien werden viele Testläufe benötigt. Um die einzelnen Testläufe einer Testserie nicht manuell über das Webfrontend starten zu müssen, wurde das Programm „BatchIt“ entwickelt. Dadurch können die Testläufe automatisiert durchgeführt werden.

Das Programm wurde in der freien Skriptsprache AUTOHOTKEY[1] erstellt. Mit AUTOHOTKEY können viele Vorgänge in MICROSOFT WINDOWS automatisiert werden. Dazu steht eine umfangreiche Befehlsbibliothek bereit. Weiterhin besteht die Möglichkeit, den Skripten eine grafische Benutzeroberfläche zu geben. Wie in Abbildung 5.6 ersichtlich, kann dadurch die voraussichtliche Restlaufzeit visualisiert werden.

[1] http://de.autohotkey.com/

Abbildung 5.6: Das Programm „BatchIt“ führt Testläufe durch

Folgende Parameter können vom Programm variiert werden:

- Die Anzahl der Datensätze in der Tabelle. Dies wird durch die Nutzung von verschiedenen Tabellen mit der entsprechenden Anzahl von Datensätzen realisiert.
- Die Anzahl der Dimensionen, an welche die Anfrage gestellt wird.
- Die Algorithmen, mit denen das Skyline-Programm läuft.

Für jede mögliche Kombination aus allen variierenden Parametern wird das Skyline-Programm ausgeführt. Die gewünschten Parametervariationen werden aus der Initialisierungsdatei „BatchIt.ini“ gelesen. Die gemessene Laufzeit für jeden dieser Testläufe wird in einer Ausgabedatei geloggt. Zusätzlich wird die Standardausgabe der Testläufe in Dateien umgeleitet. Auf diese Weise kann auch ausgewertet werden, wie viele Datensätze zur Skyline gehören und ob die Algorithmen die gleiche Skyline ausgeben, wenn identische Skyline-Varianten genutzt wurden.

Teilweise unterscheiden sich die Laufzeiten bei Testläufen mit gleichen Parametern. Gründe dafür können Speicher- und Dateifragmentierung oder Hintergrundprozesse sein. Auf Grund der unterschiedlichen Laufzeiten ist es im Programm „BatchIt“ auch möglich, einen Testlauf mehrere Male zu

wiederholen und anschließend den Durchschnittswert aus den Laufzeiten zu bilden. Dadurch werden genauere Ergebnisse erzielt.

6. Auswertung

Bei der Auswertung werden zunächst die Varianten des BNL-Algorithmus untereinander verglichen. Anschließend wird die beste Variante des BNL-Algorithmus mit dem D&C-Algorithmus verglichen. Zuletzt werden die verschiedenen Skyline-Varianten gegenübergestellt. Das Hauptaugenmerk liegt dabei auf der Laufzeit der jeweiligen Berechnungen.

Der verwendete Testrechner hat einen „AMD Athlon 64 X2"-Dualprozessor. Beide Prozessorkerne arbeiten jeweils bei einer Taktfrequenz von 2,3 Gigahertz. Von den Prozessorkernen kann aber nur einer genutzt werden, da das Skyline-Programm nicht auf mehrere Threads aufgeteilt ist. Der Testrechner arbeitet mit 1 Gigabyte Arbeitsspeicher.

Grundlage für die folgenden Vergleiche sind Testserien mit jeweils 10.000, 50.000 und 100.000 Datensätzen in den Tabellen. Die Werte der Datensätze sind in jeder Dimension im Intervall von [0,1000] gleichverteilt. Diese Wahrscheinlichkeitsverteilung bildet die Anordnung der Datensätze in der bestehenden Aktordatenbank gut ab.

Innerhalb jeder Testserie wird auch die Anzahl der Dimensionen variiert, an welche die Anfrage gestellt wird. Dabei variiert die Anzahl zwischen einer Dimension und zehn Dimensionen. An jede einzelne Dimension wird die Anfrage gestellt, ob sich der Wert der Datensätze in der jeweiligen Dimension im Intervall von [450,550] befindet. Jeder Testfall wurde zehn Mal getestet, um den Einfluss von Laufzeitschwankungen zu reduzieren. Für das Berechnen dieser Testserien benötigte der Testrechner etwa fünf Tage Rechenzeit.

