Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

Regelbasiertes Tiling für Collaborative Filtering

Diplomarbeit

Simone Daum

Betreuung durch
Prof. Dr. Johannes Fürnkranz

Juli 2005

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, Juli 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Einführung und Motivation

Traditionell werden Produkte oft von Mund-zu-Mund weiterempfohlen. Beim Griff ins Regal des Buchhändlers erinnert man sich gerne, dass ein Bekannter ein Buch sehr gelobt hat; und bei der Suche nach Fachliteratur verlässt man sich auf die Empfehlung des Händlers. Heute wird in Online-Shops zunehmend versucht, diese Mund-zu-Mund-Propaganda zu automatisieren. Nutzer können Bewertungen über Produkte abgeben und beim Collaborative Filtering werden diese Bewertungen dazu genutzt, mögliche Interessen von Nutzern zu vergleichen und Empfehlungen personalisiert für den Einzelnen zu generieren.

Das wohl zur Zeit prominenteste Beispiel für ein solches System ist der Online-Händler Amazon, der es sich zum Ziel gesetzt hat, das 'kundenorientierteste Unternehmen der Welt' zu sein [22]. Die Empfehlungen, die hier generiert werden, sollen exakt auf die Bedürfnisse und Interessen des Kunden zugeschnitten sein. Jeff Bezos (Präsident und CEO von Amazon.com) sagte dazu in einem Interview: „... if we have 20 million customers, then we should have 20 million stores: If you never buy romance novels, then we shouldn't clutter your view with them. If you like literary fiction, then we should tilt your store toward literary fiction." [23].

Ein System, das solche Empfehlungen generiert, wird Recommender-System genannt.

Collaborative Filtering und Recommender-Systeme

Unter Collaborative Filtering versteht man das Sammeln und Strukturieren von Meinungen und Bewertungen über Produkte oder Sachverhalte von verschiedenen Nutzern. Fast immer wird Collaborative Filtering mit Recommender-Systemen kombiniert, die die gesammelten Daten benutzen, um personalisierte Empfehlungen zu geben. So können beispielsweise einem Kunden gezielt Angebote unterbreitet werden, die ihn wahrscheinlich ansprechen werden, anstatt ihn mit pauschaler Werbung zu überhäufen. Anders als beim Content-Based Filtering, wo Empfehlungen aufgrund von Ähnlichkeiten der Produkte, also beispielsweise Bücher des gleichen Autors, ausgegeben werden, wird hier die Ähnlichkeit in den Profilen der Kunden ausgenutzt. Die Daten, die hierbei benutzt werden,

können dabei explizit gesammelt werden, indem Bewertungen, sogenannte Ratings, von Kunden bezüglich der Produkte abgegeben werden. Sie können aber auch implizit vom System gewonnen werden. Eine Kaufentscheidung ist eine indirekte Bewertung, aber auch allein die Dauer, die sich ein Nutzer mit bestimmten Webseiten beschäftigt, gibt Hinweise über seine Interessen. Unterschiedlich ist daher auch die Form, wie die Daten zur späteren Analyse vorliegen. Bewertungen können beispielsweise auf einer Skala von eins bis fünf erfolgen, oder für ein Paar aus einem Kunden und einem Produkt wird lediglich gespeichert, ob der Kunde das Produkt gekauft hat oder nicht. Diese sogenannten 0/1-Daten für eine Menge von Kunden und Produkten liegen auch dieser Arbeit zu Grunde. Bei Recommender-Systemen unterscheidet man das sogenannte Memory-Based Collaborative Filtering und das Model-Based Collaborative Filtering. Bei der ersten Variante wird zur Generierung einer Empfehlung für einen Kunden die gesamte Kundendatenbank herangezogen und alle enthaltenen Profile mit dem des Kunden verglichen, für den eine Empfehlung erstellt werden soll. Bei der modellbasierten Variante wird stattdessen zunächst ein Modell der Datenbank gebildet, aufgrund dessen dann eine Einordnung des Kunden erfolgt. Bei einem solchen Modell kann es sich zum Beispiel um ein neuronales Netz oder ein Clustering-Modell handeln. Der Vorteil eines Modells liegt darin, dass große Datenmengen besser bewältigt werden können. Allgemein ist auch die Dichte der Daten ein Problem bei Recommender-Systemen, da ein Kunde normalerweise nur einen sehr kleinen Bruchteil der Produkte bewertet hat. Hier kann ein Clustering-Modell sehr hilfreich sein. Um eine Art von Clustering-Modell handelt es sich auch bei Tilings, zu deren Erstellung hier ein neuer Ansatz vorgestellt werden soll.

Tiling

Bei einem Clustering-Modell handelt es sich um eine Einteilung der Kunden oder Produkte in Gruppen. Kunden können nach den Produkten, die sie bevorzugen, eingeteilt werden oder Produkte nach den Kunden gruppiert werden, die alle Produkte einer Gruppe mögen. Alle Mitglieder einer Gruppe sollten dabei möglichst große Ähnlichkeiten aufweisen. Zwei der bekanntesten Clustering-Algorithmen sind k-Means und Hierarchisches Clustering. Beide Verfahren ordnen ein Element exakt einem Cluster zu. Dies ist im Rahmen des Collaborative Filtering nicht immer sinnvoll, da es durchaus nützlich ist, einen Kunden mehreren Clustern zuordnen zu können. Dies gilt für Fälle, in denen ein Benutzerkonto von mehreren Kunden gleichzeitig, beispielsweise einer Familie, genutzt wird.

Ein Clustering kann auch benutzt werden, um eine Beschreibung der Datenbank zu erlangen. Dazu sind Cluster, die aus statistischen Verfahren entstehen, jedoch weniger geeignet, da sie erst näher analysiert werden müssen, um eine Beschreibung der Cluster zu erhalten. Deshalb werden hier Tiles eingesetzt, deren Inhalte schnell ersichtlich sind. Sie sind durch die enthaltenen Kunden und Produkte vollständig definiert, und es muss nicht, wie zum Beispiel bei k-Means, von einem Vektor abstrahiert werden, der das Zentrum des Clusters repräsentiert. Tiles gruppieren genauso wie Cluster Kunden und Produkte, die stark miteinander in Beziehung stehen. Beispielsweise hat jeder Kunde des Tiles alle Produkte des Tiles gekauft. Dies ist eine sehr starke Forderung, wie sie aber im verwandten Frequent Itemset Mining gestellt wird. Hier geht es darum, Mengen von Produkten zu finden, die von einer Mindestanzahl an Kunden gekauft wurden. Dabei muss ein Kunde alle Produkte einer Menge gekauft haben. Diese Forderung soll hier abgeschwächt werden, sodass der

Datenbankteil, der von einem Tile abgedeckt wird, nur noch einen gewissen Prozentsatz an Paaren von Kunden und gekauften Produkten enthalten muss. Diesen Prozentsatz nennt man die Dichte eines Tiles. Ein Tile sollte dabei möglichst viele Kunden und Produkte umfassen. Die Menge aller Tiles wird Tiling genannt.

In den meisten Anwendungen der Clustering-Verfahren erfolgt eine Gruppierung der Kunden anhand der Produkte, die sie bewertet oder gekauft haben, und Produkte werden anhand der Kunden gruppiert. In dem entwickelten Tiling soll jedoch versucht werden, Kunden und Produkte gleichzeitig zu berücksichtigen.

Regellernen

Die hier vorgestellte Methode zum Entwickeln eines Tilings orientiert sich an den Prinzipien des Regellernens. Regeln werden oft zur Klassifikation benutzt. Eine Regel kann zum Beispiel besagen, dass ein Objekt zur Klasse A gehört, wenn es die Eigenschaften X, Y und Z besitzt. Um diese Regeln zu lernen, wird auf einer Trainingsmenge von Beispielen gearbeitet. Dabei sollte eine erstellte Regel möglichst viele Beispiele abdecken und natürlich richtig klassifizieren. Im induktiven Top-Down-Regellernen wird mit einer Regel begonnen, deren Bedingungsteil leer ist und die somit alle Beispiele abgedeckt. Anschließend werden Bedingungen hinzugefügt, bis nur noch positive Beispiele überdeckt sind. Der Bottom-Up-Ansatz beginnt mit einer Regel, deren Bedingungsteil alle möglichen Bedingungen umfasst. Anschließend werden Bedingung herausgestrichen, sodass möglichst viele positive Beispiele überdeckt werden. Welche Bedingungen zum Einfügen oder Herausstreichen gewählt werden, wird durch Heuristiken gesteuert, die später näher vorgestellt werden. Da mit einer Regel normalerweise nicht die gesamte Trainingsmenge abgedeckt werden kann, müssen mehrere Regeln erzeugt werden. Dazu wird üblicherweise ein Separate-&-Conquer-Ansatz gewählt. Es wird eine erste Regel auf der Trainingsmenge gelernt und alle Beispiele, die von ihr abgedeckt werden, aus der Trainingsmenge entfernt. Auf der verbleibenden Beispielmenge wird eine neue Regel gelernt, etc., bis alle Beispiele durch Regeln abgedeckt wurden. Diese Vorgehensweisen wurden in dieser Arbeit auf einen neuen Ansatz zum Entwickeln von Tilings übertragen.

1.2 Ziel der Arbeit

Es soll ein Ansatz zum Erstellen eines Tilings für Collaborative Filtering Daten entwickelt werden. Die Daten liegen als 0/1-Daten vor. Die einzelnen Tiles umfassen Kunden und Produkte, die gehäuft zusammen auftreten. Sie müssen eine vorgegebene Mindestdichte einhalten und sollten dabei eine möglichst große Fläche aufweisen. Kunden und Produkte werden gleichzeitig zum Erstellen der Tiles herangezogen. Die enthaltenen Tiles sollten die gesamte Datenbank überdecken, wobei ein Überlappen der Tiles zulässig ist.

1.3 Gliederung

Kapitel 2 gibt zunächst eine genaue Definition der Problemstellung und der später verwendeten Begriffe. Außerdem werden die Testdatenbanken, die zur Evaluation verschiedener Parameter benutzt werden, skizziert.

Der regelbasierte Ansatz zum Erstellen eines Tilings wird in Kapitel 3 vorgestellt. Das Tiling wird sukzessive in einem Separate-&-Conquer-Ansatz aufgebaut. Es wird eine einfache Heuristik zum Finden eines minimalen Tilings beschrieben.

Wie einzelne Tiles erstellt werden, ist in Kapitel 4 gezeigt. Analog zum Lernen von Regeln wird ein Tile, ausgehend von einem Paar aus einem Kunden und dem gekauften Produkt, generalisiert, indem weitere Kunden und Produkte in das Tile eingefügt werden. Zur Auswahl des einzufügenden Kunden oder Produktes werden verschiedene Heuristiken vorgestellt, die möglichst große Tiles erzeugen sollen.

In Kapitel 5 wird der vorgestellte Algorithmus auf künstlichen Datenbanken sowie auf der MovieLens-Datenbank, einer im Internet frei verfügbaren Datenbank eines Recommender-Systems für Filme, evaluiert.

Ein kurzer Überblick über ähnliche Problemstellungen und Lösungsansätze aus der Literatur wird in Kapitel 6 gegeben.

Kapitel 7 fasst noch einmal die gewonnenen Ergebnisse zusammen und gibt einen Ausblick auf mögliche Ansatzpunkte zur Verbesserung des Algorithmus.

Kapitel 2

Problemdefinition und Testdatenbanken

Im Folgenden werden einige grundlegende Begriffe und Strukturen, wie Kunden- und Produktprofil, Tile und Tiling definiert und an Beispielen veranschaulicht. Das betrachtete Problem des Findens eines Tilings der Datenbank wird genau eingegrenzt. Anschließend wird der Aufbau der Testdatenbanken, die zur Evaluation der später eingeführten Heuristiken benutzt werden, vorgestellt.

2.1 Grundlagen

Sei $K = \{knr_0, knr_1, \ldots, knr_m\}$ eine Menge von Kundennummern und $P = \{pnr_0, pnr_1, \ldots, pnr_n\}$ eine Menge von Produktnummern, wobei $K \cap P = \emptyset$ gilt.
Ein Kunde ist durch seine Kundennummer sowie ein Kundenprofil gekennzeichnet, welches die Nummern aller Produkte enthält, die der Kunde gekauft oder bewertet hat.

Definition 1 (Kundenprofil, Kunde) *Sei $knr \in K$. Die Funktion k bildet die Kundennummer knr in die Menge der nicht-leeren Teilmengen von P ab, $k : K \to \mathfrak{P}(P) \setminus \emptyset$. Eine solche Teilmenge $k(knr) = k_{knr}$ von P nennt man* Kundenprofil. *Ein* Kunde *ist ein Tupel (knr, k_{knr}).*

Umgekehrt besteht jedes Produkt aus einer Produktnummer und seinem Produktprofil:

Definition 2 (Produktprofil, Produkt) *Sei $pnr \in P$. Die Funktion p bildet die Produktnummer pnr in die Menge der nicht-leeren Teilmengen von K ab, $p : P \to \mathfrak{P}(K) \setminus \emptyset$. Eine solche Teilmenge $p(pnr) = p_{pnr}$ von K nennt man* Produktprofil. *Ein* Produkt *ist ein Tupel (pnr, p_{pnr}).*

Kunden- und Produktprofile stehen in folgendem Zusammenhang:

$$\forall pnr \in P.\ p_{pnr} = \{knr \in K \mid pnr \in k_{knr}\}$$

bzw.

$$\forall knr \in K.\ k_{knr} = \{pnr \in P \mid knr \in p_{pnr}\}$$

Definition 3 (Positives/Negatives Kunden-/Produkt-Paar) *Ein Tupel* $(x, y) \in K \times P$ *heißt* positives Kunden-/Produkt-Paar *falls* $y \in k_x$*, ansonsten heißt es* negatives Kunden-/Produkt-Paar.

Die Datenbank besteht aus einer Menge von Kunden und Produkten. Sie lässt sich auch leicht als eine Matrix $D \in \{0,1\}^{|K|\times|P|}$ veranschaulichen. Die Zeilen repräsentieren hierbei die Kundenprofile, und die Produktprofile sind in den Spalten dargestellt:

$$\forall knr \in K, \forall pnr \in P.\ d_{knr,pnr} = 1 \leftrightarrow pnr \in k_{knr}$$

bzw.

$$\forall knr \in K, \forall pnr \in P.\ d_{knr,pnr} = 1 \leftrightarrow knr \in p_{pnr}$$

Alternativ könnte die Datenbank auch als Matrix $D \in \{0,1\}^{|P|\times|K|}$ mit Produktprofilen in Zeilen und Kundenprofilen in Spalten aufgefasst werden. Da für den entwickelten Algorithmus Kunden und Produkte ausdrücklich gleiche Relevanz haben, sollte dies bis auf zufällige Komponenten im Algorithmus keinen Unterschied hervorrufen. Im Folgenden wird die Matrix immer wie im ersten Fall beschrieben dargestellt. Der Begriff der Datenbank wird dabei als Synonym für die Matrix D benutzt. In Beispiel 2.1 ist eine Menge von Kunden und Produkten, deren Profile und die aus diesen entstehende Datenbank dargestellt.

Beispiel 2.1 K ist die Menge von Kundennummern, $K = \{1,2,3,4,5,6\}$. P bezeichnet die Menge von Produktnummern, $P = \{A,B,C,D,E,F\}$. Die Kundenprofile enthalten die Information, welcher Kunde welche Produkte gekauft hat.

$k_1 = \{B,C,E\}$
$k_2 = \{A,F\}$
$k_3 = \{A,B,C,E,F\}$
$k_4 = \{B,D\}$
$k_5 = \{A,F\}$
$k_6 = \{B,D\}$

} Kundenprofile

$p_A = \{2,3,5\}$
$p_B = \{1,3,4,6\}$
$p_C = \{1,3\}$
$p_D = \{4,6\}$
$p_E = \{1,3\}$
$p_F = \{2,3,5\}$

} Produktprofile

	A	B	C	D	E	F
1		X	X		X	
2	X					X
3	X	X	X		X	X
4		X		X		
5	X					X
6		X		X		

2.2 Tiles

Ein Tile ist eine Untermatrix der Matrix D, die durch eine Auswahl von Zeilen und Spalten, also Kunden- und Produktnummern, bestimmt wird:

Definition 4 (Tile) *Sei $K' \subseteq K, K' \neq \emptyset$, $P' \subseteq P, P' \neq \emptyset$, $D \in \{0,1\}^{|K|\times|P|}$. Dann ist ein Tile t definiert als*

$$t(K', P') \;=\; \{d_{knr,pnr} \mid knr \in K', pnr \in P'\}$$

Die Menge aller Tiles wird als $\mathcal{T}$ bezeichnet. Der Einfachheit halber wird ein Tile im Folgenden auch als Tupel aus einer Menge von Kundennummern und einer Menge von Produktnummern beschrieben: $t = (K', P')$.

Wird von Kunden- und Produktprofilen in Zusammenhang mit einem Tile t gesprochen, so reduziert sich die Menge der Produkt-/Kundennummern in einem Kunden- bzw. Produktprofil auf maximal jene Produkt-/Kundennummern, die im Tile t enthalten sind:

$$\hat{k}_{knr}(t(K', P')) = k_{knr} \cap P'$$

bzw.

$$\hat{p}_{pnr}(t(K', P')) = p_{pnr} \cap K'$$

Die Fläche φ eines Tiles t ist gleich der Anzahl seiner Elemente:

Definition 5 (Fläche φ) *Sei $t(K', P') \in \mathcal{T}$. Dann ist die Fläche $\varphi(t)$ gegeben durch:*

$$\varphi(t(K', P')) \;=\; |K'| \cdot |P'|$$

Die Dichte δ eines Tile bezeichnet den Anteil der positiven Kunde/Produkte-Paare an der Fläche des Tiles:

Definition 6 (Dichte δ) *Sei $t(K', P') \in \mathcal{T}$. Dann ist die Dichte $\delta(t)$ gegeben durch:*

$$\begin{aligned} \delta(t(K', P')) &= \frac{|\{d_{knr,pnr} | knr \in K', pnr \in P', d_{knr,pnr} = 1\}|}{\varphi(t(K', P'))} \\ &= \frac{\sum_{knr \in K', pnr \in P'} d_{knr,pnr}}{|K'| \cdot |P'|} \end{aligned}$$

Analog wird die Dichte $\hat{\delta}$ eines Kundenprofils $\hat{k}$ bzw. eines Produktprofil $\hat{p}$ in einem Tile berechnet als

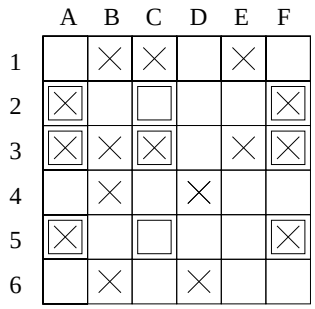
$$\hat{\delta}(\hat{k}_{knr}(t(K', P'))) \;=\; \frac{|\hat{k}_{knr}(t(K', P'))|}{|P'|}$$

bzw.

$$\hat{\delta}(\hat{p}_{pnr}(t(K', P'))) \;=\; \frac{|\hat{p}_{pnr}(t(K', P'))|}{|K'|}$$

Beispiel 2.2 zeigt ausführlich ein Tile in einer Datenbank, sowie die enthaltenen Kunden- und Produktprofile und deren Dichte.