6.1 Varianten des BNL-Algorithmus

Einfluss der Indexdateien

Zunächst wird untersucht, ob es beim BNL-Algorithmus Laufzeitunterschiede bei der Verwendung von Indexdateien anstelle von Datenbankanfragen gibt. In allen getesteten Fällen wird ein Geschwindigkeitsvorteil beim Verwenden der Indexdateien festgestellt. Die Laufzeit wird dabei im Durchschnitt um etwa 20% verringert. Durch das Verwenden von Indexdateien werden durchschnittlich etwa 3,5 Sekunden pro 100.000 eingelesener Datensätze eingespart.

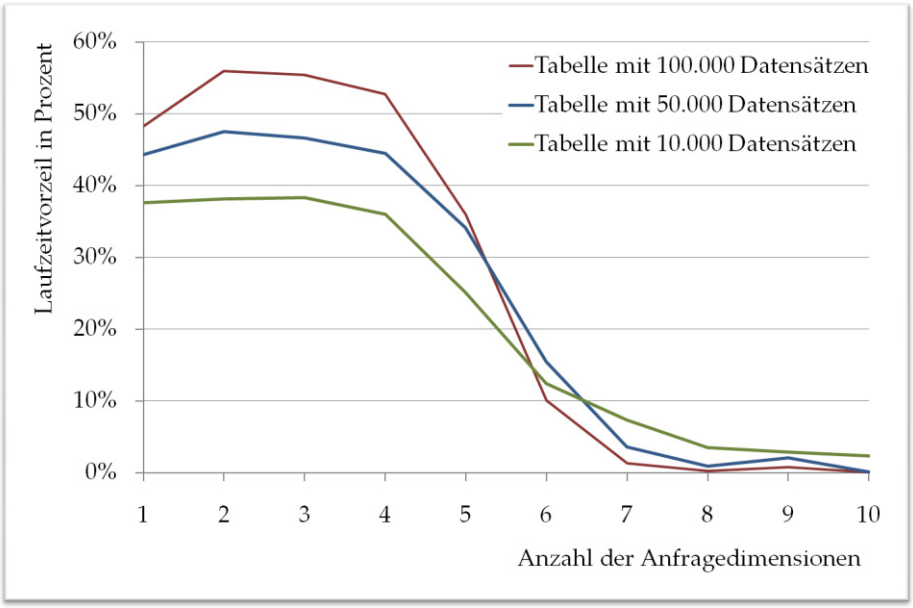

Abbildung 6.1: Laufzeitvorteil in Abhängigkeit der Anfragedimensionen

Bei Testläufen, in denen das Einlesen der Eingabedatensätze einen großen Anteil an der Gesamtlaufzeit hat, kann die Laufzeit sogar bis zu etwa 55% verbessert werden. Der prozentuale Geschwindigkeitsvorteil sinkt mit zunehmender Anzahl der Spalten, an die Anfragen gestellt werden. Wie der Laufzeitvorteil in Abhängigkeit der Anfragedimensionen abnimmt, ist in Abbildung 6.1 dargestellt.

Bei den berechneten Testserien werden in den Tabellen nur numerische Spalten verwendet. Es ist anzunehmen, dass durch das Hashing in den kategorischen Spalten weitere Geschwindigkeitsvorteile erzielt werden.

Einfluss des selbstorganisierten BNL-Ansatzes

Weiterhin wird der Einfluss des selbstorganisierten BNL-Ansatzes im Gegensatz zum normalen BNL-Ansatz untersucht. In den getesteten Fällen haben beide Ansätze ähnliche Laufzeiten, denn durchschnittlich wird die Laufzeit durch den selbstorganisierten BNL-Ansatz nur um 4,2% verringert. Die maximal erreichten Geschwindigkeitsvorteile des selbstorganisierten BNL-Ansatzes liegen bei etwa 50%, jedoch kann es andererseits zu Laufzeitverlängerungen von bis zu 5% kommen.

Einfluss des dynamischen Anzahlparameters

In Kapitel 4.4 wird die Nutzung eines dynamischen Anzahlparameters vorgeschlagen, dessen Aufgabe es ist, die Anzahl der Elemente in der temporären Skyline zu Beginn stärker einzuschränken als notwendig. Nach jedem Durchlauf des Algorithmus wird dieser Anzahlparameter verdoppelt. Bei diesem Ansatz kommt es im Durchschnitt zu einem Laufzeitvorteil von 7,7%. In den besten Fällen können Geschwindigkeitsvorteile von bis zu 65% erreicht werden. Es kann allerdings auch eine Laufzeitverlängerung von bis zu 5% entstehen.

6.2 BNL- und D&C-Algorithmus im Vergleich

Aufgrund der Laufzeitanalysen in Kapitel 6.1 wird im Folgenden stets der selbstorganisierte BNL-Algorithmus mit Indexdateien und mit dem dynamischen Anzahlparameter zum Vergleich mit dem D&C-Algorithmus verwendet. In den Testserien konnte festgestellt werden, dass beide Algorithmen für jeweils eine Skyline-Variante die gleichen Ergebnisse liefern. Das wichtigste Vergleichskriterium der beiden Algorithmen ist demnach die Laufzeit. Aufgrund der stark steigenden Laufzeiten bei großen Anfragedimensionen wird in den Abbildungen eine logarithmische Achseneinteilung verwendet.

Laufzeit der Algorithmen

Die Abbildungen 6.2, 6.3 und 6.4 zeigen das Laufzeitverhalten der beiden Algorithmen für die im Rahmen der Diplomarbeit entstandenen Skyline-Varianten. In jedem Diagramm wird dabei jeweils die Anzahl der Datensätze zwischen 10.000 und 100.000 Datensätzen variiert.