Beispiel 2.2 Ein Beispiel für ein Tile in der Datenbank D aus Beispiel 2.1



Tile $t_1 = (\{2,3,5\}, \{A,C,F\})$
Die Kunden- und Produktprofile reduzieren sich auf:

$$\hat{k}_2(t_1) = \{A,F\} \qquad \hat{p}_A(t_1) = \{2,3,5\}$$
$$\hat{k}_3(t_1) = \{A,C,F\} \qquad \hat{p}_C(t_1) = \{3\}$$
$$\hat{k}_5(t_1) = \{A,F\} \qquad \hat{p}_F(t_1) = \{2,3,5\}$$
$$\hat{k}_1(t_1) = \hat{t}_4(t_1) \qquad \hat{p}_B(t_1) = \hat{p}_D(t_1)$$
$$= \hat{k}_6(t_1) = \emptyset \qquad = \hat{p}_E(t_1) = \emptyset$$

Dichte der Kundenprofile:
$\hat{\delta}(\hat{k}_2(t_1)) = \hat{\delta}(\hat{k}_5(t_1)) = \frac{2}{3}$, $\hat{\delta}(\hat{k}_3(t_1)) = 1$
Fläche des Tiles: $\varphi(t_1) = 9$
Dichte des Tiles: $\delta(t_1) = \frac{7}{9}$
Dichte der Produktprofile:
$\hat{\delta}(\hat{p}_A(t_1)) = \hat{\delta}(\hat{p}_F(t_1)) = 1$, $\hat{\delta}(\hat{p}_C(t_1)) = \frac{1}{3}$

Weitere Beispiele für Tiles sind $t_2 = (\{4,5\}, \{C,D,E,F\})$ mit $\varphi(t_2) = 8$, $\delta(t_2) = \frac{1}{4}$, $t_3 = (\{4,6\}, \{B,D\})$ mit $\varphi(t_3) = 4$, $\delta(t_3) = 1$ und $t_4 = (\{1\}, \{A\})$ mit $\varphi(t_4) = 1$, $\delta(t_4) = 0$.

Tiles sind invariant gegenüber der Anordnung der einzelnen Kunden- oder Produktprofile in der Matrix. Es spielt also zum Beispiel keine Rolle, ob ein Kundenprofil in der ersten oder dritten Zeile der Matrix eingetragen ist. Deshalb können Tiles auch einfach durch Permutation der Profile visualisiert werden. In Abbildung 2.1 wurden Kunden- und Produktprofile von D so permutiert, dass Tile t_1 besser sichtbar ist.

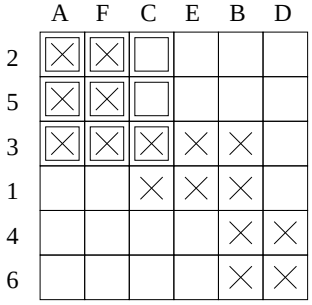


Abbildung 2.1: Permutation der Profile in Datenbank aus Beispiel 2.2

2.3 Tilings

Gegeben ist eine Dichte Δ. Ziel ist es, ein Tiling, also eine Menge von Tiles $T \subseteq \mathcal{T}$, zu finden, so dass die nachfolgenden Bedingungen möglichst gut erfüllt sind.

- Die einzelnen Tiles sollten eine möglichst große Fläche haben.
- Die Dichte der Tiles sollte möglichst hoch sein. Auf jeden Fall muss für jedes Tile t gelten, dass $\delta(t) \geq \Delta$.

- Die Datenbank sollte durch das Tiling möglichst gut überdeckt werden, d.h. dass möglichst jedes positive Kunden-/Produkt-Paar der Matrix D in einem Tile enthalten ist. Dabei dürfen sich die Tiles durchaus überschneiden.

Das resultierende Tiling liefert also eine Beschreibung der Datenbank. Ein Tile fasst mehrere Benutzer und Produkte zusammen, die eine starke Bindung haben, dadurch dass zum Beispiel die Kunden gleiche Produkte bewertet haben oder die Produkte gleiche Käufer haben. Durch die Berücksichtigung einer Dichte Δ des Tiles ist es nicht mehr notwendig, dass jeder Kunde alle Produkte des Tiles gekauft oder bewertet haben muss. Geerts, Goethals und Mielikäinen betrachten in 'Tiling Databases' [12] ebenfalls das Problem des Findens eines Tilings in einer Datenbank aus Sicht des Frequent Itemset Minings. Allerdings sind hier keine negativen Kunden-/Produkt-Paare innerhalb eines Tiles zulässig, wodurch einige Flexibilität verloren geht.
Es ist leicht ersichtlich, dass die Forderungen nach großer Fläche und hoher Dichte eines Tiles gegenläufig sind. Ein Tile mit hoher Dichte, aber nur kleiner Fläche, kann vergrößert werden, indem eine weitere Kunden- oder eine weitere Produktnummer hinzugenommen wird. Dadurch sinkt jedoch (wahrscheinlich) die Dichte. Im Extremfall wird eine Kundennummer knr zum Tile t hinzugenommen, deren Profil eine Dichte $\hat{\delta}(\hat{k}_{knr}(t))$ von Null aufweist, das also keinen weiteren Eintrag der Datenbank abdeckt. In Abbildung 2.2 kann das Tile $(\{2,3,4,5\},\{B,C,D\})$ bei $\Delta = 0,75$ zum Beispiel um die Produktnummer E erweitert werden.

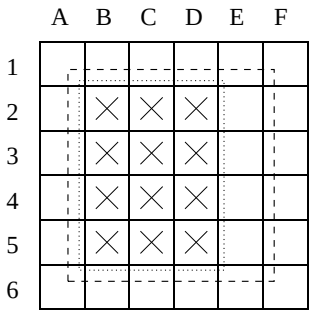


Abbildung 2.2: Trittbrettfahrer

Hier sinkt die Dichte des Tiles zugunsten der Fläche erheblich. Im Rahmen des Tilings ist dieses Phänomen unerwünscht, da so entstandene Tiles ein verzerrtes Abbild der Datenbank geben. Es tritt sehr ähnlich bei fehlertolerantem Frequent Itemset Mining auf und wird von Seppänen und Mannila, die sich mit dem Finden von 'dichten' Frequent Itemsets beschäftigen [16], als 'Trittbrettfahrer' bezeichnet.
Es gilt also, einen Mittelweg zwischen Fläche und Dichte zu finden, um solche Beispiele zu vermeiden.

In Beispiel 2.3 sind einige exemplarische Tilings der Datenbank aus Beispiel 2.2 zu sehen, die versuchen, obigen Anforderungen zu genügen.

Beispiel 2.3 Tilings

Δ	1	0,75	0,5
mögliches Tiling	$(\{1,3\},\{B,C,E\})$ $(\{2,3,5\},\{A,F\})$ $(\{4,6\},\{B,D\})$	$(\{1,3,4\},\{B,C,E\})$ $(\{2,3,5\},\{A,B,F\})$ $(\{1,3,4,6\},\{B,D\})$	$(\{1,3,4,6\},\{B,C,D,E\})$ $(\{2,3,5\},\{A,F\})$

2.4 Testdatenbanken

Der im Folgenden vorgestellte Algorithmus verwendet verschiedene Heuristiken für unterschiedliche Teilprobleme. Um für jedes Teilproblem die optimale Heuristik zu finden, werden Tests auf künstlichen Daten durchgeführt. Zur späteren Evaluation des gesamten Algorithmus werden ebenfalls künstliche Daten verwendet und zusätzlich als realer Datensatz ein Teil der MovieLens-Datenbank benutzt. Hier handelt es sich um Nutzungsdaten des Recommender-Systems MovieLens für Filme [25], die von dem GroupLens Research Projekt [24] zur Verfügung gestellt werden. In die künstlichen Datenbanken wurden Tiles eingestreut, deren Inhalte bekannt sind. So sind Testergebnisse leichter nachvollziehbar als auf realen Daten, indem zum Beispiel überprüft werden kann, wie viele der enthaltenen Tiles vom Algorithmus wiedergefunden werden. Zusätzlich ist es möglich, die Daten durch Einfügen von Rauschen zu stören, um die Auswirkung des Schwellenwertes Δ zu testen. Zum Erzeugen der Datenbanken wird die Anzahl der Kunden und Produkte sowie die Dichte der Datenbank, also das Verhältnis positiver zu negativer Kunden-/Produkt-Paare, vorgegeben. Außerdem wird die Anzahl der einzufügenden Tiles festgelegt. Aus Dichte der Datenbank und Anzahl der Tiles wird die Durchschnittsgröße eines Tiles errechnet. Die effektive Größe der einzelnen Tiles wird dann zufällig gewählt, sodass sich eine Normalverteilung um den Mittelwert mit einer Standardabweichung von 25% des Mittelwertes ergibt. Um die Anzahl der Kunden und Produkte in einem Tile t festzulegen, wird die Anzahl x der Kunden zufällig aus dem Intervall $1/2\sqrt{\varphi(t)} \leq x \leq 3/2\sqrt{\varphi(t)}$ gewählt. Die Anzahl der Kunden ergibt sich dann zu $\varphi(t)/x$. Kunden und Produkte selbst werden dann zufällig gewählt. Dadurch ergeben sich auch Überlappungen der Tiles. Zum Einfügen von Rauschen wird ein gewünschter Prozentsatz der Datenbankeinträge gewechselt.

Zum Ermitteln der geeignetsten Heuristiken werden 10 Datenbanken mit den gleichen Parametern benutzt, um eine Überanpassung an eine bestimmte Datenbank zu vermeiden. Sie enthalten jeweils 300 Kunden und Produkte. Bei einer Dichte von 10% sind 150 Cluster enthalten. Die Datenbanken enthalten kein Rauschen. Dieser Faktor wird erst in der späteren Evaluation betrachtet.

Kapitel 3

Regelbasiertes Tiling

Um ein Tiling der Datenbanken zu finden, wird ein Ansatz gewählt, der sich an Prinzipien des Regellernens orientiert. Nacheinander werden Tiles gebildet, die dann zusammen ein Tiling der Datenbank ergeben. Dazu wird jeweils ein Startpunkt, ein positives Kunden-/Produkt-Paar, ausgewählt, um das ein Tile inkrementell entwickelt wird. Wie das Bilden eines Tiles geschieht, wird in Kapitel 4 erläutert. Um eine Überdeckung der Datenbank zu erreichen, wird ein Separate-&-Conquer-Ansatz gewählt, wie er ähnlich im Regellernen üblich ist (siehe Abschnitt 3.2). Hierzu müssen gezielt Startpunkte ausgewählt werden, die Tiles in noch nicht überdeckten Bereichen der Datenbank entstehen lassen. Verschiedene Strategien zur Auswahl geeigneter Startpunkte werden in Abschnitt 3.3 vorgestellt und evaluiert. Da jedoch Überschneidungen der Tiles nicht zu vermeiden sind, ist auch die Frage nach der minimalen Anzahl an Tiles, die zur Überdeckung der Datenbank benötigt werden, von Interesse. Ein einfacher Ansatz zum Ermitteln dieses minimalen Tilings wird in 3.4 gezeigt.

3.1 Prinzipielle Vorgehensweise

In traditionellen Clustering-Verfahren, wie zum Beispiel k-Means, wird zu Beginn des Algorithmus eine feste Anzahl an Clustern vom Benutzer vorgegeben, und der Algorithmus bearbeitet alle Cluster gleichzeitig. In dem hier entwickelten Ansatz muss keine feste Anzahl von Tiles vorgegeben werden, sondern die Tiles werden einzeln nacheinander erstellt, wobei mit jedem Tile ein neuer Teil der Datenbank überdeckt wird. Es werden so lange Tiles hinzugefügt, bis die Datenbank komplett abgedeckt ist. Prinzipiell wäre es möglich, alle Tiles t, die dem Kriterium $\delta(t) > \Delta$ genügen, zu enumerieren, um aus diesen ein Tiling zu bilden. Eine solche Enumeration der gesuchten Strukturen wird zum Beispiel bei verwandten Problemen aus dem Assoziationsregellernen wie dem Finden von 'Dense Itemsets' [16] bzw. 'Fault-tolerant Itemsets' [15, 26] vorgenommen. Dies würde hier dazu führen, dass jedes einzelne positive Kunden-/Produkt-Paar der Datenbank als gültiges Tile gefunden wird. Im Rahmen des Tilings sind jedoch lediglich möglichst große Tiles von Interesse, und das Finden aller zulässigen Tiles ist sehr aufwendig. Deshalb wird in der vorgestellten Methode auf diese Enumeration verzichtet und stattdessen direkt mittels Heuristiken

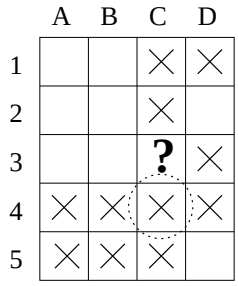


Abbildung 3.1: Gesucht ist das größte Tile um Punkt (4,C)

Δ	t	$\varphi(t)$	$\delta(t)$
1.0	({4,5},{A,B,C})	6	1.0
0.85	({4,5},{A,B,C,D})	8	0.875
0.75	({1,4,5},{A,B,C,D})	12	0.75
0.5	({1,2,3,4,5},{A,B,C,D})	20	0.55

Tabelle 3.1: Größtes Tile abhängig von Dichte

versucht, möglichst große und dicht besetzte Tiles zu finden, die die Charakteristika der Datenbank beschreiben.

Beim Erstellen eines Tiles wurde von folgender Fragestellung ausgegangen: ,,Wenn man einen einzelnen Punkt der Datenbank betrachtet, welches ist das größte Tile, das diesen Punkt beinhaltet?". Betrachtet man die Datenbank in Abbildung 3.1, so kann man zum Beispiel fragen, welches das größte Tile ist, das den Punkt (4,C) beinhaltet. Die Antwort darauf hängt von der vorgegebenen minimalen Dichte Δ ab. In Tabelle 3.1 sind die größten Tiles um den Punkt (4,C) abhängig von der Dichte dargestellt.

Mit dem Finden dieser größten Tiles um einen Punkt beschäftigt sich das nächste Kapitel. Dort wird ein Konzept zum inkrementellen Aufbau eines Tiles vorgestellt, das sich stark an das des induktiven Regellernens anlehnt. Ausgehend vom vorgegebenen Startpunkt wird das Tile immer weiter um Kunden- und Produktnummern vergrößert, bis keine Erweiterung mehr möglich ist. Dies entspricht dem Spezialisieren einer Regel durch Hinzunahme von Elementen in ihren Bedingungsteil im Regellernen. Wie beim Regellernen kann die Erweiterung nach verschiedenen Heuristiken erfolgen, von denen einige vorgestellt und evaluiert werden. Außerdem wird dort auf das bereits erwähnte Problem der Abwägung zwischen Dichte und Fläche eines Tiles eingegangen.

3.2 Separate-&-Conquer

Durch ein Tile wird ein Teil der Datenbank überdeckt. Nun gilt es, durch die Tiles eine komplette Überdeckung der Datenbank zu finden. Auch bei der Überdeckung lässt sich eine Analogie zum Regellernen herstellen. Hier liegt eine Menge von Beispielen als Trainingsmenge vor. Eine Regel deckt eine gewisse Untermenge dieser Beispiele ab. Es muss eine Menge von Regeln gefunden werden, die möglichst alle Beispiele abdeckt. Dazu

wird meist eine Separate-&-Conquer-Strategie verwendet. Dabei wird zu Beginn auf der Trainingsmenge eine erste Regel gelernt. Anschließend werden im Separate-Schritt alle Beispiele, die von dieser Regel abgedeckt werden, aus der Menge entfernt. Auf den verbleibenden Beispielen wird im Conquer-Schritt eine neue Regel gelernt. Die überdeckten Beispiele werden wieder entfernt etc., bis alle Beispiele abgedeckt wurden.

Beim Tiling wird, wie in Algorithmus 3.1 dargestellt, sehr ähnlich vorgegangen. Es wird ein Startpunkt gewählt und das größte Tile um diesen Punkt gesucht. Allerdings ist es nun nicht sinnvoll, alle abgedeckten positiven Kunden-/Produkt-Paare aus der Datenbank zu streichen (indem sie als negatives Paar gespeichert werden). Dadurch wäre es nicht mehr möglich, überlappende Tiles zu finden, da dann eines der überlappenden Tiles einen 'Trittbrettfahrer' enthalten müsste. Diese sind aber ausdrücklich nicht gewünscht. Dennoch muss gekennzeichnet werden, dass die überdeckten Paare schon 'benutzt' wurden. Sie müssen nicht noch einmal in einem anderen Tile enthalten sein. Danach wird ein neuer Startpunkt gewählt, der noch nicht überdeckt wurde und um diesen erneut ein Tile aufgebaut.

Algorithmus 3.1 Separate-&-Conquer-Covering

1: Markiere alle positiven Kunden-/Produkt-Paare als 'nicht überdeckt'
2: **while** Es gibt noch nicht überdeckte positive Paare **do**
3: Wähle nicht überdecktes positives Paar (knr,pnr)
4: Markiere (knr,pnr) als *überdeckt*
5: Bilde Tile t um (knr,pnr)
6: Markiere alle positiven Paare in t als *überdeckt*
7: **end while**

In Abbildung 3.2 - 3.4 ist ein Beispiel für das schrittweise Entstehen eines Tiling gegebenen. Als minimale Dichte ist $\Delta = \frac{2}{3}$ gegeben.

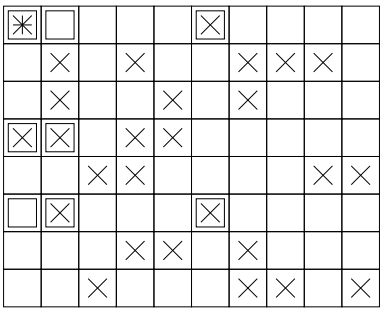


Abbildung 3.2: Der erste Startpunkt ist als Stern markiert. Auch das größte umliegende Tile ist bereits vorgezeichnet.

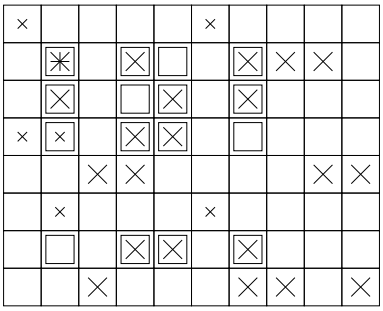


Abbildung 3.3: Die positiven Paare, die im Tile in 3.2 gefunden wurden, sind nun als überdeckt markiert. Sie werden nur noch durch ein kleines 'x' gekennzeichnet. Der nächste Startpunkt mit seinem größten Tile ist zu sehen.

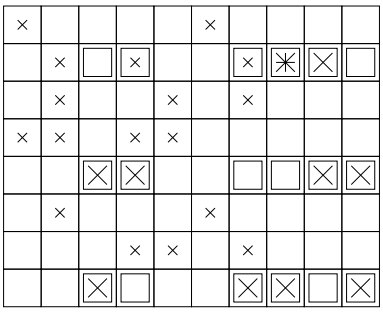


Abbildung 3.4: Die Paare aus 3.4 sind als überdeckt markiert. Mit dem Tile um den letzten Startpunkt ist die Datenbank komplett überdeckt. Sowohl in 3.3 als auch in 3.4 wurden bereits überdeckte Punkte für ein neues Tile verwendet.

3.3 Startpunkte

Um ein neues Tile aufzubauen, muss ein Startpunkt gewählt werden, der bisher noch nicht von anderen Tiles überdeckt wurde. Es stellt sich die Frage, welcher Punkt dazu herangezogen werden sollte. Von dieser Wahl der Startpunkte hängt das spätere Aussehen des Tilings ab. Es ist zum Beispiel möglich, dass um den einen Startpunkt ein Tile entsteht, das einen komplett neuen Teil der Datenbank abdeckt, während um einen anderen möglichen Startpunkt ein Tile entsteht, das eine große Überschneidung mit dem bereits überdeckten Teil der Datenbank aufweist. In welche Bereiche sich das Tile entwickelt, ist aber a priori nur anhand des Startpunktes nicht bekannt. Mit jedem neuen Tile sollten möglichst große Bereiche erfasst werden, die bisher noch nicht überdeckt waren. Gelingt dies, sollte insgesamt die Anzahl der Tiles, die zum Überdecken der gesamten Datenbank benötigt werden, möglichst gering sein.