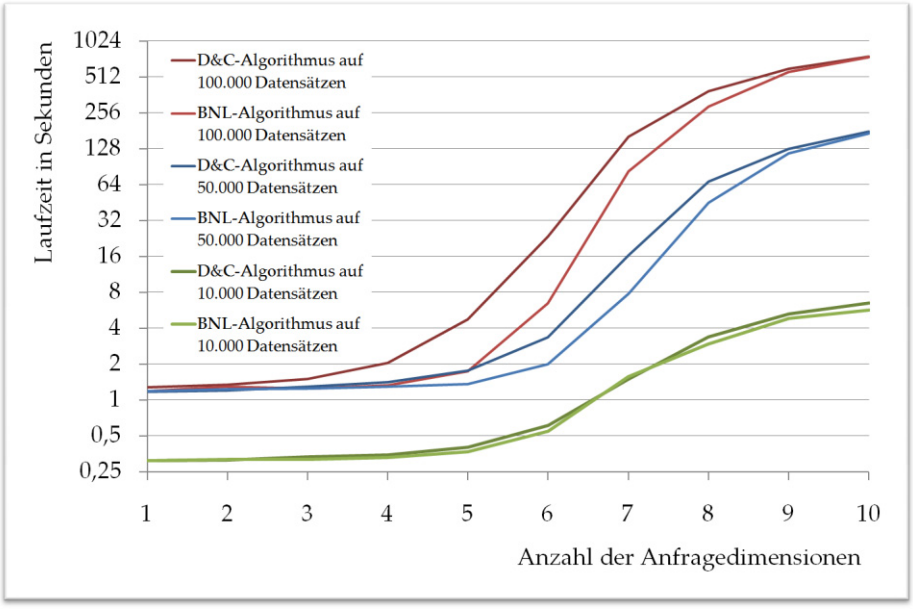

Abbildung 6.2: Laufzeitverhalten der dynamischen Bereichs-Skyline

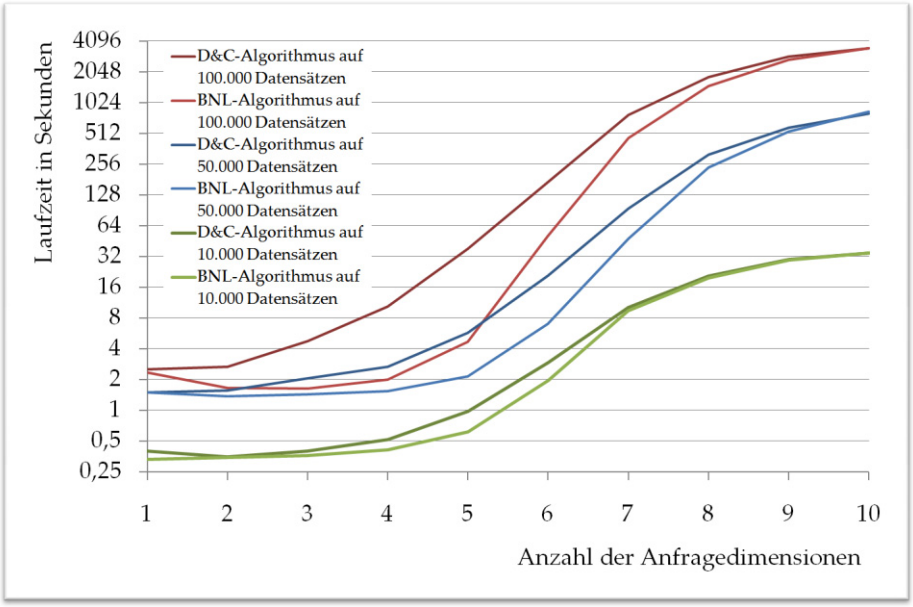

Abbildung 6.3: Laufzeitverhalten mit Winkeldominanz

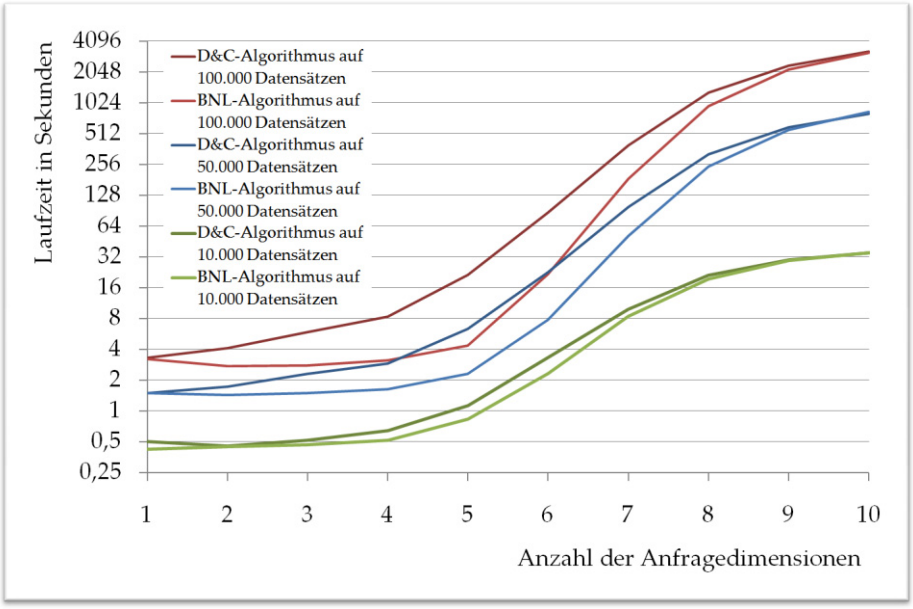

Abbildung 6.4: Laufzeitverhalten mit doppelter Winkeldominanz

Der BNL-Algorithmus verfügt gegenüber dem D&C-Algorithmus durchschnittlich über einen Laufzeitvorteil von 28,5%. Teilweise läuft der BNL-Algorithmus sogar bis zu 88% schneller ab. Laufzeitverlängerungen treten nur vereinzelt auf und betragen maximal 5%. Bei beiden Algorithmen ist ab fünf Anfragedimensionen ein großer Geschwindigkeitsabfall zu verzeichnen.

Laufzeit beim Erstellen der Indexdateien

Wenn Indexdateien für eine Tabelle mit 100.000 Datensätzen und 10 numerischen Spalten erstellt werden sollen, benötigt der BNL-Algorithmus dafür 4,3 Sekunden. Der D&C-Algorithmus dagegen benötigt dafür mit 53,9 Sekunden fast die 13-fache Zeit. Außerdem werden dabei statt 8 Megabyte fast 400 Megabyte Speicher auf dem Datenträger benötigt.

6.3 Die verwendeten Skyline-Varianten im Vergleich

Laufzeit der Skyline-Varianten

Die Abbildungen 6.5, 6.6 und 6.7 zeigen das Laufzeitverhalten der drei Skyline-Varianten für Tabellen mit jeweils 10.000, 50.000 und 100.000 Datensätzen. Dabei wurde die Abstandssuche als Referenz hinzugefügt. In den Testläufen ist jeweils der schnellere BNL-Algorithmus verwendet worden.

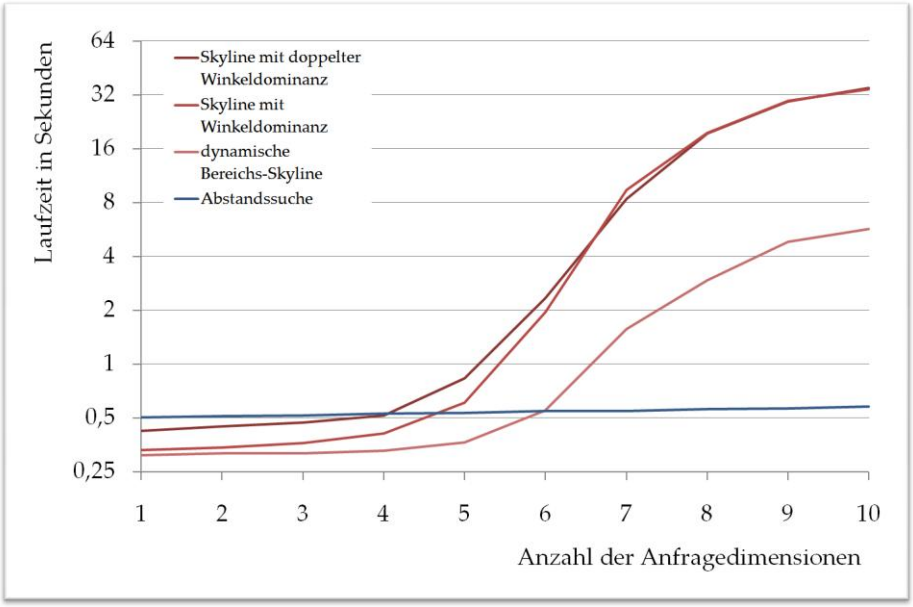

Abbildung 6.5: Laufzeitverhalten bei einer Tabelle mit 10.000 Datensätzen

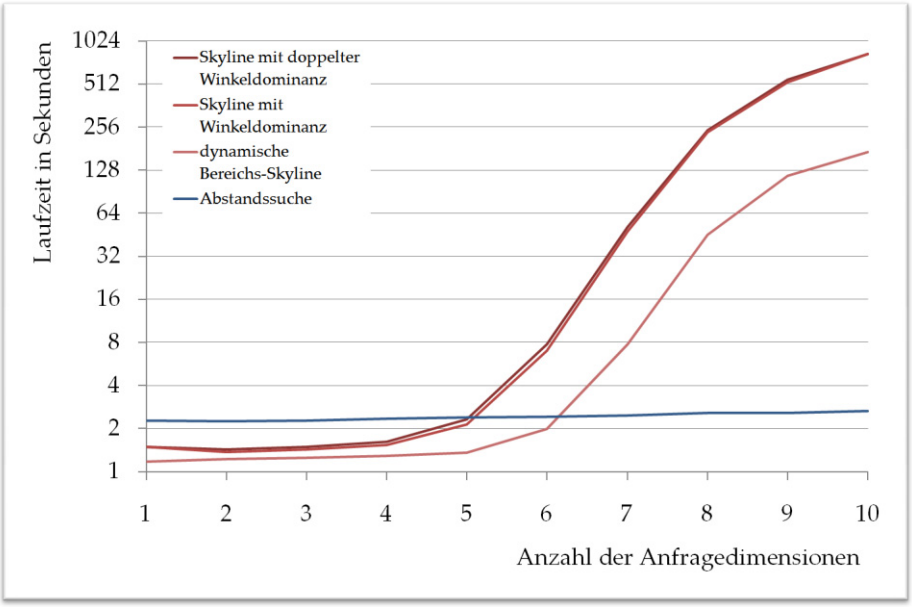

Abbildung 6.6: Laufzeitverhalten bei einer Tabelle mit 50.000 Datensätzen