3.3.1 Strategien

Es werden drei einfache Strategien zur Auswahl der Startpunkte betrachtet:

(SP1) Sequentielles Durchlaufen der Datenbank Zum Finden des nächsten Startpunktes wird die Datenbank in Form der Matrix zeilen- oder spaltenweise durchlaufen. D.h. dass ein Kunden- bzw. ein Produktprofil nach dem anderen abgearbeitet wird. Die Datenbank wird kundenweise durchlaufen (siehe Algorithmus 3.2). Weder bei einem kundenweisen noch produktweisen Durchlaufen sind Vorteile zu erwarten. Diese Strategie entspricht derjenigen, die im Tiling in Abbildung 3.2 - 3.4 angewendet wurde.

(SP2) Round-Robin auf Kunden- oder Produktprofile Um die Startpunkte möglichst breit über die Datenbank zu streuen, werden im Round-Robin-Verfahren alle Kundenprofile betrachtet und jeweils eine Produktnummer aus dem Kundenprofil gewählt, das mit der entsprechenden Kundennummer ein noch nicht überdecktes Paar bildet. Dieses Verfahren kann auch auf den Produktnummern durchgeführt werden, allerdings sind auch hier keine Vor- oder Nachteile zu erwarten. Die grobe Struktur dieses Verfahrens findet sich in Algorithmus 3.3.

Algorithmus 3.2 Sequentielles Durchlaufen der Datenbank

```
1: Markiere alle positiven Kunden-/Produkt-Paare als 'nicht überdeckt'
2: for all knr in K do
3:    for all pnr in k_knr do
4:       if (knr, pnr) noch nicht überdeckt then
5:          Wähle (knr, pnr) als nächsten Startpunkt
6:          Bilde Tile t um (knr/pnr)
7:          Markiere alle in t enthaltenen positiven Kunden-/Produkt-Paare als überdeckt
8:       end if
9:    end for
10: end for
```

Algorithmus 3.3 Round-Robin

```
1: Markiere alle positiven Kunden-/Produkt-Paare als 'nicht überdeckt'
2: while Es gibt noch nicht überdeckte positive Kunden-/Produkt-Paare do
3:    for all knr in K do
4:       if ∃pnr ∈ k_knr. (knr, pnr) nicht überdeckt then
5:          Wähle (knr, pnr) als nächsten Startpunkt
6:          Bilde Tile t um (knr/pnr)
7:          Markiere alle in t enthaltenen positiven Kunden-/Produkt-Paare als überdeckt
8:       end if
9:    end for
10: end while
```

(SP3) Kunden-/Produktprofil mit den meisten nicht überdeckten Elementen Mit einem neuen Tile sollen möglichst viele noch nicht überdeckte positive Kunden-/Produkt-Paare erfasst werden. Deshalb bietet es sich an, einen Startpunkt aus einem Kunden- oder Produktprofil zu wählen, das möglichst viele noch nicht überdeckte Produkt- bzw. Kundennummern enthält. In Algorithmus 3.4 ist eben dieses Prinzip dargestellt.

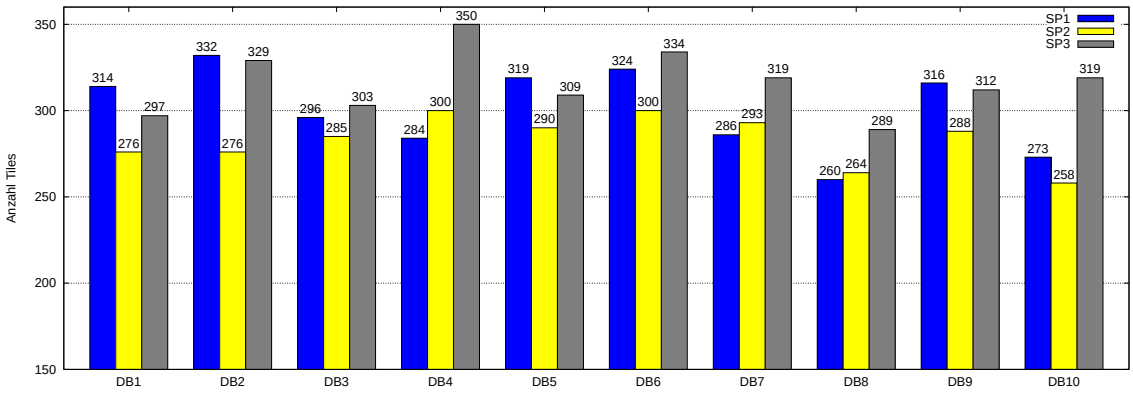
Algorithmus 3.4 Kunden-/Produktprofil mit den meisten nicht überdeckten Elementen

```
1: Markiere alle positiven Kunden-/Produkt-Paare als 'nicht überdeckt'
2: while Es gibt noch nicht überdeckte positive Kunden-/Produkt-Paare do
3:    Suche Profil x mit den meisten noch nicht überdeckten positiven Paaren
4:    if x ∈ P then
5:       Wähle knr ∈ p_x. (knr, x) nicht überdeckt. Startpunkt s = knr, x
6:    else
7:       Wähle pnr ∈ k_x. (x, pnr) nicht überdeckt. Startpunkt s = x, pnr
8:    end if
9:    Bilde Tile t um s
10:   Markiere alle in t enthaltenen positiven Kunden-/Produkt-Paare als überdeckt
11: end while
```

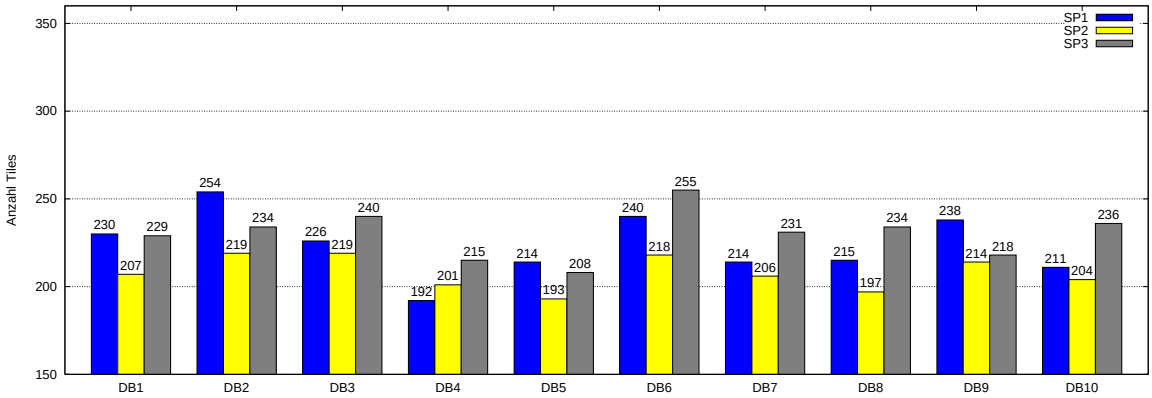
3.3.2 Evaluation

In Abschnitt 2.4 wurden bereits die Datenbanken beschrieben, die zur Evaluation der verschiedenen Parameter benutzt werden. In den 10 Testdatenbanken wurden jeweils 150 Tiles eingestreut. Die Anzahl der vom Algorithmus generierten Tiles hängt einerseits von der Auswahl der Startpunkte ab, andererseits natürlich besonders von der Art und Weise, wie einzelne Tiles gebildet werden. Verschiedene Heuristiken zum Finden des größten Tiles werden erst im folgenden Kapitel vorgestellt. Hier sollen zunächst die Auswirkungen der Wahl des Startpunktes betrachtet werden. Deshalb wird an dieser Stelle bereits vorgegriffen und eine Heuristik zum Finden einzelner Tiles festgelegt. Es handelt sich dabei um die Heuristik *Weighted Accuracy* (siehe Abschnitt 4.3.3) mit nachfolgender Präzisierung durch *maxKand* (siehe Abschnitt 4.3.4). Werden weniger Tiles generiert, so ist es möglich, dass entweder die generierten Tiles eine größere Fläche haben und so prinzipiell mehr positive Kunden-/Produkt-Paare abdecken, oder die Tiles weniger Überlappungen aufweisen.

Die Fläche der Tiles wird hauptsächlich von den verwendeten Heuristiken bei der Tilebil-



(a) $\Delta = 1,0$



(b) $\Delta = 0,75$

Abbildung 3.5: Anzahl generierter Tiles bei unterschiedlichen Strategien zur Auswahl des Startpunktes

dung gesteuert. Da diese für die Strategien SP1-SP3 fix sind, wird also hier hauptsächlich der Grad der Überlappungen gemessen. In Abbildung 3.5(a) und 3.5(b) ist die Anzahl der generierten Tiles unter Verwendung der jeweiligen Strategie dargestellt. Es lässt sich erkennen, dass das Round-Robin-Verfahren (SP2) auf fast allen Datenbanken, unabhängig vom gewählten Schwellenwert Δ, die wenigsten Tiles erzeugt. Prinzipiell werden weitaus mehr Tiles erzeugt, als ursprünglich in die Datenbank eingestreut wurden. Dies resultiert aus großen Überschneidungen der Tiles untereinander, wie im nächsten Abschnitt 3.4 deutlich wird. Entgegen den Erwartungen erzeugt die Heuristik, die die Anzahl der nicht überdeckten Elemente in den Profilen berücksichtigt, oft am meisten Tiles. Das Tile, das für ein nicht überdecktes positives Paar des gewählten Profils erzeugt wird, deckt viele Elemente dieses Profils ab, wobei aber oft einige wenige Elemente des Profils unüberdeckt bleiben. Für die verbleibenden Elemente muss dann oft ein eigenes Tile erzeugt werden, da sie sich nicht in einem Tile sinnvoll zusammenfassen lassen. Im Folgenden wird also zur Auswahl der Startpunkte das Round-Robin-Verfahren gewählt.

3.4 Minimales Tiling

Ein Tiling liefert eine Beschreibung der Datenbank, indem charakteristische Häufungspunkte der positiven Kunden-/Produkt-Paare in Tiles zusammengefasst werden. Obwohl zum Erstellen eines neuen Tiles immer ein noch nicht überdecktes Paar benutzt wird, kommt es häufig vor, dass ein um diesen Startpunkt gebildetes Tile eine große Überschneidung mit anderen Tiles aufweist und auch der Startpunkt später von einem weiteren Tile abgedeckt wird. Deshalb sind oft viele Tiles eines Tilings redundant in dem Sinne, dass sie nicht benötigt werden, um die Datenbank abzudecken. Sie beinhalten dennoch wertvolle Informationen zum Beispiel über spezielle Kombinationen von Produkten, die gleiche Käufer haben. Genauso kann aber auch für eine kompaktere Beschreibung eine Überdeckung der Datenbank mit möglichst wenigen Tiles interessant sein. Dazu wird aus den vorhandenen Tiles die minimale Menge herausgenommen, die die Datenbank überdeckt. In der Literatur wird diese Problemstellung als Minimum-Set-Cover bezeichnet. In Algorithmus 3.5 ist eine einfache Heuristik zum Finden eines möglichst kleinen Tilings aus einem gegebenen Tiling gezeigt, die versucht, möglichst große Tiles in das minimale Tiling einzubeziehen.

Algorithmus 3.5 Finden eines minimalen Tilings

```
1: INPUT:  T                    // Ausgangs-Tiling
2: OUTPUT: T'                   // minimales Tiling
3: Sortiere T aufsteigend nach Fläche der Tiles
4: for all t ∈ T do
5:   if t enthält positives K.-/P.-Paar, das in keinem anderen Tile enthalten ist then
6:     T' = T' ∪ {t}
7:   else
8:     T = T \ {t}
9:   end if
10: end for
```

Für jedes Tile wird geprüft, ob es ein positives Kunden-/Produkt-Paar enthält, das von keinem anderen Tile abgedeckt wird. Dann muss das Tile zum minimalen Tiling gehören. Ansonsten wird das Tile gelöscht. Kleine Tiles werden zuerst überprüft; dadurch werden sie gelöscht, sobald noch ein oder mehrere größeres Tiles vorhanden sind, die die gleiche Fläche abdecken.

In den Tabellen 3.2 und 3.3 ist die Anzahl der benötigten Tiles im minimalen Tiling für die Tilings aus Abschnitt 3.3 im Round-Robin-Verfahren (SP2) gezeigt. Hier wird deutlich, dass weitaus weniger Tiles zum Überdecken der Datenbanken benötigt werden, als vom Algorithmus erzeugt werden.

	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8	DB9	DB10
Anzahl erzeugter Tiles	276	276	285	300	290	300	293	264	288	258
Anzahl Tiles im min. Tiling	164	177	163	164	181	185	168	169	170	166

Tabelle 3.2: Anzahl Tiles im minimalen Tiling für $\Delta = 1,0$

	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8	DB9	DB10
Anzahl erzeugter Tiles	207	219	219	201	193	218	206	197	214	204
Anzahl Tiles im min. Tiling	150	150	152	150	150	157	152	150	163	150

Tabelle 3.3: Anzahl Tiles im minimalen Tiling für $\Delta = 0,75$

Kapitel 4

Finden eines Tiles

Im letzten Kapitel wurde gezeigt, wie ein Tiling mittels einer Separate-&-Conquer-Strategie aus Tiles gebildet wird. Nun soll gezeigt werden, wie ein einzelnes Tile entsteht. Dazu wird ausgehend von einem Tile, das zunächst nur aus dem Startpunkt besteht, dieses Tile generalisiert, indem nacheinander Kunden oder Produkte eingefügt werden. Die Auswahl der Kunden und Produkte, die zur Generalisierung benutzt werden, erfolgt nach verschiedenen Heuristiken, die in Abschnitt 4.3 vorgestellt werden. Zuvor wird jedoch in Abschnitt 4.2 beschrieben, wie durch das Aufstellen einer Dichteanforderung für im Tile enthaltene Profile ein Gleichgewicht zwischen Dichte und Fläche des Tiles hergestellt werden kann. Da diese Forderung sehr strikt ist, wird in Abschnitt 4.6 nochmals eine Relaxierung dieser Dichteanforderung betrachtet. Welche Kunden oder Produkte als Kandidaten zum Einfügen zur Verfügung stehen, wird in Abschnitt 4.4 erläutert. Da die Anzahl der Kandidaten zu Beginn der Generalisierungsphase relativ hoch ist, wird zudem in Abschnitt 4.7 ein Verfahren vorgestellt, das durch Benutzen größerer Starttiles anstelle von Startpunkten die Anzahl der Kandidaten verringert. Abschließend werden in Abschnitt 4.8 zwei Suchstrategien zum Finden des größten Tiles vorgestellt.

4.1 Generalisierung

Beim Top-Down Regellernen (hier für ein Klassifikationsproblem mit zwei Klassen, positiv und negativ) wird mit einer leeren Regel gestartet und diese dann spezialisiert, d.h. nacheinander werden Bedingungen in den Körper der Regel eingefügt, so dass möglichst viele positive und möglichst wenige negative Beispiele von der Regel abgedeckt werden.

Entsprechend soll nun hier für ein gegebenes Paar (knr_x,pnr_y) vorgegangen werden:

- Starte mit einem Tile, das lediglich den Startpunkt (knr_x,pnr_y) enthält, $t = (\{knr_x\}, \{pnr_y\})$
- Füge nacheinander Kunden- und Produktnummern hinzu, sodass ein möglichst großes und dichtes Tile entsteht.

Durch das Einfügen von Kunden und Produkten in das Tile wird dieses jedoch nicht spezialisiert, sondern generalisiert, da mehr Punkte der Datenbank abgedeckt werden. Es

handelt sich hier also um einen Bottom-Up-Ansatz zum Bilden von Tiles. Beim Regellernen muss entschieden werden, welche Bedingung jeweils als nächste in den Körper der Regel eingefügt wird. Genauso muss bei dem Erstellen eines Tiles die nächste Kunden- bzw. die nächste Produktnummer gewählt werden, welche dem Tile hinzugefügt wird. Dazu können verschiedene Heuristiken verwendet werden. Diese werden im Abschnitt 4.3 vorgestellt. Nicht alle Kunden- und Produktnummern kommen in Frage, um in das Tile eingefügt zu werden. Dazu gehören zum Beispiel solche, die zu dem Phänomen der Trittbrettfahrer gehören (siehe Abbildung 2.2). Diejenigen, die in Frage kommen, werden Kandidaten genannt. Sie werden während des Tileaufbaus nach jedem Generalisierungsschritt gepflegt, d.h. es kommen neue Kandidaten hinzu und gegebenenfalls müssen andere Kundennummern oder Produktnummern von der Kandidatenliste gestrichen werden. In Algorithmus 4.1 ist der Generalisierungsablauf noch einmal dargestellt.

Algorithmus 4.1 Generalisierung

1: $t = (\{knr_x\}, \{pnr_y\})$
2: Initialisiere Kandidatenliste
3: Finde nächsten Kandidaten
4: **while** $\delta(t) \geq \Delta$ & Kandidat gefunden **do**
5: Füge Kandidaten in Tile ein
6: Passe Kandidatenliste an
7: **end while**
8: Führe Nachbearbeitung durch

Wie das Pflegen der Kandidatenliste genau geschieht und welche Kunden- und Produktnummern als Kandidaten zum Einfügen in Frage kommen, wird im Abschnitt 4.4 erläutert. Im Nachbearbeitungsschritt wird noch einmal die Dichte der Kunden- und Produktprofile im Tile überprüft. Mehr dazu findet sich jedoch im nächsten Abschnitt, in dem zunächst noch einmal auf das Problem eingegangen wird, wie ein Gleichgewicht zwischen Dichte und Fläche eines Tiles hergestellt werden kann.

4.2 Gleichgewicht zwischen Dichte und Fläche

Es ist gefordert, dass Tiles einerseits eine möglichst große Fläche haben, diese aber andererseits nicht zu stark auf Kosten der Dichte erreicht wird, indem zum Beispiel für das Tile leere Kundenprofile hinzugenommen werden. Noch einmal wird das Beispiel aus Abbildung 3.1 betrachtet. Mit kleiner werdender Dichte steigt die Fläche des Tiles um den gegebenen Punkt an. Beim Betrachten des größten Tiles ({1,2,3,4,5},{A,B,C,D}) fällt auf, dass die Gesamtdichte zwar über $\Delta = 50\%$ liegt, allerdings stellt sich die Frage, ob dies wirklich ein 'sinnvolles' Tile darstellt. Die Kundenprofile mit den Nummern 2 und 3 haben beide nur eine Dichte von 25%, bringen also aus Sichtweise der Matrix nur eine weitere Eins in das Tile ein. Für dieses kleine Beispiel kann diskutiert werden, ob diese Kundennummern zum Tile gehören sollten oder nicht. In größeren Beispielen oder in einem Extremfall wie in Beispiel 2.2, sollten solche schon als 'Trittbrettfahrer' vorgestellten Kunden- oder Produktprofile vermieden werden.

Um diese Ausgewogenheit zwischen Dichte und Fläche zu gewährleisten, wird folgende *Dichteanforderung* gestellt:

> *Die Dichte $\hat{\delta}(t)$ aller in einem Tile t enthaltenen Kunden- und Produktprofile muss mindestens Δ betragen!*

Je geringer Δ gewählt wird, desto mehr negative Kunden-/Produkt-Paare können mit einem Kunden- bzw. Produktprofil einhergehen. Bei großem Δ und flächenmäßig kleinen Tiles werden also wohl fast vollständige Tiles zu erwarten sein, wohingegen bei größeren Tiles anteilig mehr negative Paare auftreten können. Prinzipiell wird aber die Gesamtdichte δ des Tiles deutlich über Δ bleiben. Für das Beispiel in Abbildung 3.1 ergeben sich dann als neue größte Tiles die in Tabelle 4.1 gezeigten.