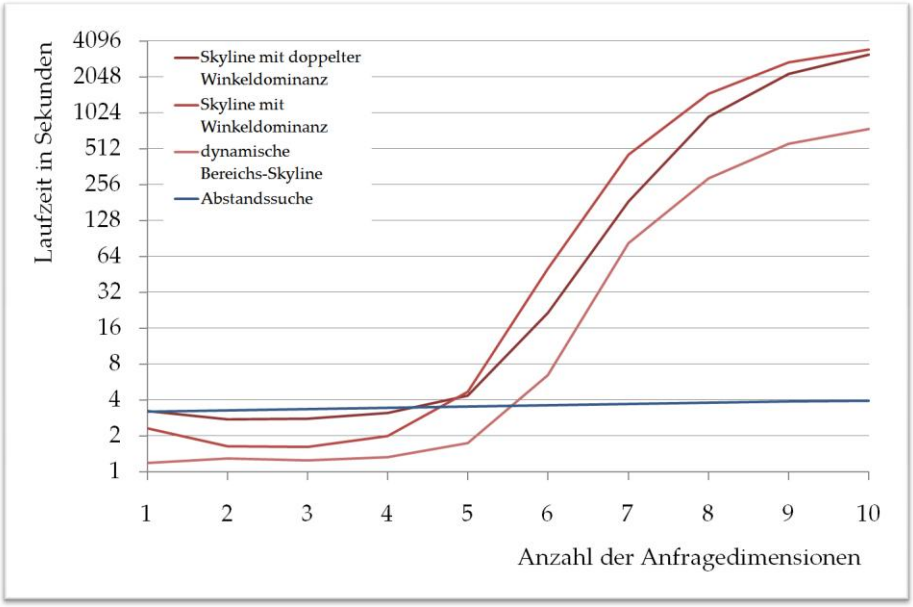

Abbildung 6.7: Laufzeitverhalten bei einer Tabelle mit 100.000 Datensätzen

Durchschnittlich hat die dynamische Bereichs-Skyline gegenüber den Winkeldominanzansätzen einen Laufzeitvorteil von 69,9%. Innerhalb der ersten vier Anfragedimensionen ist der durchschnittliche Laufzeitvorteil mit 32,4% bedeutend kleiner. Die Berechnung einer Skyline mit doppelter

Winkeldominanz dauert etwa 10% länger als die Berechnung der Skyline mit einfacher Winkeldominanz.

Vergleich der Ergebnisanzahl bei den Skyline-Varianten

Im Folgenden soll überprüft werden, ob die größere Vielfalt der Winkeldominanzansätze in der Praxis benötigt wird. Dazu wird die Ergebnisanzahl der verschiedenen Skyline-Varianten in Abbildung 6.8 gegenüber gestellt. Die Ergebnismenge besteht aus den Datensätzen erster und zweiter Klasse. Um das Beispiel stärker an die Aktordatenbank anzupassen, wurde eine Tabelle mit 1.000 Einträgen verwendet.