Es stellt sich die Frage, ob diese Forderung zu restriktiv ist. Betrachtet man beispielsweise ein Tile wie in Abbildung 4.1 mit $\Delta = 0,9$, so ist dieses Tile mit dieser Forderung nicht zulässig, obwohl die Gesamtdichte δ ebenfalls $0,9$ beträgt. Die Dichte $\hat{\delta}$ der Produktprofile A-I beträgt jedoch nur $0,88$.

Um dieses Phänomen in den Griff zu bekommen, kann jedoch einfach der Parameter Δ variiert werden. Er dient zwar prinzipiell als fester Eingabewert, bestimmt aber maßgeblich die Struktur der Tiles und muss so für jede Datenbank neu optimiert werden, um eine gute Beschreibung der Datenbank zu erhalten. In [16] wird ein ähnlicher Ansatz gewählt. Hier wird gefordert, dass alle Teilmengen eines Itemsets bei gegebenem Support eine Mindestdichte haben. Allerdings wird hierbei die Symmetrie der Kunden- und Produktnummern nicht beachtet, so dass für Kundenprofile nicht zwangsläufig eine gewisse Mindestdichte vorliegt. Bei einem Support von 9 hat das Tile bzw. Itemset in Abbildung 4.1 ebenfalls nicht genügend hohe Dichte. Alternativ zur Änderung des Parameters Δ wird in Abschnitt 4.6 ein Ansatz mit einer leicht relaxierten Dichteanforderung betrachtet.

Δ	t	$\varphi()$	$\delta(t)$
1,0	({4,5},{A,B,C})	6	1,0
0,85	({4,5},{A,B,C})	6	1,0
0,75	({4,5},{A,B,C})	6	1,0
0,5	({1,4,5},{A,B,C,D})	12	0,75

Tabelle 4.1: Größte Tiles mit $\hat{\delta} \geq \Delta$.

	A	B	C	D	E	F	G	H	I	J
1		×	×	×	×	×	×	×	×	×
2	×		×	×	×	×	×	×	×	×
3	×	×		×	×	×	×	×	×	×
4	×	×	×		×	×	×	×	×	×
5	×	×	×	×		×	×	×	×	×
6	×	×	×	×	×		×	×	×	×
7	×	×	×	×	×	×		×	×	×
8	×	×	×	×	×	×	×		×	×
9	×	×	×	×	×	×	×	×		×

Abbildung 4.1: Für $\Delta = 0,9$ kein zulässiges Tile

Die Überprüfung der Dichte der im Tile enthaltenen Kunden- und Produktprofile erfolgt nach der Generalisierungsphase. Während des Tileaufbaus können Kunden- oder Produktprofile, die zuvor genügend hohe Dichte hatten, durch das Einfügen anderer Produkt- oder Kundennummern wieder ungültig geworden sein. Diese müssen deshalb in einer Nachbearbeitungsphase aus dem Tile herausgestrichen werden. Ein permanentes Überprüfen der Dichte aller Elemente könnte prinzipiell auch während der Generalisierungsphase bei jedem Einfügen erfolgen. Dies kostet jedoch zusätzliche Rechenzeit; vor allem können mögliche bessere Lösungen verloren gehen, dadurch dass Kunden-/Produktprofile durch das Einfügen anderer Elemente abwechselnd genügende, unzureichende aber auch eventuell wieder genügende Dichte haben. Allgemein lässt sich bei dieser Problemstellung sagen, dass sich die Dichte der Elemente im Tile, des Tiles selbst sowie die Heuristikwerte der Kandidaten weder monoton noch antimonoton verhalten. Dadurch kann zum Beispiel ein Kandidat, der einmal zu geringe Dichte aufweist, nicht für immer verworfen werden. Genauso kann wie oben beschrieben nicht davon ausgegangen werden, dass ein schon eingefügtes Element immer genügend Dichte aufweisen wird.

Im Nachbearbeitungsschritt genügt es nicht, die Kunden- und Produktnummern einmal auf ihre Dichte zu prüfen und gegebenenfalls bei zu geringer Dichte aus dem Tile herauszunehmen. Durch das Herausnehmen einer Produktnummer ändert sich die Dichte der Kundenprofile und andersherum. Deshalb wird auch hier schrittweise vorgegangen. Es wird jeweils das nächste Element mit der niedrigsten Dichte $\hat{\delta}(\hat{k})$ oder $\hat{\delta}(\hat{p})$ und $\hat{\delta} < \Delta$ gesucht und aus dem Tile entfernt, bis alle Kunden- und Produktprofile die Mindestdichte erreicht haben. Weisen sowohl ein Kunden- als auch eine Produktprofil die gleiche niedrigste Dichte kleiner Δ auf, so wird jenes Profil aus dem Tile entfernt, welches weniger Kunden- bzw. Produktnummern enthält, da so die Fläche des Tiles am größten bleibt. Dabei kann es auch vorkommen, dass die Kunden- oder Produktnummer aus dem Startpunkt verloren gehen. Durch vorausschauende Auswahl der Profile, die in das Tile eingefügt werden (siehe Abschnitt 4.4), wird dies weitestgehend vermieden. Auf jeden Fall wird der Startpunkt als *überdeckt* markiert, da bei dem vorgestellten deterministischen Algorithmus immer wieder das gleiche Tile gefunden wird und deshalb immer wieder der Punkt als Startpunkt gewählt werden müsste. Außerdem ist es meistens der Fall, dass dieser Startpunkt von einem anderen Tile abgedeckt wird. Der Prozentsatz der positiven Kunden-/Produkt-Paare, die durch diesen Effekt nicht abgedeckt werden, ist sehr gering. Bei der späteren Evaluation wird ihr Prozentsatz unter allen positiven Paaren kurz betrachtet werden.

4.3 Heuristiken

Aus dem Regellernen sind verschieden Heuristiken zur Wahl der nächsten in die Regel einzufügenden Bedingung bekannt. Diese können auch zur Generalisierung beim Tileaufbau verwendet werden.

4.3.1 Heuristiken aus dem Regellernen

Es sollen Regeln gelernt werden, die eine Klassifikation von Beispielen in zwei Klassen, Positiv und Negativ, vornehmen. Dazu steht eine Trainingsmenge M von bereits klassifizierten Beispielen zur Verfügung. Sei N^+ die Anzahl aller positiven Beispiel in M und

N^- die Anzahl aller negative Beispiele in M. Die Anzahl der positiven bzw. negativen Beispiele, die von einer Regel r abgedeckt werden, wird als n_r^+ bzw. n_r^- bezeichnet. Die Heuristiken bewerten eine Regel r in Abhängigkeit von n_r^+ und n_r^-.

Zwei Heuristiken lassen sich als *äquivalent* bezeichnen, wenn sie die gleiche Regel erzeugen würden, d.h. $heur_1(r_1) > heur_1(r_2) \Leftrightarrow heur_2(r_1) > heur_2(r_2)$.

Mit einem Maß *heur* lässt sich auch eine einzelne Bedingung b bei einer gegebenen Regel r als $heur(b, r)$ anhand der durch die Bedingung zusätzlich abgedeckten positiven und negativen Beispiele bewerten. Für das Einfügen einer Bedingung b in eine Regel r wird kurz $r + b$ notiert. Ein Maß ist *additiv*, wenn gilt $heur(r + b) = heur(r) + heur(b, r)$.

Precision

Die Precision berechnet den Anteil der positiven Beispiele unter allen durch die Regel abgedeckten Beispielen:

$$prec(r) = prec(n_r^+, n_r^-) = \frac{n_r^+}{n_r^+ + n_r^-}$$

Accuracy

Accuracy berechnet den Anteil aller durch die Regel korrekt klassifizierten Beispiele, also den Anteil der abgedeckten positiven Beispiele und der nicht abgedeckten negativen Beispiele:

$$acc(r) = acc(n_r^+, n_r^-) = \frac{n_r^+ + (N^- - n_r^-)}{N^+ + N^-}$$

Da N^+ und N^- konstant sind, lässt sich auch äquivalent schreiben:

$$acc(r) = acc(n_r^+, n_r^-) \simeq n_r^+ - n_r^-$$

Weighted Relative Accuracy

Weighted Relative Accuracy (WRA) setzt die Genauigkeit der Regel in Relation zur Genauigkeit der Defaultregel, die einfach alle Beispiele positiv klassifiziert. Zusätzlich wird diese Relation mit dem Anteil der abgedeckten Beispiele gewichtet:

$$wra(r) = wra(n_r^+, n_r^-) = \frac{n_r^+ + n_r^-}{N^+ + N^-}\left(\frac{n_r^+}{n_r^+ + n_r^-} - \frac{N^+}{N^+ + N^-}\right)$$

Auch dies lässt sich wieder durch Multiplikation und Division mit den Konstanten N^+ und N^- äquivalent umformen zu:

$$wra(r) = wra(n_r^+, n_r^-) \simeq \frac{n_r^+}{N^+} - \frac{n_r^-}{N^-}$$

Accuracy und Weighted Relative Accuracy sind zwei mögliche Ausprägungen einer allgemeinen linearen Kostenfunktion $n_r^+ \cdot c - n_r^- \cdot (1-c)$, wobei $c_{acc} = 0,5$ und $c_{wra} = N^-/N^+ + N^-$. Dies lässt sich so interpretieren, dass es genauso gut ist, ein positives Beispiel abzudecken wie nicht $c/(1-c)$ negative abzudecken.

Genaueres zu diesen Heuristiken sowie andere lineare und nicht-lineare Metriken finden sich zum Beispiel in [9, 10, 8]. Nun soll gezeigt werden, wie sich diese Heuristiken zum Tileaufbau verwenden lassen.

4.3.2 Precision

Im Regellernen werden zum Berechnen der Heuristiken die positiven und negativen Beispiele gezählt, die von der Regel abgedeckt werden bzw. in der gesamten Trainingsmenge vorhanden sind. Eine Regel wird nun durch ein Tile repräsentiert und statt positiven und negativen Beispielen werden positive und negative Kunden-/Produkt-Paare herangezogen. Sei $n^+(t)$ also nun die Anzahl der positiven Kunden-/Produkt-Paare und $n^-(t)$ die Anzahl der negativen Kunden-/Produkt-Paare im Tile t aus den Kunden K' und Produkten P'. Dann gilt hier:

$$prec(t(K',P')) = \frac{n^+(t)}{n^+(t)+n^-(t)} = \frac{\sum_{knr \in K'} |\hat{k}_{knr}(t)|}{\varphi(t(K',P'))} = \delta(t(K',P'))$$

Die Precision entspricht also der Dichte des Tiles. Bei der Suche nach der nächsten einzufügenden Kunden- oder Produktnummer wird diejenige gewählt, welche die Precision nach Einfügen in das Tile maximiert. Für das Einfügen einer Kunden- oder Produktnummer x wird abkürzend geschrieben:

$$t(K',P') + x = \begin{cases} t(K' \cup \{x\}, P') & \text{für } x \in K \\ t(K', P' \cup \{x\}) & \text{für } x \in P \end{cases}$$

Es genügt, die Dichte des entsprechenden Kundenprofils $\hat{k}$ bzw. des Produktprofil $\hat{p}$ zu betrachten:

Die Anzahl $n^+(t)$ bzw. $n^-(t)$ der positiven bzw. negativen Kunden-/Produkt-Paare lässt sich analog für Kunden- und Produktnummern definieren:

$$\begin{aligned} n^+_{knr}(t(K',P')) &= |\hat{k}_{knr}(t(K',P'))| \\ n^-_{knr}(t(K',P')) &= |P' \setminus \hat{k}_{knr}(t(K',P'))| \\ n^+_{pnr}(t(K',P')) &= |\hat{p}_{pnr}(t(K',P'))| \\ n^-_{pnr}(t(K',P')) &= |K' \setminus \hat{p}_{pnr}(t(K',P'))| \end{aligned}$$

Sei C die Menge der Kandidaten aus Kunden und Produkten, die als nächstes in das Tile eingefügt werden könnten. Die Precision eines Kandidaten $c \in C$ für das Tile t sei definiert als

$$prec(c,t) = \frac{n_c^+(t)}{n_c^+(t) + n_c^-(t)} = \hat{\delta}_c(t)$$

Die Precision nach Einfügen in das Tile beträgt:

$$prec(t+c) = \frac{n^+(t) + n_c^+(t)}{n^+(t) + n_c^+(t) + n^-(t) + n_c^-(t)}$$

Die Precision $prec(t+c)$ wird also maximal, wenn die Precision $prec(c,t)$ des nächsten Kandidaten maximal ist.

In Abbildung 4.2 ist ein Beispiel für die Auswahl des nächsten Kandidaten nach der Heuristik Precision dargestellt.

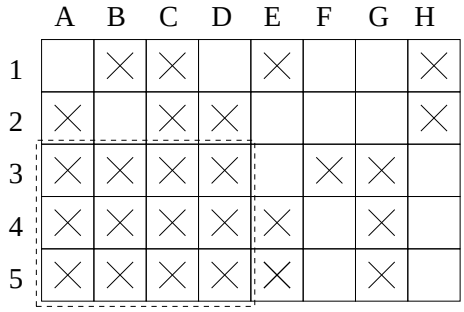


Abbildung 4.2: Precision

prec(1,t) = $\frac{1}{2}$
prec(2,t) = $\frac{3}{4}$
prec(E,t) = $\frac{2}{3}$
prec(F,t) = $\frac{1}{3}$
prec(G,t) = 1
prec(H,t) = 0

⇒ Wähle G zum Einfügen

4.3.3 Weighted Accuracy

Accuracy, wie sie aus dem Regellernen vorgestellt wurde, berechnet die Differenz aus positiven und negativen Beispielen. Dies lässt sich einfach auf das Tiling übertragen, indem positive und negative Kunden-/Produkt-Paare des Tiles t gezählt werden:

$$acc(t) = n^+(t) - n^-(t)$$

Dieses Maß liefert solange positive Werte, wie die Anzahl der positiven Kunden-/Produkt-Paare größer ist als die der negativen. Positive und negative Paare werden gleich stark gewichtet. Fügt man Kunden- und Produktnummern hinzu, solange die Accuracy des Kandidaten größer null ist, so lässt sich eine Dichte des Tiles von 50% oder etwas darüber erwarten. Dies entspricht allerdings nicht unbedingt der vorgegebenen Dichte Δ. Es ist also sinnvoll, schon von Beginn an das gewünschte Verhältnis positiver und negativer Paare einfließen zu lassen. Dazu eignet sich die oben vorgestellte Weighted Relative Accuracy nicht, da sie nur einen Zusammenhang zur Gesamtdichte der Datenbank herstellt; allerdings lässt sich die Idee einer Gewichtung weiterverwenden. Ist beispielsweise eine Dichte Δ von 80% gewünscht, so müssen mindestens 4 von 5 Kunden-/Produkt-Paaren positiv sein. Anders ausgedrückt sind mindestens 4 positive Paare notwendig, um 1 negatives aufzuwiegen. Dies lässt sich durch das Konzept der Weighted Accuracy ausdrücken:

$$wa(t) = n^+(t) \cdot (1-\Delta) - n^-(t) \cdot \Delta$$

Mit diesem Maß erhält man positive Werte, wenn ein negatives Paar durch mindestens $\Delta/(1-\Delta)$ positive Paare ausgeglichen wird. Es handelt sich also um ein typisches Kostenmaß, wie es schon im Abschnitt 4.3.1 vorgestellt wurde.

Wie bei der Precision genügt es auch hier, die Weighted Accuracy eines Kandidaten c mit $wa(c,t) = n_c^+(t) \cdot (1-\Delta) - n_c^-(t) \cdot \Delta$ zu betrachten, da das Maß additiv ist:

$$\begin{aligned} wa(t+c) &= (n^+(t) + n_c^+(t)) \cdot (1-\Delta) - (n^-(t) + n_c^-(t)) \cdot \Delta \\ &= (n^+(t) \cdot (1-\Delta) - n^-(t) \cdot \Delta) + (n_c^+(t) \cdot (1-\Delta) - n_c^-(t) \cdot \Delta) \\ &= wa(t) + wa(c,t) \end{aligned}$$

Wählt man also den Kandidaten c mit maximaler Weighted Accuracy $wa(c,t)$ aus, so ergibt sich automatisch die maximale Weighted Accuracy $wa(t)$ für das Tile. Solange das Maß positive Werte liefert, ist dann auch automatisch gewährleistet, dass das Tile eine Dichte von $\delta \geq \Delta$ besitzt.

Abbildung 4.3 zeigt ein Beispiel für das Maß Weighted Accuracy, wobei die gleiche Datenbank wie in Abbildung 4.2 zugrunde liegt. Beide Heuristiken wählen den gleichen Kandidaten aus. Ein Beispiel für eine unterschiedliche Wahl des Kandidaten ist in Abbildung 4.4 gegeben.

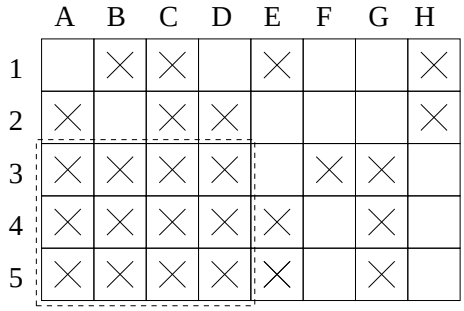


Abbildung 4.3: Weighted Accuracy

$\Delta = 0,75$

wa(1,t)	=	-1
wa(2,t)	=	0
wa(E,t)	=	-0,25
wa(F,t)	=	-1,25
wa(G,t)	=	0,75
wa(H,t)	=	-2,25

$\Rightarrow$ Wähle G zum Einfügen

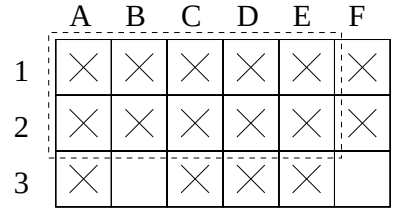


Abbildung 4.4: Unterschied zwischen Precision und Weighted Accuracy bei $\Delta = 0,5$. Precision würde Produktnummer F auswählen, Weighted Accuracy dagegen Kundennummer 3.

4.3.4 Verfeinerungen

Häufig kommt es vor, dass eine Heuristik für verschiedenen Kandidaten den gleichen Wert liefert. Um in diesem Fall eine sinnvolle Auswahl treffen zu können, werden weitere Verfeinerungen der Heuristiken angegeben.

Maximale Fläche

Um den Fall gleicher Heuristikwerte zu illustrieren, werden die in [9] vorgestellten PN-Räume aus dem Regellernen genutzt. In Abbildung 4.5 ist ein solcher PN-Raum dargestellt. Ähnlich wie bei den ROC-Räumen [6], die zur Evaluierung von Klassifizierern genutzt werden, wird hier die Anzahl der von der Regel, bzw. vom Tile, abgedeckten positiven und negativen Beispiele, bzw. Kunden-/Produkt-Paare, gegeneinander aufgetragen. P und N bezeichnen dabei die Gesamtanzahl der positiven und negativen Beispiele/Paare in der Datenbank. Sogenannte Isometriken verbinden alle Punkte, für die eine Heuristik h den gleichen Wert x liefern würde, also $h(a) = h(b)$, $h(c) = h(d)$, aber $h(a) \neq h(c)$.

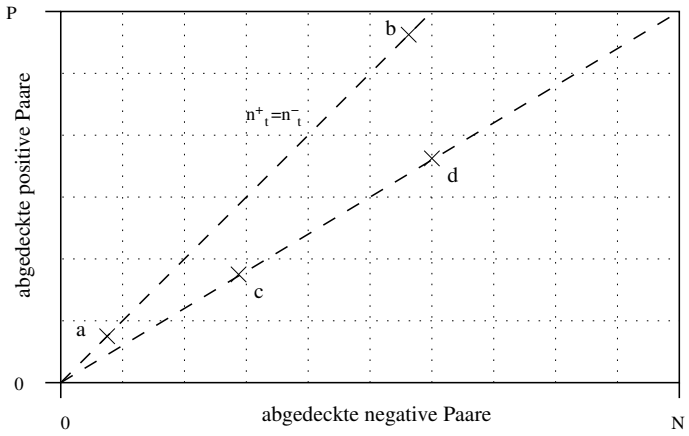


Abbildung 4.5: Isometriken im PN-Raum

In Abbildung 4.6 sind die Isometriken der Heuristik Precision dargestellt. Die hervorgehobene Linie markiert jene Punkte, für die die Anzahl der positiven und negativen Punkte gleich ist, die Precision also 0,5 beträgt. Wenn $\Delta = 0,5$ gelten würde, so wären alle Tiles gültig, deren Precision links/oberhalb dieser Linie liegt.

Die Isometriken der Accuracy für verschiedene Werte von Δ sind in Abbildung 4.7 zu

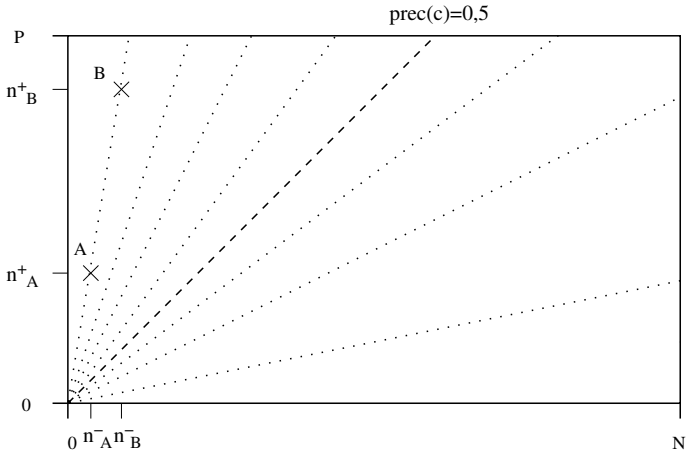
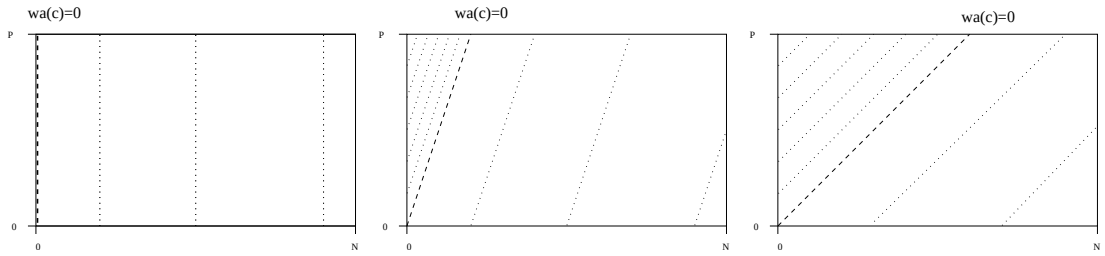


Abbildung 4.6: Precision



(a) $\Delta = 1$ (b) $\Delta = 0,75$ (c) $\Delta = 0,5$

Abbildung 4.7: Weighted Accuracy

sehen. Besonders ist hier die Linie hervorgehoben, die $wa(t) = 0$ markiert. Je kleiner Δ wird, desto größer wird der Raum der zulässigen Tiles.

Liefert eine Heuristik für zwei Tiles den gleichen Wert, so ist zu überlegen, ob sie im Kontext des Tilings wirklich gleich wertvoll sind. Betrachtet man beispielsweise die Tiles A und B in Abbildung 4.6, so haben sie die gleiche Precision bzw. Dichte, unterscheiden sich jedoch in der Fläche ($n_B^+ + n_B^- > n_A^+ + n_A^-$). Da es ein Ziel ist, möglichst große Tiles zu erzeugen, wäre also bei gleicher Dichte Tile B vorzuziehen. Die gleichen Überlegungen gelten für die Heuristik Accuracy. Um möglichst große Tiles zu erzeugen, werden also bei gleichem Heuristikwert jene bevorzugt, die mehr positive Kunden-/Produkt-Paare zum Tile beisteuern. Bei der Auswahl des nächsten Kandidaten wird jedoch nicht die Precision oder Accuracy des gesamten Tiles sondern nur die des Kandidaten berechnet. Dies macht aber für die Überlegungen keinen Unterschied, da der Kandidat mit den meisten positiven Paaren auch den größten Flächenzuwachs für das Tile bedeutet. Diese Heuristik, die die Anzahl der neu zum Tile t hinzukommenden positiven Paare für einen Kandidaten c bewertet, wird mit $maxPos(c,t)$ bezeichnet:

$$maxPos(c,t) = n_c^+(t) = \begin{cases} |\hat{k}_c(t)| & \text{für } c \in K \\ |\hat{p}_c(t)| & \text{für } c \in P \end{cases}$$

Ein Beispiel ist in Abbildung 4.8 dargestellt.

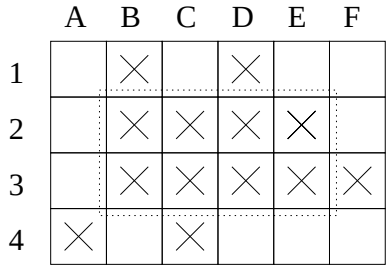


Abbildung 4.8: Für das Tile haben sowohl Kundennummer 1 als auch Produktnummer F eine Precision von 1/2. Kundennummer 1 wird vorgezogen, da sie mehr positive Kunden-/Produkt-Paare beisteuert.

Vorausschauende Auswahl

Liefern die Heuristiken Precision oder Weighted Accuracy gleiche Werte, sollte bei der Auswahl des nächsten einzufügenden Kandidaten versucht werden, jenen Kandidaten aus-

zuwählen, der dem Tile möglichst viel Spielraum für eine weitere Generalisierung lässt. In Abbildung 4.9 ist ein Fall dargestellt, in dem eine solche Entscheidung eine Rolle spielen könnte. Nach Heuristik Precision und Weighted Accuracy haben Kundennummer 1 und 5 den gleichen Wert. Fügt man zunächst jedoch Nummer 5 ein, so ergibt sich anschließend die Möglichkeit die Produktnummern D, E und F einzufügen.

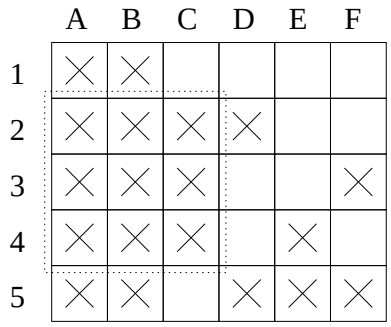


Abbildung 4.9: Auswahl der Kandidaten, die möglichst viele weitere Generalisierungen zulassen.

Um dies zu erreichen werden zwei Heuristiken untersucht:

maxKard wählt den Kandidaten mit größtem Kunden- oder Produktprofil. Es wird der Kandidat c ausgewählt, dessen Kundenprofil die meisten Produktnummern bzw. dessen Produktprofil die meisten Kundennummern enthält. Maximiere

$$maxKard(c,t) = \begin{cases} |k_c| & \text{für } c \in K \\ |p_c| & \text{für } c \in P \end{cases}$$

maxKand wählt den Kandidaten, dessen Kunden- oder Produktprofil die größte Schnittmenge mit anderen Kandidaten bildet. Hier wird nicht betrachtet, wie viele Elemente das zugehörige Kunden-/Produktprofil insgesamt besitzt, sondern es wird geprüft wie viele positive Kunden-/Produkt-Paare das Profil des Kandidaten mit den anderen Kandidaten bildet.
Sei

$$C_K = \{C \cap K\}, \;\; C_P = \{C \cap P\}$$

Wähle c mit maximalem

$$maxKand(c,t) = \begin{cases} |k_c \cap C_P| & \text{für } c \in K \\ |p_c \cap C_K| & \text{für } c \in P \end{cases}$$

In Abbildung 4.10 stehen drei Kandidaten zum Einfügen zur Verfügung, Kundennummer 3 und die Produktnummern C und D. Dabei haben die Kandidaten 3 und D die gleiche Precision von 1. Kundennummer 3 bildet dabei noch ein positives Kunden-/Produkt-Paar mit anderen Kandidaten aus P, nämlich mit Produktnummer C. Produktnummer D dagegen hat keinen Schnittpunkt mit anderen Kandidaten aus K. Es wird also nach Heuristik maxKand Kundennummer 3 gewählt. Heuristik maxKard würde dagegen Produktnummer D vorziehen, da p_D deutlich größer ist.

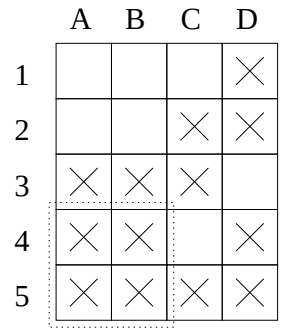


Abbildung 4.10: Heuristik maxKand

4.3.5 Verwendete Kombinationen

Die vorgestellten Heuristiken Precision und Weighted Accuracy werden wie in Abbildung 4.11 mit den Verfeinerungen maxPos, maxKard und maxKand kombiniert, um den nächsten einzufügenden Kandidaten zu finden. Zunächst wird nach Precision oder Weighted Accuracy ausgewählt. Die Kandidaten mit gleichem höchsten Wert werden weiter nach maxPos unterschieden, da ein Maximieren der Fläche bei beispielsweise gleicher Precision im Vordergrund steht. Sollten noch immer mehrere Kandidaten zur Verfügung stehen, werden diese nach den Heuristiken maxKard und maxKand weiter selektiert. Insgesamt ergeben sich also vier unterschiedliche Kombinationen der Heuristiken.

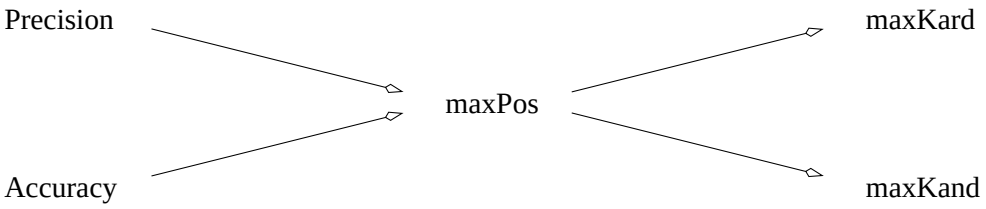


Abbildung 4.11: Die vier untersuchten Kombinationen der Heuristiken

Für die Heuristiken Precision und maxKand läuft die Suche nach dem nächsten Kandidaten wie in Algorithmus 4.2 gezeigt ab. Für die anderen Kombinationen der Heuristiken sieht der Ablauf entsprechend aus.

Algorithmus 4.2 Finde besten Kandidaten mit Precision und maxKand

```
1: t ← Tile
2: K ← Menge der Kandidaten
3: max = dummy // bester Kandidat
4:                // prec(dummy,t) = 0
5:                // maxPos(dummy,t) = 0
6:                // maxKand(dummy,t) = 0
7: for all c ∈ C do
8:    if prec(c,t) < prec(max,t) then
9:       continue
10:   end if
11:   if prec(c,t) > prec(max,t) then
12:      max = c
13:   else
14:      if maxPos(c,t) > maxPos(max,t) then
15:         max = c
16:      else
17:         if maxPos(c,t) = maxPos(max,t) and maxKand(c,t) > maxKand(max,t) then
18:            max = c
19:         end if
20:      end if
21:   end if
22: end for
```

4.4 Kandidaten

In jedem Generalisierungsschritt wird ein neuer Kunde oder ein neues Produkt in das Tile eingefügt. Um zu entscheiden, welcher bzw. welches am geeignetsten ist, werden die Heuristiken aus Abschnitt 4.3 benutzt. Allerdings würde es einen erheblichen Rechenaufwand bedeuten, für jeden in der Datenbank enthaltenen Kunden bzw. für jedes Produkt den entsprechenden Wert der Heuristik zu berechnen. Deshalb werden Listen der Kunden und Produkte geführt, die als *Kandidaten* zum Einfügen in das Tile überhaupt in Frage kommen.

4.4.1 Anforderungen an Kandidaten

Ein Kandidat sollte viele Gemeinsamkeiten mit den bereits im Tile enthaltenen Kunden und Produkten haben. Sie sollten also zum Beispiel keine 'Trittbrettfahrer' für das jeweilige Tile darstellen. Um diese zu vermeiden, wurde bereits in Abschnitt 4.2 die Forderung erhoben, dass die Profile aller in einem Tile enthaltenen Kunden und Produkte eine Dichte $\hat{\delta}(\hat{k}_{knr})$ bzw. $\hat{\delta}(\hat{p}_{pnr})$ von mindestens Δ aufweisen. Da Profile, die nicht diese Dichte aufweisen, auf jeden Fall im Nachbearbeitungsschritt aus dem Tile herausgestrichen werden, macht es Sinn, nur solche Kunden und Produkte als Kandidaten zu führen, die zumindest bezogen auf die Tilegröße im aktuellen Generalisierungsschritt eine ausreichende Dichte aufweisen. In Abschnitt 4.2 wurde bereits auf eine weitere Forderung an die Kandidaten hingewiesen. Im Nachbearbeitungsschritt sollte es möglichst selten vorkommen, dass der Startpunkt, um den ein Tile entwickelt wurde, aufgrund der Dichteanforderung wieder mit einem zu löschenden Profil aus dem Tile entfernt wird. Wird ein Startpunkt auf diese

Weise entfernt, gilt er zwar als *überdeckt*, ist aber vielleicht nicht wirklich in einem Tile des Tilings enthalten. Deshalb sollten nur Kunden oder Produkte eingefügt werden, die die Dichte der Profile des Startpunktes größer als Δ belassen. Insgesamt gibt es also zwei Anforderungen an einen Kandidaten.

Sei Tile $t = (K', P')$ mit Startpunkt (knr, pnr). Dann gilt für einen Kandidaten c:

1. Ausreichende Dichte des Profils des Kandidaten:

$$\Delta \leq \begin{cases} \hat{\delta}(\hat{k}_c(t)) = \dfrac{|k_c \cap P'|}{|P'|} & \text{für } c \in K \\ \\ \hat{\delta}(\hat{p}_c(t)) = \dfrac{|p_c \cap K'|}{|K'|} & \text{für } c \in P \end{cases}$$

2. Weiterhin ausreichende Dichte der Profile des Startpunktes:

$$\Delta \leq \begin{cases} \dfrac{|\hat{p}_{pnr}(t)| + |\{c\} \cap p_{pnr}|}{|K' + 1|} & \text{für } c \in K \\ \\ \dfrac{|\hat{k}_{knr}(t)| + |\{c\} \cap k_{knr}|}{|P' + 1|} & \text{für } c \in P \end{cases}$$

4.4.2 Generierung von Kandidaten

Da es zu aufwändig ist, in jedem Generalisierungsschritt alle Produkte und Kunden der Datenbank zu betrachten, werden Listen der Kunden und Produkte geführt, die den Anforderungen an einen Kandidaten gerecht werden. Diese Listen werden nach einem Generalisierungsschritt aktualisiert. Eventuell müssen zum Beispiel aufgrund der Dichteanforderung einige Kunden und Produkte von der Kandidatenliste gestrichen werden, wogegen wieder neue Kandidaten hinzukommen können.

Zu Beginn der Generalisierungsphase ist nur der Startpunkt im Tile vorhanden. Vom Startpunkt (knr, pnr) ausgehend kommen zunächst nur die Kunden- und Produktnummern aus den Profilen des Startpunktes als Kandidaten in Frage. In einem Generalisierungsschritt wird der beste Kandidat nach einer Heuristik ausgewählt und in das Tile eingefügt. Handelt es sich bei diesem Kandidaten um einen Kunden, so ändert sich zum Beispiel die Dichteanforderung an die Produktprofile und die bisherigen Kandidaten müssen neu überprüft werden. In Algorithmus 4.3 ist dargestellt, wie die Kandidatenlisten bei Einfügen eines Kunden aktualisiert werden. Für Kunden und Produkte werden dabei getrennte Kandidatenlisten C_K und C_P geführt. Nach Einfügen des Kandidaten c in das Tile t wird auf dem Kundenprofil k_c weitergearbeitet. Aus diesem werden zunächst alle Produkte entfernt, die bereits im Tile enthalten sind. Anschließend werden die Produktkandidaten überprüft, da sich die Mindestdichte zum Einfügen eines Produktes geändert hat. Ein vorhandener Produktkandidat muss aber nur dann überprüft werden, wenn er mit dem eingefügten Kunden kein positives Kunden-/Produkt-Paar bildet. Ansonsten bleibt die Dichte des Produktprofils mindestens genauso hoch wie vor dem Einfügen des Kunden. Auch diese Produktkandidaten werden aus dem Profil des eingefügten Kunden gelöscht.

Algorithmus 4.3 Anpassen der Kandidatenlisten bei Einfügen eines Kunden in ein Tile

```
1: INPUT: (knr, pnr)                    // Startpunkt
2:        t = (K', P')                  // Tile
3:        C_K                           // Menge der Kundenkandidaten
4:        C_P                           // Menge der Produktkandidaten
5:        c ∈ C_K                       // Kandidat, der in das Tile eingefügt wird
6:
7: t = (K' ∪ {c}, P')                   // Einfügen in das Tile
8: X = k_c
9: X = X \ P'                           // Entfernen der bereits enthaltenen Produkte
10: for all c_P ∈ C_P do                // Überprüfen der Produktkandidaten
11:    if c_P ∉ X then
12:       if δ̂(p̂_{c_P}(t)) < Δ then
13:          C_P = C_P \ {c_P}
14:       end if
15:    else
16:       X = X \ {c_P}
17:    end if
18: end for
19: for all x ∈ X do                    // Überprüfen neuer Produktkandidaten
20:    if δ̂(p̂_x(t)) ≥ Δ and δ̂(k̂_{knr}(t + x)) < Δ then
21:       C_P = C_P ∪ {x}
22:    end if
23: end for
24: if c ∉ p_{pnr} then                 // Überprüfen der Kundenkandidaten
25:    for all c_K ∈ C_K do
26:       if pnr ∉ c_K then
27:          if δ̂(p̂_{pnr}(t + c_K)) < Δ then
28:             C_K = C_K \ {c_K}
29:          end if
30:       end if
31:    end for
32: end if
```

Die verbleibenden Produkte im Profil des Kunden werden auf ihre Dichte und die resultierende Dichte des Kundenprofils des Startpunktes überprüft und gegebenenfalls in die Liste der Produktkandidaten aufgenommen. Hier können also neue Kandidaten hinzugewonnen werden, die zu Beginn noch keine Verbindung zum Startpunkt hatten. Auf diese Weise werden auch alle entsprechenden Kandidaten gefunden; ein weiteres Durchsuchen der Produkte nach Kandidaten ist nicht notwendig. Wenn der eingefügte Kunde nicht im Produktprofil des Startpunktes enthalten ist, sinkt die Dichte des Produktprofiles des Startpunktes und die vorhandenen Kundenkandidaten müssen daraufhin überprüft werden, ob sie diese Dichte nicht zu gering werden lassen. Wenn der eingefügte Kunde im Produktprofil des Startpunktes enthalten ist, sind die bereits gefundenen Kundenkandidaten nicht betroffen, da die Dichte des Kundenprofiles des Startpunktes steigt. Es kann dann allerdings vorkommen, dass Kunden wieder in die Kandidatenliste aufgenommen werden könnten, die ausgeschlossen wurden, da sie die Dichte dieses Profil zu sehr erniedrigt haben. Um diese Kandidaten wieder zu finden, müsste allerdings die Menge der gesamten Kunden durchsucht werden. Dies bedeutet einen erheblichen Rechenaufwand, weshalb das Verlorengehen der wenigen Kandidaten aufgrund dieses Effektes in Kauf genommen wird.