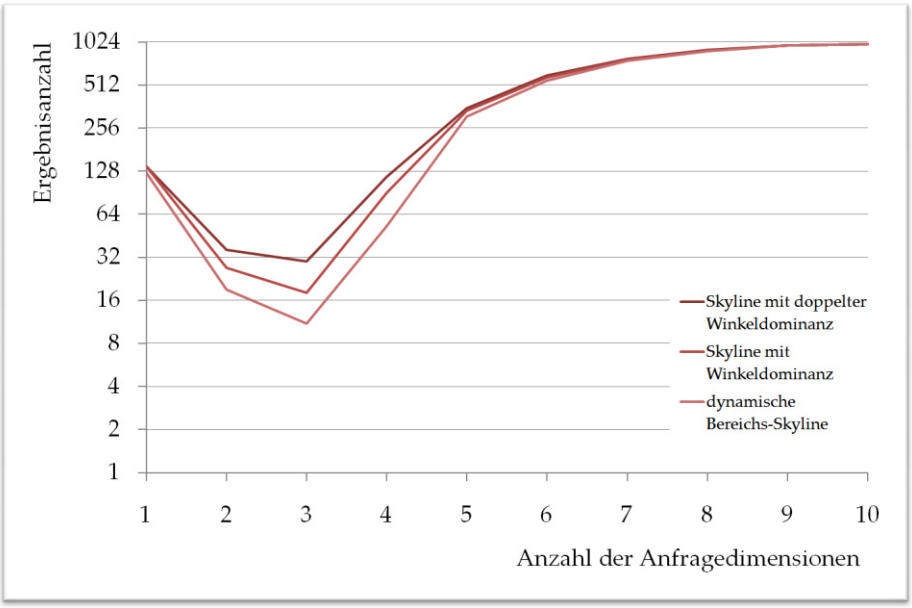

Abbildung 6.8: Ergebnisanzahl der einzelnen Skyline-Varianten

Bei zwei bis vier Anfragedimensionen liefern die Winkeldominanzansätze eine breitere Vielfalt, in den anderen Dimensionen liefert die dynamische Bereichs-Skyline bereits genügend Ergebnisse. Selbst bei einem Beispiel mit einer Tabelle von 10.000 Datensätzen sind bei drei Anfragedimensionen nur 16 Ergebnisse in der dynamischen Bereichs-Skyline.

7. Fazit

In dieser Arbeit sollte ein Suchalgorithmus für die Aktordatenbank entstehen. Dabei wurden auf der Basis des dynamischen Skyline-Ansatzes drei Skyline-Varianten entwickelt. Zusätzlich wurden zwei Algorithmen an die neu entwickelten Varianten angepasst.

Der Vergleich zwischen den beiden Algorithmen liefert ein klares Ergebnis: Der BNL-Algorithmus hat gegenüber dem D&C-Algorithmus eine deutlich bessere Laufzeit. Zusätzlich wird nur eine einzige Indexdatei verwendet, wodurch weniger Speicherplatz auf dem Datenträger benötigt wird. Ferner hat der BNL-Algorithmus dadurch einen Laufzeitvorteil bei Updates. Aus diesen Gründen sollte der BNL-Algorithmus bei den Skyline-Berechnungen in der Aktordatenbank verwendet werden.

Eine eindeutige Aussage dieser Art kann beim Vergleich der Skyline-Varianten nicht getroffen werden. Denn einerseits bietet die dynamische Bereichs-Skyline die beste Laufzeit andererseits liefern die Winkeldominanzansätze mehr Vielfalt bei den Ergebnissen. Werden jedoch mehr als fünf Anfragedimensionen verwendet, sollten die Winkeldominanzansätze nicht genutzt werden, da die dynamische Bereichs-Skyline sowohl eine ausreichend große Vielfalt als auch eine bessere Laufzeit bietet.

Das Skyline-Programm könnte auf verschiedene Arten erweitert werden:

- Statt der verwendeten Normierung des Gewichtsvektors könnte die Möglichkeit geschaffen werden, den Gewichtsvektor manuell anzulegen. Dadurch wäre es möglich, die Ergebnisse besser zu sortieren.