Sie werden außerdem wieder in Betracht gezogen, wenn sie im Produktprofil eines neu eingefügten Produktes enthalten sind. Das Anpassen der Kandidatenlisten bei Einfügen eines Produktes in das Tile erfolgt genau entsprechend.

4.5 Evaluation der Heuristiken

Die letztendlich angewendeten Heuristiken wurden durch Aneinanderreihen verschiedener Teilheuristiken gebildet. Zur Evaluation bietet es sich an, besonders Precision und Weighted Accuracy sowie die Heuristiken maxKard und maxKand miteinander zu vergleichen.

4.5.1 Vergleich von Precision und Weighted Accuracy

Durch die Forderung, dass die Dichte $\hat{\delta}$ jedes in einem Tile enthaltenen Profils größer als Δ sein muss, wurde bereits sicher gestellt, dass die negativen Kunden-/Produkt-Paare gleichmäßig über Kunden und Produkte verteilt sind. Deshalb ist es Ziel der Heuristiken, möglichst große Tiles zu generieren. Ein Hauptkriterium für den Vergleich der Heuristiken ist daher die durchschnittliche Fläche eines Tiles. In Tabelle 4.2 sind die durchschnittlichen Flächen der Tiles für die Heuristiken Precision und Weighted Accuracy mit anschließenden Heuristiken maxPos und maxKard für unterschiedliche Dichten aufgetragen.

	DB1		DB2		DB3		DB4		DB5	
Δ	prec	wa	prec	wa	prec	wa	prec	wa	prec	wa
1,0	19	19	19	19	19	19	19	19	20	20
0,75	27	28	26	26	28	28	28	28	28	28
0,5	87	90	84	89	83	85	90	92	83	86
	DB6		DB7		DB8		DB9		DB10	
Δ	prec	wa	prec	wa	prec	wa	prec	wa	prec	wa
1,0	19	19	19	19	19	19	19	19	19	19
0,75	28	29	27	28	27	27	26	26	29	29
0,5	88	90	84	86	79	82	80	82	88	90

Tabelle 4.2: Vergleich von Precision und Weighted Accuracy - Durchschnittliche Fläche φ eines Tiles

Es fällt sofort auf, dass die Heuristiken mit Precision und Accuracy für hohe Werte von Δ meistens gleiche Durchschnittsflächen der Tiles liefern. Bei genauem Betrachten einzelner Tiles und deren Entstehen kann man sehen, dass beide Heuristiken tatsächlich fast immer zu den gleichen Generalisierungsschritten führen. Dies liegt daran, dass Precision und Accuracy die Kandidaten zwar leicht unterschiedlich bewerten, durch die nachfolgende Heuristik maxPos jedoch immer der gleiche Kandidat ausgewählt wird. Ein typisches Beispiel hierfür ist, dass Weighted Accuracy für zwei Kandidaten den gleichen Wert liefert, also beispielsweise -1 für einen Kandidaten mit zwei positiven und zwei negativen Kunden-/Produkt-Paaren und -1 für einen Kandidaten mit fünf positiven und drei negativen Paaren. Precision entscheidet sich hier eindeutig für den zweiten Kandidaten. Durch die nachfolgende Heuristik maxPos wird jedoch auch bei der Heuristik mit Weighted Accuracy der zweite Kandidat gewählt. Unterschiede zwischen den Heuristiken mit Precision

	DB1		DB2		DB3		DB4		DB5	
	prec	wa	prec	wa	prec	wa	prec	wa	prec	wa
φ	87	90	84	89	83	85	90	92	83	86
δ	0,78	0,76	0,78	0,77	0,79	0.78	0.78	0,78	0,79	0,78
	DB6		DB7		DB8		DB9		DB10	
	prec	wa	prec	wa	prec	wa	prec	wa	prec	wa
φ	88	90	84	86	79	82	80	82	88	90
δ	0,78	0,78	0,80	0,79	0,80	0,80	0,80	0,79	0,79	0,79

Tabelle 4.3: Vergleich von Precision und Weighted Accuracy - Durchschnittliche Dichte δ eines Tiles für $\Delta = 0,5$

und Weighted Accuracy treten oft erst bei kleineren Werten von Δ auf, in Tabelle 4.2 also ab $\Delta = 0,5$. Während mit Precision zunächst immer so weit wie möglich vollständige Tiles erzeugt werden, bevorzugt Weighted Accuracy Kandidaten mit größerer Fläche, die aber einige wenige negative Kunden-/Produkt-Paare enthalten. Ein Beispiel hierfür wurde bereits in Abbildung 4.4 gezeigt. Dieser Trend der Weighted Accuracy zu größeren Flächen und geringeren Dichten zeigt sich in Tabelle 4.2 ab $\Delta = 0,5$. In Tabelle 4.3 sind dazu noch einmal die durchschnittlichen Dichten der Tiles für $\Delta = 0,5$ aufgetragen. Während die Fläche bei der Heuristik mit Weighted Accuracy im Durchschnitt etwas größer ist, fällt die Dichte leicht ab. Insgesamt ist die durchschnittliche Dichte der Tiles aufgrund der Dichteanforderung wesentlich größer als Δ.

Da die Heuristik mit Weighted Accuracy zumindest für kleine Δ größere Tiles erzeugt, ist diese also einer Heuristik mit Precision vorzuziehen.

4.5.2 Vergleich der Heuristiken maxKard und maxKand

Zunächst sollen wieder die durchschnittlichen Flächen der Tiles für die Heuristiken gegenübergestellt werden. Als Basis dient hier die Heuristik Weighted Accuracy. In Tabelle 4.4 sind die durchschnittlichen Flächen für unterschiedliche Werte von Δ dargestellt.

Der Tabelle ist zu entnehmen, dass maxKand für große Δ wesentlich größere Tiles erzeugt. Hier wird deutlich, dass es nicht genügt, pauschal große Profile zu bevorzugen, sondern dass immer die weiteren Entwicklungsmöglichkeiten des Tiles betrachtete werden müssen. Mit kleinerem Δ nimmt dieser Effekt ab und maxKard erzeugt genauso große Tiles wie maxKand. Da für $\Delta = 0,5$ maxKard sogar noch größere Tiles erzeugt, wurde diese Entwicklung für $\Delta = 0,3$ überprüft. Allerdings lässt sich hier kein Vorteil von maxKard gegenüber maxKand erkennen, sodass maxKand eindeutig als bessere Heuristik anzusehen ist. Die durchschnittliche Dichte der Tiles ist bei beiden Heuristiken gleich. Diese Beobachtungen bestätigen sich auch für eine Kombination von Precision mit maxKard und maxKand, die hier nicht mehr gezeigt wurde.

Im Folgenden wird also immer als Heuristik die Kombination von Weighted Accuracy und maxKand benutzt!

Zusätzlich soll noch einmal betrachtet werden, wie viele der Tiles, die künstlich erzeugt wurden, durch den Algorithmus wiedergefunden wurden. Der Prozentsatz der Tiles, die

	DB1		DB2		DB3		DB4		DB5	
Δ	max-Kard	max-Kand	max-Kard	max-Kand	max-Kard	max-Kand	max-Kard	max-Kand	max-Kard	max-Kand
1,0	19	39	19	38	19	39	19	39	20	38
0,75	28	49	26	47	28	48	28	51	28	51
0,5	90	89	89	82	85	83	92	84	86	86
0,3	422	427	396	388	382	383	437	471	389	385

	DB6		DB7		DB8		DB9		DB10	
Δ	max-Kard	max-Kand	max-Kard	max-Kand	max-Kard	max-Kand	max-Kard	max-Kand	max-Kard	max-Kand
1,0	19	37	19	38	19	39	19	38	19	42
0,75	29	49	28	50	27	49	26	47	29	50
0,5	90	86	86	81	82	82	82	81	90	86
0,3	423	413	398	397	336	338	365	373	433	434

Tabelle 4.4: Vergleich von maxKard und maxKand - Durchschnittliche Fläche φ eines Tiles

Δ	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8	DB9	DB10
1,0	98,00	96,00	98,00	97,33	95,33	93,33	97,33	96,00	98,00	97,33
0,75	100,00	100,00	99,33	100,00	100,00	99,33	99,33	100,00	98,67	100,00
0,5	98,67	99,33	100,00	100,00	100,00	99,33	100,00	100,00	99,33	99,33

Tabelle 4.5: Prozentsatz der abgedeckten Tiles aus dem original enthaltenen Tiling (Weighted Accuracy, maxKand)

komplett durch ein Tile des gefundenen Tilings abgedeckt wurden, ist in Tabelle 4.5 gezeigt. Fast alle Tiles werden bereits bei $\Delta = 1,0$ gefunden. Es kann also davon ausgegangen werden, dass Strukturen, die sich im Datensatz befinden, auch vom Algorithmus gefunden werden. Im Kapitel 5 soll dies nochmal unter Einfügen von Rauschen getestet werden.

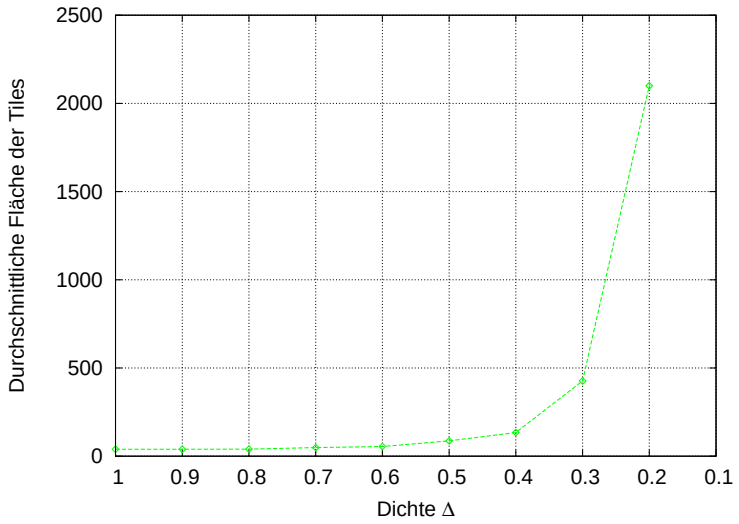


Abbildung 4.12: Flächenzuwachs mit fallendem Δ

Interessant ist auch der Flächenzuwachs der Tiles mit fallendem Δ. Die in die Datenbanken eingefügten Tiles hatten eine durchschnittliche Größe von $\varphi = 60$. In Abbildung 4.12 ist die durchschnittliche Fläche der Tiles am Beispiel der Datenbank 1 abhängig von Δ aufgetragen.

Es ist verständlich, dass die Fläche mit Δ nahe 1,0 nur wenig steigt, da die Tiles bereits sehr viele Kunden- oder Produktprofile enthalten müssen, um ein negatives Kunden-/Produkt-Paar enthalten zu können. Ab $\Delta = 0,6$ steigt die Fläche dann aber rapide an.

4.6 Relaxierung der Dichteanforderung

Das Tiling der Datenbank sollte bei einem vorgegebenen Schwellenwert von Δ möglichst große und dichte Tiles enthalten, deren Dichte in jedem Fall mindestens Δ beträgt. Um ein Vergrößern der Tiles stark zu Lasten der Dichte vermeiden zu können, wurde die Dichteanforderung für Profile aufgestellt, die besagt, dass die Dichte jedes Profils innerhalb eines Tiles ebenfalls mindestens Δ betragen muss. Dadurch kann es jedoch zu Problemen beim inkrementellen Aufbau des Tiles kommen. In Abbildung 4.13 ist ein Datenbankausschnitt mit den Kunden 1-4 und Produkten A-D zu sehen, der bei $\Delta = 75\%$ ein gültiges Tile darstellt, in dem die maximale Anzahl an negativen Kunden-/Produkt-Paaren optimal verteilt ist, sodass auch die Dichteanforderung für alle enthaltenen Profile erfüllt ist. Allerdings ist es mit dem bisher gewählten Ansatz nicht möglich, dieses Tile zu finden, da innerhalb der Generalisierungsphase keine gültigen Kandidaten nach den Anforderungen in Abschnitt 4.4.1 zur Verfügung stehen. Ausgehend vom Startpunkt (1,D) wird bei jeder Heuristik zunächst das Tile ({1,2},{C,D}) gebildet. Jetzt sind sowohl unter den Kunden als auch unter den Produkten keine Kandidaten mehr vorhanden. Beispielhaft wurden als Kandidaten Kunde 3 und Produkt B angedeutet. Beide hätten im Tile nur eine Dichte von 50%. Sie könnten also nicht in das Tile eingefügt werden.

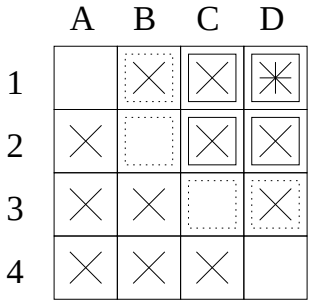


Abbildung 4.13: Probleme beim Aufbau mit bisheriger Dichteanforderung

Um dieses Problem zu umgehen, könnte die Anforderung an die Kandidaten gelockert werden. Es könnten also auch Profile in das Tile eingefügt werden, die nicht eine Mindestdichte von Δ innerhalb des Tiles haben. Ist es jedoch nicht möglich, das Tile so zu erweitern, dass diese Profile am Ende der Generalisierungsphase die Dichteanforderung erfüllen, so müssen sie wieder aus dem Tile gelöscht werden. Oft ist es dann der Fall, dass viele Einfügeoperationen sinnlos durchgeführt werden und nur unnötigen Rechenaufwand bedeuten. Wird allerdings auch die Dichteanforderung ein wenig relaxiert, so ist ein Auf-

finden von Tiles wie in Abbildung 4.13 leichter möglich. Dabei muss die Gesamtdichte des Tiles aber weiterhin mindestens Δ betragen.

4.6.1 Kandidaten

Bei vorgegebenem Schwellenwert Δ für die Dichte eines Tiles $t = (K', P')$ muss die Anzahl der positiven Kunden-/Produkt-Paare mindestens

$$n_t^+ = \Delta \cdot |K'| \cdot |P'|$$

betragen. Die maximale Anzahl der negativen Paare beträgt dann:

$$n_t^- = (1 - \Delta) \cdot |K'| \cdot |P'|$$

Nimmt man an, dass die positiven Paare gleichmäßig auf alle Kundenprofile verteilt sind, so entspricht dies in etwa einer Binomialverteilung mit der Wahrscheinlichkeit für die Auswahl eines Kunden von $p = 1/|K'|$ und N_t^+ Bernoulli-Experimenten. Es handelt sich jedoch nicht wirklich um eine Binomialverteilung, da einem Kundenprofil maximal $|P'|$ positive Paare zugeordnet werden können, aber für die nachfolgenden Betrachtungen ist der Vergleich ausreichend. Der Erwartungswert für die Anzahl der positiven Paare in einem Kundenprofil beträgt dann

$$E_{k,t}^+ = \frac{1}{|K'|} \cdot n_t^+ = \Delta \cdot |P'|$$

Entsprechend ergibt sich ein Erwartungswert für die Anzahl der negativen Paare zu

$$E_{k,t}^- = \frac{1}{|K'|} \cdot n_t^- = (1 - \Delta) \cdot |P'|$$

Die Varianz und Standardabweichung der Anzahl der negativen Paare bezüglich eines Kunden beträgt dann

$$s^{2-}_{k,t} = n_t^- \cdot \frac{1}{|K'|} \cdot (1 - \frac{1}{|K'|}), \quad s_{k,t}^- = \frac{1}{|K'|} \cdot \sqrt{n_t^- \cdot (|K'| - 1)}$$

Diese Überlegungen lassen sich in gleicher Weise auf die Verteilung der positiven und negativen Paare der Produkte übertragen. Erwartungswert, Varianz und Standardabweichung betragen dann:

$$E_{p,t}^+ = \frac{1}{|P'|} \cdot n_t^+ = \Delta \cdot |K'|, \quad E_{p,t}^- = \frac{1}{|P'|} \cdot n_t^- = (1 - \Delta) \cdot |K'|$$

$$s^{2-}_{p,t} = n_t^- \cdot \frac{1}{|P'|} \cdot (1 - \frac{1}{|P'|})$$

$$s_{p,t}^- = \frac{1}{|P'|} \cdot \sqrt{n_t^- \cdot (|P'| - 1)}$$

Grundgedanke der Relaxierung ist es, dass die Anzahl der negativen Paare nicht mehr unbedingt unter dem Erwartungswert liegen muss (wie es bisher der Fall war), sondern eine

Abweichung im Rahmen der Standardabweichung zulässig ist. Die Anzahl der negativen Paare bezüglich eines Kunden- bzw. Produktprofils darf dann

$$E^-_{k,t} + s^-_{k,t}$$

bzw.

$$E^-_{p,t} + s^-_{p,t}$$

betragen.

Für einen Kundenkandidaten c muss also bezogen auf die positiven Kunden-/Produkt-Paare folgendes gelten:

$$\begin{aligned} |\hat{k}_c(t)| = |k_c \cap P'| &\geq E^+_{k,t+c} - s^-_{k,t+c} \\ &\geq \Delta \cdot |P'| - \frac{1}{|K'|+1} \cdot \sqrt{(1-\Delta) \cdot (|K'|+1) \cdot |P'| \cdot |K'|} \end{aligned}$$

Für einen Kandidaten c aus der Menge der Produkte entsprechend:

$$\begin{aligned} |\hat{p}_c(t)| = |p_c \cap K'| &\geq E^+_{p,t+c} - s^-_{p,t+c} \\ &\geq \Delta \cdot |K'| - \frac{1}{|P'|+1} \cdot \sqrt{(1-\Delta) \cdot (|P'|+1) \cdot |K'| \cdot |P'|} \end{aligned}$$

Die jetzigen Anforderungen passen sich also auch der Größe des aktuellen Tiles an und stellen unterschiedliche Anforderungen an die Dichte von Kunden- und Produktprofilen.

Wird ein Kundenkandidat in das Tile eingefügt, so müssen anschliessend nur die Produktkandidaten auf ihre Zulässigkeit überprüft werden. Durch die Änderung der Fläche des Tiles ändert sich zwar auch die Anforderung an die Kundenkandidaten, aber die Standardabweichung steigt mit größer werdender Fläche, sodass zuvor gültige Kandidaten auch nach dem Einfügen eines Kunden weiterhin gültig sind.

4.6.2 Varianz als Heuristik

Bei den bisherigen Heuristiken wurde immer nur die Anzahl der negativen Kunden-/Produkt-Paare zu einem Profil und einem Tile bewertet. Dabei wurde nicht betrachtet, wie die negativen Paare innerhalb eines Tiles verteilt waren. In Abbildung 4.14 sind ein Tile und zwei mögliche Kandidaten bei $\Delta = \frac{2}{3}$ dargestellt. Nach den bisherigen Heuristiken Precision und Weighted Accuracy haben beide Kandidaten den gleichen Wert. Betrachtet man jedoch die Verteilung der negativen Kunden-/Produkt-Paare, so wäre Produkt G dem Produkt F vorzuziehen, da hier die negativen Paare gleichmäßig über Kunden und Produkte verteilt werden.

Ein Maß, das diese Verteilung der positiven und negativen Paare bewertet, ist die Varianz. Mit der aktuellen Dichte $\delta(t)$ beträgt die Varianz bezüglich der Kundenprofile des Tiles t

$$var_{K',t} = \frac{1}{|K'|} \cdot \sum_{knr \in K'} \left(\hat{\delta}(\hat{k}_{knr}(t)) - \delta(t) \right)^2$$

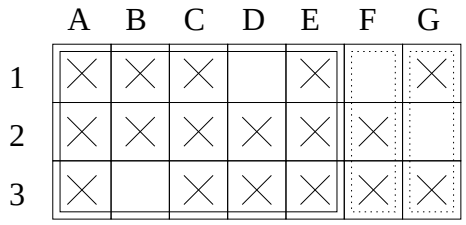


Abbildung 4.14: Varianz als Gütemaß

Für die Produktprofile beträgt sie analog

$$var_{P',t} = \frac{1}{|P'|} \cdot \sum_{pnr \in P'} \Big(\hat{\delta}(\hat{p}_{pnr}(t)) - \delta(t)\Big)^2$$

Ein Kandidat c lässt sich also anhand der Varianz des Tiles bezüglich Kunden- und Produktprofilen bewerten, das nach Einfügen des Kandidaten entstehen würde:

Bewertung eines Kundenkandidaten $c \in K$:

$$var(c, t + c) =$$

$$\frac{1}{|K'| + |P'| + 1} \cdot \Bigg(\sum_{knr \in K' \cup \{c\}} \Big(\hat{\delta}(\hat{k}_{knr}(t)) - \delta(t)\Big)^2 + \sum_{pnr \in P'} \Big(\frac{|p_{pnr} \cap (K' \cup \{c\})|}{|K'| + 1} - \delta(t)\Big)^2 \Bigg)$$

Bewertung eines Produktkandidaten $c \in P$:

$$var(c, t + c) =$$

$$\frac{1}{|K'| + |P'| + 1} \cdot \Bigg(\sum_{knr \in K'} \Big(\frac{|k_{knr} \cap (P' \cup \{c\})|}{|P'| + 1} - \delta(t)\Big)^2 + \sum_{pnr \in P' \cup \{c\}} \Big(\hat{\delta}(\hat{p}_{pnr}(t)) - \delta(t)\Big)^2 \Bigg)$$

Da durch die Relaxierung der Dichteanforderung die Dichte der Profile kleiner als Δ werden kann, muss außerdem sichergestellt werden, dass die Gesamtdichte $\delta(t)$ hoch genug bleibt. Deshalb muss zusätzlich für einen Kandidaten c gelten:

$$\delta(t + c) \geq \Delta$$

Ausgewählt wird der Kandidat mit der geringsten Varianz. Allerdings ist auch hier die Varianz alleine als Heuristik nicht ausreichend, und die weiteren Verfeinerungen *maxPos* und *maxKand* werden wie bei Precision und Weighted Accuracy zusätzlich angewendet. Bei alleiniger Benutzung der Varianz erfolgt ansonsten wahrscheinlich eine Erweiterung in ausschließlich einer Dimension, also nur um Kunden oder nur um Produkte. Da maxKand sich als bessere Heuristik als maxKard erwiesen hat, wird auf eine Kombination der Varianz mit maxKard verzichtet. Der Pseudocode zur Auswahl des besten Kandidaten sieht also sehr ähnlich aus wie der in Algorithmus 4.2.

Als zusätzliche Änderung zum bisherigen Vorgehen wird die Überprüfung der Gültigkeit der im Tile enthaltenen Kunden und Produkte nach jeder Einfügeoperation durchgeführt,

da die Zulässigkeit der Kandidaten und vor allem die Varianz als Heuristik stark vom aktuellen Aufbau des Tiles abhängen. Da jedoch die Varianz für eine ausgeglichene Verteilung der negativen Kunden-/Produkt-Paare sorgt, ist ein Entfernen von Kunden oder Produkten aus einem Tile sehr selten. Haben mehrere Profile zu geringe Dichte, so wird jenes Profil zuerst entfernt, nach dessen Entfernung das Tile die geringste Varianz aufweist.

4.6.3 Evaluation

Auch hier wird die durchschnittliche Fläche der entstehenden Tiles betrachtet. In Tabelle 4.6 ist diese für die relaxierte und die strikte Dichteanforderung mit der gewählten Heuristik Weighted Accuracy gegenübergestellt.

	DB1		DB2		DB3		DB4		DB5	
Δ	var	wa	var	wa	var	wa	var	wa	var	wa
1,0	39	39	38	38	39	39	39	39	38	38
0,75	59	49	57	47	59	48	60	51	58	51
0,5	209	87	227	82	214	83	250	84	219	86
	DB6		DB7		DB8		DB9		DB10	
Δ	var	wa	var	wa	var	wa	var	wa	var	wa
1,0	37	37	38	38	39	39	38	38	42	42
0,75	58	40	58	50	57	49	57	47	60	50
0,5	240	86	225	81	208	82	207	81	230	86

Tabelle 4.6: Durchschnittliche Fläche φ der Tiles für die relaxierte Dichteanforderung mit Heuristik Varianz (var) und die strikte Dichteanforderung mit Heuristik Weighted Accuracy (wa)

Für eine geforderte Dichte von $\Delta = 100\%$ ergeben beide Heuristiken gleich große Tiles. Mit fallendem Δ sind die Tiles, die der strikten Dichteanforderung genügen müssen, jedoch wesentlich kleiner als die, die aus der relaxierten Variante entstehen. Der Grund hierfür wird deutlich, wenn man die durchschnittliche Dichte der Tiles abhängig von Δ betrachtet.

	DB1		DB2		DB3		DB4		DB5	
Δ	var	wa	var	wa	var	wa	var	wa	var	wa
0,75	0,94	0,99	0,94	0,98	0,94	0,99	0,94	0,99	0,94	0,99
0,5	0,53	0,84	0,53	0,86	0,53	0,85	0,53	0,86	0,53	0,84
	DB6		DB7		DB8		DB9		DB10	
Δ	var	wa	var	wa	var	wa	var	wa	var	wa
0,75	0,94	0,99	0,95	0,98	0,95	0,99	0,95	0,99	0,94	0,99
0,5	0,53	0,84	0,53	0,87	0,53	0,86	0,53	0,85	0,53	0,86

Tabelle 4.7: Durchschnittliche Dichte δ der Tiles für die relaxierte und strikte Dichteanforderung

Mit der relaxierten Dichteanforderung wird die Dichte der Tiles bei kleinerem Schwellenwert fast bis zu Δ gesenkt. Es wird also fast die maximal zulässige Anzahl an negativen

Kunden-/Produkt-Paaren ausgeschöpft, wodurch wesentlich größere Tiles entstehen. Da die Dichte der enthaltenen Profile laufend kontrolliert wird, ist eine optimale Verteilung der negativen Kunden-/Produkt-Paare gewährleistet. Der Prozentsatz der Tiles in der Datenbank, die vom Algorithmus wiedergefunden werden, ist bei der relaxierten Dichteanforderung vergleichbar zur strikten Dichteanforderung.

Eine Relaxierung der Dichteanforderung kombiniert mit einem Varianzmaß ist also durchaus sinnvoll, da sehr große Tiles erzeugt werden, in denen negative Paare sehr gleichmäßig über das Tile verteilt sind.

4.7 Starttiles

Nach Festlegen des Startpunktes werden zunächst alle Kandidaten ermittelt, die in das Tile eingefügt werden könnten. Das sind bei Startpunkt (knr,pnr) zunächst alle Elemente aus $k_{knr} \setminus \{pnr\}$ und $p_{pnr} \setminus \{knr\}$. Mit jedem Kunden-/Produktprofil, das in das Tile eingefügt wird, kommen dann weitere Kandidaten hinzu. Dagegen werden andere Kunden- und Produktnummern aus der Kandidatenliste gestrichen, da ihre Dichte mit neu eingefügten Produkt bzw. Kundenprofilen nicht mehr ausreichend ist. Tendenziell ist die Anzahl der Kandidaten zu Beginn am größten und fällt danach mit jedem neu eingefügten Element ab.

Um diese große Anzahl zu untersuchender Kandidaten zu Beginn zu vermeiden, wird versucht, bereits in einer mittleren Phase des Generalisierungsprozesses einzusetzen, d.h. nicht mit einem einzelnen Punkt zu beginnen, sondern als Ausgangsbasis bereits ein größeres Tile zu wählen. Dann ist zu erwarten, dass die Anzahl der zur Verfügung stehenden Kandidaten weitaus geringer ist. Allein um das größte Starttile aus den Elementen der Profile des Startpunktes zu finden, müsste bereits eine Suche gestartet werden. Stattdessen werden auch hier zwei heuristische Strategien betrachtet. Beide gehen wie zuvor von einem Startpunkt aus, um den ein Starttile gebildet wird, allerdings nicht schrittweise wie in Abschnitt 4.1 und ohne Beachtung der in Abschnitt 4.4 oder 4.6.2 vorgestellten Heuristiken. Ausgehend von diesem Starttile erfolgt dann eine normale Generalisierungsphase, in der die Kandidaten nacheinander in das Tile eingefügt werden.

4.7.1 Eindimensionale Suche

In dieser Variante wird das Tile ausgehend vom Startpunkt zunächst in eine Dimension, also nur um Kundenprofile oder nur um Produktprofile erweitert. Anschließend werden alle Produkt- bzw. Kundenprofile hinzugefügt, die nach der Dichteanforderung aus 4.4 zulässig sind. Kandidaten, die nach der relaxierten Dichteanforderung zulässig sind, können auch in der anschließenden Generalisierungsphase noch eingefügt werden. Um das größte Starttile zu finden, werden beide möglichen Dimensionen untersucht, da sich zum Beispiel bei stark unterschiedlicher Kunden- und Produktanzahl sehr verschiedene Tiles ergeben können. Das Prinzip der Bildung des Starttiles wird anhand der Kunden vorgestellt. Werden zuerst Produkte in das Tile eingefügt, so geschieht dies auf analoge Weise.

Zum Einfügen in das Tile werden alle Kunden betrachtet, die in p_{pnr} enthalten sind. Alle Kundennummern in das Tile einzufügen ist nicht sinnvoll, da dann wahrscheinlich nur noch

sehr wenige, vielleicht auch keine Produkte in das Tile eingefügt werden können. Deshalb werden Kunden nach dem Kriterium eingefügt, wie viele gemeinsame Produktnummern ihr Profil mit dem Kundenprofil der Kundennummer im Startpunkt hat:

$$|k_x \cap k_{knr}| \qquad (x \in p_{pnr})$$

Wenn die Überschneidung möglichst groß ist, wird gewährleistet, dass möglichst viele Produktnummern in das Tile eingefügt werden können. Die Anzahl der Kunden, die zu Beginn in das Tile eingefügt werden, sollte wie bereits beschrieben nicht allzu hoch sein, um für die spätere Generalisierungsphase genügend Spielraum zu lassen. Werden andererseits zu wenig Kunden ausgewählt, werden eventuell zu viele Produkte eingefügt, sodass in der späteren Generalisierungsphase keine weiteren Kunden eingefügt werden können. Aus Tests ergibt sich 1/3 der Kundennummern aus p_{pnr} als sinnvolle Grenze. Nach dem Einfügen der Kundenprofile werden alle Produktnummern aus k_{knr} eingefügt, deren Profile genügend hohe Dichte aufweisen. In Algorithmus 4.4 ist die Vorgehensweise noch einmal als Pseudocode zusammengefasst.

Algorithmus 4.4 Eindimensionale Suche zum Finden eines Starttiles

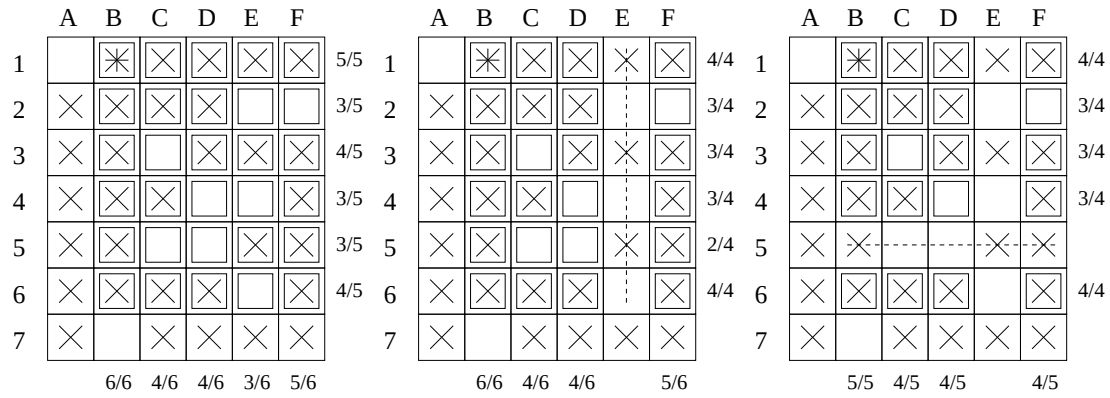
```
 1: s := (knr,pnr)                    // Startpunkt
 2: t := ({knr},{pnr})                // Tile
 3: A := p_pnr \ {knr}
 4: Sortiere A absteigend nach |k_x ∩ k_knr|
 5: n := ⌊|p_pnr|/3⌋
 6: for all i in 0 ≤ i < n do
 7:    t = t + A[i]
 8: end for
 9: for all x in k_knr do
10:    if δ̂(p̂_x(t)) ≥ Δ then
11:       t = t + x
12:    end if
13: end for
```

Aus den beiden Tiles, die durch Einfügen von Kunden oder Produkten entstanden sind, wird das größere ausgewählt und für dieses die weiteren Kandidaten berechnet.

4.7.2 Zweidimensionale Tilebildung

Nun werden Kundenprofile und Produktprofile gleichzeitig zum Entwickeln eines Starttiles herangezogen. Dadurch sollen weder Kunden- noch Produktprofile bevorzugt werden. Im Gegensatz zu den bisherigen Verfahren wird hier ein Top-Down-Ansatz gewählt. Es wird also keine Generalisierung eines Tiles vorgenommen, sondern stattdessen von einem großen – wahrscheinlich nicht zulässigen – Tile ausgegangen und dieses auf ein zulässiges Tile reduziert. Dieses große Tile um den Startpunkt (knr, pnr) erhält man, indem alle Produktnummern aus k_{knr} und alle Kundennummern aus p_{pnr} in das Tile eingefügt werden. Anschließend müssen die enthaltenen Kunden- und Produktprofile wieder auf ihre Dichte geprüft werden. Das Herausstreichen der Profile mit ungenügender Dichte erfolgt auf die gleiche Art und Weise wie in der üblichen Nachbearbeitungsphase nach Fertigstellen des Tiles (siehe Abschnitt 4.2) bzw. bei relaxierter Dichteanforderung nach dem in Abschnitt 4.6.2 beschriebenen Vorgehen unter Berücksichtigung der Varianz.

In Abbildung 4.15 sind die einzelnen Schritte aufgeführt, die zu einem Starttile führen. Die gewünschte minimale Dichte Δ beträgt 75% (ohne Relaxierung). Neben den Kunden- und Produktprofilen der Matrix sind deren Dichte $\hat{\delta}$ bezüglich des Tiles angegeben.



(a) Starttile aus $\hat{k}_1$ und $\hat{p}_B$

(b) Produktprofil $\hat{p}_E$ mit kleinster Dichte $\hat{\delta} < \Delta$ wird gelöscht

(c) Produktprofil $\hat{k}_5$ mit kleinster Dichte $\hat{\delta} < \Delta$ wird gelöscht

Abbildung 4.15: Bilden eines Starttiles durch zweidimensionale Tilebildung für $\Delta = 0.75$

4.7.3 Evaluation

Starttiles wurden eingeführt, um die Anzahl der Kandidaten, die in jedem Generalisierungsschritt betrachtet werden, zu reduzieren. Deshalb wird zunächst verglichen, wie viele Kandidaten im Durchschnitt maximal betrachtet werden, d.h. bei jedem Tile wird die maximale Anzahl der Kandidaten, die jemals im Generalisierungsprozess auftritt, ermittelt und für diese der Durchschnitt gebildet. In Tabelle 4.8 ist diese maximale Anzahl für die normale Dichteanforderung aufgelistet. Auf die Auflistung der Werte für die Datenbanken 7-10 wurde verzichtet. Auf diesen Datenbanken haben sich aber keine Abweichungen zu den Werten auf den aufgeführten Datenbanken ergeben. In der ersten Spalte ist die maximale Anzahl der Kandidaten ohne Verwendung eines Starttiles angegeben. Die jeweils zweite und dritte Spalte enthält die Anzahl der Kandidaten bei ein- bzw. zweidimensionaler Starttilebildung.

Die maximale Anzahl der Kandidaten geht bei Verwendung von Starttiles deutlich zurück. Besonders bei der Bildung von Starttiles aus Kunden- und Produktprofilen gleichzeitig werden sehr wenige Kandidaten betrachtet. Bei der 1-dimensionalen Starttilebildung ist die Anzahl der Kandidaten nicht immer niedriger. Bei kleinem Δ wird die Anzahl der Kandidaten sogar größer. Dies liegt daran, dass die Anzahl der Kunden- oder Produktprofile, die zu Beginn in das Tile eingefügt werden, schlecht gewählt ist. Es entstehen Tiles, die wenige Kunden aber sehr viele Produkte enthalten, oder andersherum. Deshalb gibt es viele Kandidaten, deren Profile ebenfalls zum Beispiel 50% der wenigen enthaltenen Kunden oder Produkte umfassen. Insgesamt ergeben sich so sehr unausgewogene Tiles, die beispielsweise für die obigen Datenbanken circa drei Produkte, aber dafür 30-40 Kunden enthalten. Es ist sehr schwierig, die Anzahl der zu Beginn einzufügenden Profile

	DB1			DB2			DB3		
Δ	ohne	1-dim	2-dim	ohne	1-dim	2-dim	ohne	1-dim	2-dim
1,0	56,7	13,6	0,2	57,0	12,8	0,1	58,8	14,1	0,2
0,75	57,3	16,9	0,4	56,7	17,0	0,4	59,7	17,1	0,4
0,5	107,9	158,1	3,4	109,9	148,3	4,8	110,8	155,3	4,2
	DB4			DB5			DB6		
Δ	ohne	1-dim	2-dim	ohne	1-dim	2-dim	ohne	1-dim	2-dim
1,0	58,1	13,6	0,1	56,8	13,2	0,03	59,9	13,6	0,2
0,75	57,2	16,8	0,3	56,1	16,7	0,3	60,1	16,8	0,7
0,5	112,1	144,0	4,1	108,0	179,2	4,8	113,7	146,5	6,3

Tabelle 4.8: Durchschnittliche maximale Anzahl Kandidaten mit und ohne Starttiles bei strikter Dichteanforderung

festzulegen, da dies nach Dichte der Datenbank, Δ und auch individuell nach den einzelnen Profilen variiert. Deshalb wird die eindimensionale Tilebildung als wenig sinnvoll eingestuft und sich im Folgenden auf die zweidimensionale Variante konzentriert.