- Bei der Ausgabe der Winkeldominanzansätze könnten die Datensätze in vier Klassen eingeteilt werden. Dabei würden die Datensätze zweiter Klasse nochmals unterteilt: Zum einen in die Datensätze, die nur in der dynamischen Bereichs-Skyline enthalten sind und zum anderen in Datensätze, die nur durch die Winkeldominanz zur Skyline hinzugefügt wurden.
- Der Quellcode könnte überarbeitet werden, denn durch die schrittweise Entwicklung des Skyline-Programms, wurde dieser an manchen Stellen unübersichtlich.
- Bei großen Anfragedimensionen könnte die Ergebnisanzahl durch eine weitere Skyline-Variante reduziert werden.
- In der aktuellen Implementierung müssen die Updates der Indexdateien manuell gestartet werden. Durch eine entsprechende Änderung des Skyline-Programms könnten Veränderungen an den Tabellen automatisch erkannt und daraufhin die Indexdateien neu berechnet werden.

Literaturverzeichnis

[BKS01] S. Börzsönyi, D. Kossmann und K. Stocker. The Skyline operator. In Data Engineering, 2001.

[CJTTZ06] C. Chan, H. Jagadish, K. Tan, A. Tung und Z. Zhang. On High Dimensional Skylines. In EDBT ’06, 2006.

[CJTTZ‘06] C. Chan, H. Jagadish, K. Tan, A. Tung und Z. Zhang. Finding *k*-Dominant Skylines in High Dimensional Space. In SIGMOD ’06, 2006.

[DS07] E. Dellis und B. Seeger. Efficient Computation of Reverse Skyline Queries. In VLDB ’07, 2007.

[JHE04] W. Jin, J. Han und M. Ester. Mining Thick Skylines over Large Databases. In PKDD ’04, 2004.

[KEQS+08] E. Kallenbach, P. Eick; P. Quendt, T. Ströhla, K. Feindt, M. Kallenbach. Elektromagnete 3. Auflage. In Vieweg&Teubner, 2008.

[KLP75] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. In Journal of the ACM, Seiten 469-476, 1975.

[LYZZ05] X. Lin, Y. Yuan, Q. Zhang und Y. Zhang. Selecting Stars: The *k* Most Representative Skyline Operator. In ICDE ’05, 2005.

[PM05] A. N. Papadopoulos und Y. Manolopoulos. Nearest neighbor search. In Springer-Verlag, Seiten 25-34, 2005.

[PS85] F. P. Preparata und M. I. Shamos. Computational Geometry: An Introduction. In Springer-Verlag, 1985.

[PTFS03] D. Papadias, Y. Tao, G. Fu und B. Seeger. An optimal and progressive algorithm for skyline queries. In SIGMOD '03, 2003.

[PTFS05] D. Papadias, Y. Tao, G. Fu und B. Seeger. Progressive Skyline Computation in Database Systems. In ACM-TODS, 2005.

Thesen

1. Für die Aktordatenbank ist ein Suchalgorithmus gesucht, der dem Entwickler eine breite Vielfalt an Ergebnissen präsentiert.

2. Der Skyline-Ansatz liefert eine breitere Vielfalt an Ergebnissen als andere Standardsuchalgorithmen.

3. Der Skyline-Ansatz ist präferenzunabhängig.

4. Die bei der Aktordatenbank verwendeten Bereichsanfragen können nicht von dem Skyline-Basisansatz bearbeitet werden.

5. Die auf Grundlage der dynamischen Skyline entwickelte dynamische Bereichs-Skyline erweitert den Skyline-Ansatz um die Möglichkeit, Bereichsanfragen zu stellen.

6. Die entwickelten Winkeldominanzansätze sorgen für mehr Vielfalt bei Bereichsanfragen.

7. Beim D&C-Algorithmus sind im Gegensatz zum BNL-Algorithmus Anpassungen an die Winkeldominanzansätze nötig.

8. Durch die Verwendung von Indexdateien kann die Laufzeit der beiden Algorithmen verbessert werden.

9. Die entwickelten Skyline-Varianten und Algorithmen sind auch für die Suche in anderen Datenbanken einsetzbar.

Ilmenau, den 25. Mai 2009

Samuel Plentz