Durch die Auswahl eines Starttiles werden zwar weniger Kandidaten betrachtet, aber dadurch werden natürlich auch die Generalisierungsmöglichkeiten eingeschränkt. Deshalb werden noch einmal die entstehenden Tiles hinsichtlich ihrer Fläche verglichen. In Tabelle 4.9 sind die durchschnittlichen Flächen bei strikter Dichteanforderung eingetragen. Für einen Schwellenwert von $\Delta = 100\%$ ergeben sich mit Verwendung von Starttiles sogar deutlich größere Tiles. Dies ist aber bei kleiner werdendem Δ nicht mehr der Fall. Zumindest scheint sich aber die Verwendung von Starttiles auch hier nicht negativ auf die Größe der Tiles auszuwirken.

	DB1		DB2		DB3		DB4		DB5	
Δ	ohne	2-dim	ohne	2-dim	ohne	2-dim	ohne	2-dim	ohne	2-dim
1,0	39	54	38	51	39	55	39	55	38	52
0,75	49	56	47	53	48	57	51	56	51	54
0,5	89	86	82	83	83	86	84	84	86	89
	DB6		DB7		DB8		DB9		DB10	
Δ	ohne	2-dim	ohne	2-dim	ohne	2-dim	ohne	2-dim	ohne	2-dim
1,0	37	52	38	55	39	52	38	51	42	55
0,75	49	54	50	56	49	54	47	53	50	56
0,5	86	89	81	83	82	78	81	80	86	87

Tabelle 4.9: Durchschnittliche Fläche φ der Tiles mit und ohne Starttiles bei strikter Dichteanforderung

Anders verhält sich dies bei der Verwendung von Starttiles bei relaxierter Dichteanforderung und Varianzmaß. Wie in Tabelle 4.10 zu sehen ist, sinkt hier die Fläche für hohe Δ deutlich. Dies ist zunächst erstaunlich, da die Tilegröße zumindest genauso groß sein sollte wie bei der strikteren Dichteanforderung. Die Ursache hierfür liegt im Varianzmaß, das benutzt wird, um die Profile auszuwählen, die aus dem Tile entfernt werden. Es wird das Profil entfernt, welches nach seiner Entfernung das Tile mit der kleinsten Varianz hinterlässt. Da jedoch bei der zweidimensionalen Starttilebildung zu Beginn ein Tile mit sehr

vielen negativen Kunden-/Produkt-Paaren vorausgeht, sodass $\delta(t)$ zum Beispiel nur noch 20% beträgt, werden konsequent Profile entfernt, die sehr viele positive Paare enthalten, da diese eine große Varianz erzeugen. In dieser Form ist die zweidimensionale Starttilebildung also eher kontraproduktiv. Eine Möglichkeit dies zu verhindern, wäre es, zum Erstellen des Starttiles die gleiche Methodik zum Herausstreichen zu wählen, wie es bei der strikten Dichteanforderung erfolgt. Dann ist aber nicht mehr die gleichmäßige Verteilung der negativen Paare gewährleistet. Da in Tabelle 4.9 zu sehen ist, dass, abgesehen von sehr hohen Δ, keine große Verbesserung in der Größe der Tiles zu erwarten ist, wird eine Kombination von relaxierter Dichteanforderung und Starttiles nicht mehr durchgeführt.

	DB1		DB2		DB3		DB4		DB5	
Δ	ohne	2-dim	ohne	2-dim	ohne	2-dim	ohne	2-dim	ohne	2-dim
1,0	39	23	38	24	39	24	39	24	38	25
0,75	59	48	57	49	59	54	60	49	58	49
0,5	209	219	227	220	214	215	250	236	219	219

Tabelle 4.10: Durchschnittliche Fläche φ der Tiles mit und ohne Starttiles bei relaxierter Dichteanforderung

4.8 Suche

Mittels Starttiles wird versucht, die große Anzahl an Kandidaten zu Beginn der Tilebildung zu handhaben. Ein Problem ist hier nicht nur die große Anzahl der Kandidaten an sich, sondern es ist auch oftmals der Fall, dass viele dieser Kandidaten denselben *maximalen* Heuristikwert aufweisen. In diesem Fall würde laut Algorithmus 4.2 immer der erste Kandidat gewählt, der diesen Wert aufweist. Andere Pfade, die eventuell zu einem größeren Tile führen könnten, werden nicht weiter verfolgt. Es handelt sich also um eine typische Hill-Climbing-Suche, die oft in einem lokalen statt dem globalen Optimum endet. Um dieses Problem einzudämmen, kann eine erweiterte Suche durchgeführt werden, die auch die anderen Pfade betrachtet. Es werden zwei mögliche Suchverfahren nach dem größten Tile um einen gegebenen Startpunkt vorgestellt. Die erste ist eine klassische Beam-Search; bei der zweiten handelt es sich um eine Kombination aus Beam-Search und der parallelen Ausführung mehrerer Hill-Climbing-Suchen.

4.8.1 Beam-Search

Bei einer Beam-Search wird nicht nur die aktuelle beste Lösung, sondern die besten b Lösungen betrachtet. Diese Lösungen werden weiter verfeinert und aus allen Verfeinerungen wiederum die b besten für den nächsten Schritt ausgewählt. Zum Finden eines möglichst großen Tiles wird also nicht nur eine Vergrößerung durch den besten Kandidaten eines Tiles betrachtet, sondern es werden alle Kandidaten herangezogen, die den maximalen Heuristikwert besitzen. Diese werden in das Tile eingefügt und aus allen so entstandenen Tiles die b besten für die nächste Iteration ausgewählt. Eigentlich ist es bei der Beam-Search üblich, nicht nur die Kandidaten mit maximalem Heuristikwert zuzulassen, sondern auch eventuell schlechtere Teillösungen zu betrachten, wenn weniger als b

Teillösungen zu dem Zeitpunkt vorhanden sind, um globale statt lokale Optima zu erreichen. Hier werden nur die Kandidaten mit maximalem Heuristikwert betrachtet, da dies erfahrungsgemäß bereits sehr viele sind.

Algorithmus 4.5 Beam-Search

```
 1: INPUT:  b          // Anzahl der Elemente in der Beam
 2:         t          // Tile bestehend aus Startpunkt
 3: OUTPUT: maxT       // größtes gefundenes Tile
 4: Q := ∅             // Menge der besten Lösungen im aktuellen Schritt
 5: Q' := ∅            // Menge der besten Lösungen für den nächsten Schritt
 6: T := ∅             // Menge der fertigen Tiles
 7: Q = Q ∪ {t}
 8: while Q ≠ ∅ do
 9:   for all t ∈ Q do
10:     C := findeBesteKandidaten(t)
11:     if C = ∅ then
12:       t = Nachbearbeitung(t)
13:       if t ∉ T then
14:         T = T ∪ {t}
15:       end if
16:     else
17:       for all c ∈ C do
18:         t' = t
19:         t' = t' + c
20:         if t' ∉ Q' then
21:           Q' = Q' ∪ {t'}
22:         end if
23:       end for
24:     end if
25:   end for
26:   Q' = BBeste(Q')
27:   Q = Q'
28:   Q' = ∅
29: end while
30: maxT = findeGroesstesTile(T)
```

In Algorithmus 4.5 ist der Ablauf der Beam-Search in Pseudocode dargestellt. Die Funktion *findeBesteKandidaten* in Zeile 10 gibt für ein Tile t alle Kandidaten zurück, die den maximalen Heuristikwert besitzen. Sind für ein Tile keine Kandidaten mehr vorhanden, wird in der *Nachbearbeitung* (Zeile 12) die Dichte der Profile überprüft und die Lösung in der Menge der fertigen Tiles T gespeichert, sofern sie nicht schon einmal errechnet wurde. Ansonsten werden nacheinander die einzelnen Kandidaten in das Tile eingefügt und in Q' gespeichert. Auch hier wird geprüft, ob die Teillösung bereits auf anderem Weg errechnet wurde. *BBeste* löscht alle Teillösungen aus Q' bis auf die b besten. Befinden sich keine Elemente mehr in der Beam, wird aus den fertigen Tiles das mit der größten Fläche als Lösung zurückgegeben. Die Teillösungen in Q' sind nach einem Gütekriterium geordnet. Dieses entscheidet, welche Teillösung besser als eine andere ist. Es wurde ein Kriterium implementiert, das sich ausschließlich auf die Größe der Tiles konzentriert. Hier fließt einerseits die aktuelle Größe des Tiles ein, andererseits wird auch die potentielle Größe, die das Tile erlangen könnte, miteinbezogen. Eine potentielle Größe des Tiles lässt sich aus der Anzahl der Kandidaten ermitteln.

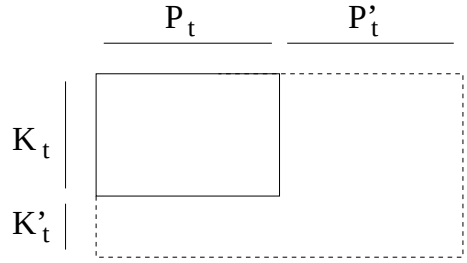


Abbildung 4.16: Ermitteln der potentiellen Größe eines Tiles

In Abbildung 4.16 ist ein Tile t mit der Fläche $\varphi(t) = |K_t| \cdot |P_t|$ gezeigt, wobei K_t die Menge der enthaltenen Kunden und P_t die Menge der enthaltenen Produkte ist. K'_t bzw. P'_t sind die Kunden- bzw. Produktnummern, die zum Einfügen in das Tile zur Verfügung stehen. Die potentielle Fläche des Tiles ist dann

$$\begin{aligned} \varphi_{pot}(t) &= (|K_t| + |K'_t|) \cdot (|P_t| + |P'_t|) \\ &= |K_t| \cdot |P_t| + |K'_t| \cdot |P_t| + |P'_t| \cdot (|K_t| + |K'_t|) \end{aligned}$$

Die beiden letzten Summanden bilden die mit Punkten umrandeten Fläche ohne den schon bestehenden Kern des Tiles. Da wahrscheinlich nicht alle Kandidaten in das Tile eingefügt werden können, sollten diese Summanden mit einem Konstanten Faktor c gewichtet werden:

$$\varphi_{pot}(t) = |K_t| \cdot |P_t| + c \cdot (|K'_t| \cdot |P_t| + |P'_t| \cdot (|K_t| + |K'_t|))$$

Wird c zu groß gewählt, wird eventuell die potentielle Größe durch die Anzahl der Kandidaten überbewertet; andererseits muss ein Tile mit aktuell größter Fläche nicht zwangsläufig auch zum Ende das größte Tile bilden. Ein Wert von $c = 0,3$ hat sich als gute Gewichtung herausgestellt.

Wird die Beam-Search so erweitert, dass auch Kandidaten betrachtet werden, die nicht den maximalen Heuristikwert besitzen, so müsste ein Gütekriterium gefunden werden, das nicht nur die potentielle Fläche der Tiles sondern auch seine Dichte bewertet.

4.8.2 Beam-Search und paralleles Hill-Climbing (BHC)

Wenn alle Elemente, die sich gerade in der Beam befinden, mit ihren besten Kandidaten erweitert werden, ergibt dies sehr schnell eine große Menge von Teillösungen, deren Anzahl die Beamgröße b meist deutlich überschreitet. Oft handelt es sich bei diesen Teillösungen um Tiles, die nur sehr wenig voneinander abweichen, dadurch dass dieselben Kandidaten in verschiedener Reihenfolge eingefügt werden. Ein Tile besteht zum Beispiel aus ({1,2,3},{A,B}), und als Kandidaten stehen {4,C} zur Verfügung, wogegen ein anderes Tile im gleichen Schritt aus ({1,2,4},{A,C}) mit den Kandidaten {3,B} besteht. Wahrscheinlich werden diese beiden Teillösungen im gleichen Tile resultieren. Um die Menge

der Teillösungen in jedem Schritt zu reduzieren, wird die Beam-Search hin zu einer parallelen Hill-Climbing-Suche auf mehreren Tiles abgewandelt. In die Beam werden nicht mehr alle möglichen Teillösungen aufgenommen und aus diesen die besten b ausgewählt, sondern es werden prinzipiell nur b Teillösungen errechnet. Dafür wird jedoch von jedem Element in der Beam mindestens ein Nachfolger in den nächsten Schritt weitergegeben. Dadurch gehen eventuell einige Teillösungen verloren, meistens handelt es sich jedoch um solche Lösungen, die unter das oben beschriebene Phänomen der unterschiedlichen Reihenfolge des Einfügens derselben Kandidaten fallen. Alle Tiles in der Beam werden parallel erweitert, sodass b Hill-Climbing-Suchen parallel durchgeführt werden. Die in diesem Algorithmus 4.6 verwendeten Funktionen sind die gleichen, wie sie bereits in der Beam-Search (Algorithmus 4.5) verwendet wurden. Nacheinander wird von jedem Kandidaten in der aktuellen Beam versucht, einen Nachfolger durch Einfügen eines Kandidaten zu finden, der bisher noch nicht errechnet wurde. Gegebenenfalls können auch von einem Tile mehrere Nachfolger für die nächste Iteration gebildet werden. Die Suche ist auch hier abgeschlossen, wenn keine Nachfolger mehr gefunden werden können. Aus den fertigen Tiles wird auch hier das größte als Ergebnis ermittelt.

Algorithmus 4.6 Beam-Search und paralleles Hill-Climbing

```
1: INPUT:  b            // Anzahl der Elemente in der Beam
2:         t            // Tile bestehend aus Startpunkt
3: OUTPUT: maxT         // größtes gefundenes Tile
4: Q := ∅               // Menge der besten Lösungen im aktuellen Schritt
5: Q' := ∅              // Menge der besten Lösungen für den nächsten Schritt
6: T := ∅               // Menge der fertigen Tiles
7: Q = Q ∪ {t}
8: while Q ≠ ∅ do
9:    for all t ∈ Q do
10:      C_t := findeBesteKandidaten(t)
11:      if C_t = ∅ then
12:         t = Nachbearbeitung(t)
13:         if t ∉ T then
14:            T = T ∪ {t}
15:         end if
16:      end if
17:   end for
18:   while |Q'| < b do
19:      for all t ∈ Q do
20:         if |Q'| < b then
21:            t' = t
22:            if ∃c ∈ C_t mit (t' + c) ∉ Q' then
23:               Q' = Q' ∪ {(t' + c)}
24:            else
25:               continue
26:            end if
27:         end if
28:      end for
29:   end while
30:   Q = Q'
31:   Q' = ∅
32: end while
33: maxT = findeGroesstesTile(T)
```

4.8.3 Evaluation

Nun soll untersucht werden, ob durch die Suchverfahren bessere Ergebnisse, also größere Tiles, gefunden werden können. Dazu werden zunächst die durchschnittlichen Tilegrößen ohne Suche und mit Beam-Search verglichen. Die Beam-Search wurde mit zwei unterschiedlichen Werten für den Parameter b getestet. In Tabelle 4.11 sind die Ergebnisse aufgelistet. Parameter $b = 1$ entspricht der normalen Generalisierungsstrategie, die bisher benutzt wurde. Auch hier wurden nur die Ergebnisse für die ersten sechs Datenbanken aufgelistet.

	DB1			DB2			DB3		
Δ	b=1	b=20	b=50	b=1	b=20	b=50	b=1	b=20	b=50
1,0	39	46	46	38	43	43	39	46	46
0,75	49	52	52	47	50	50	48	53	53
0,5	89	95	97	82	92	93	83	91	92
	DB4			DB5			DB6		
Δ	b=1	b=20	b=50	b=1	b=20	b=50	b=1	b=20	b=50
1,0	39	46	46	38	47	47	37	44	44
0,75	51	53	53	51	52	53	49	51	51
0,5	84	94	96	86	92	94	86	95	97

Tabelle 4.11: Durchschnittliche Fläche φ der Tiles für Beam-Search mit unterschiedlichen Werten des Parameters b

Mittels Beam-Search wurden etwas bessere Ergebnisse erzielt, als bei alleiniger Betrachtung eines besten Kandidaten. Dabei scheint $b = 20$ ausreichend zu sein, da mit $b = 50$ nur für $\Delta = 0,5$ geringe Verbesserungen erreicht wurden.

Die Beam-Search wurde mit einem parallelen Hill-Climbing-Ansatz kombiniert, da, wie bereits beschrieben wurde, viele Lösungswege lediglich durch andere Einfügereihenfolgen entstehen. In Tabelle 4.12 ist zu sehen, dass mit diesem Ansatz genauso gute oder sogar bessere Ergebnisse erzielt werden. Die Laufzeit ist dafür abhängig von Δ etwa drei- bis sechsmal geringer. Beam-Search mit Hill-Climbing-Ansatz ist also sowohl anhand der Ergebnisse als auch bezüglich der Laufzeit vorzuziehen.

	DB1		DB2		DB3	
Δ	B (20)	BHC (20)	B (20)	BHC (20)	B (20)	BHC (20)
1,0	46	46	44	43	46	46
0,75	52	52	50	50	53	53
0,5	98	95	92	92	96	91
	DB4		DB5		DB6	
Δ	B (20)	BHC (20)	B (20)	BHC (20)	B (20)	BHC (20)
1,0	46	46	47	47	44	44
0,75	53	53	53	53	52	51
0,5	97	94	95	92	96	95

Tabelle 4.12: Durchschnittliche Fläche φ der Tiles für Beam-Search (B) und Beam-Search mit parallelem Hill-Climbing (BHC) (b = 20)

Die Suche kann auch mit Starttiles kombiniert werden. Für große Δ ergeben sich allerdings keine merklichen Verbesserungen, da nach Erstellen eines Starttiles kaum Kandidaten zur Verfügung stehen, wie im vorigen Abschnitt gezeigt wurde. Für kleine Δ wirken sich Starttiles eher negativ aus, da hier viele Lösungspfade weggeschnitten werden.

Für die relaxierte Dichteanforderung ergeben sich ähnliche Werte. Beam-Search mit parallelem Hill-Climbing liefert genauso gute bzw. bessere Werte als die reine Beam-Search. Die durchschnittlichen Flächen steigen gegenüber keiner Verwendung einer Suchstrategie an. Eine Kombination mit Starttiles wurde bereits ausgeschlossen.

Kapitel 5

Ergebnisse

Zwischen den einzelnen Abschnitten wurden bereits Auswertungen bezüglich der Auswahl verschiedener Heuristiken vorgenommen. Zwar muss die Kombination der so auswählten Heuristiken nicht zwangsläufig die Beste sein, eine Evaluation aller Kombinationen der Heuristiken wäre jedoch zu aufwendig.

An dieser Stelle soll der Algorithmus mit dem gewählten Parametersatz noch einmal hinsichtlich der gefundenen Tilings, der Laufzeit etc. überprüft werden. Dazu werden zunächst drei Testreihen auf künstlichen Datenbanken durchgeführt, wobei mittels der ersten Testreihe die Tilings untersucht werden und mittels der weiteren beiden die Laufzeit des Algorithmus überprüft wird. Anschließend findet noch einmal eine Evaluation auf der MovieLens-Datenbank [25] statt. Es werden zwei Parametersätze unterschieden. Bei dem ersten handelt es sich um die verwendete Heuristik Weighted Accuracy mit anschließender Verfeinerung durch maxPos und maxKand. Dabei liegt die ursprüngliche, striktere Dichteanforderung zugrunde. Der zweite Parametersatz verwendet die relaxierte Dichteanforderung mit Varianzmaß und Verfeinerung durch maxPos und maxKand. Bei beiden werden Startpunkte durch das Round-Robin-Verfahren auf Kundennummern ausgewählt. Als mögliches Suchverfahren wird Beam-Search mit Hill-Climbing (BHC) verwendet. Eine Verwendung von Starttiles ist bei der strikten Dichteanforderung möglich. BHC und Starttiles werden gesondert betrachtet.

Die Berechnungen wurden auf einem AMD Opteron mit 2.4GHz und 2GB Hauptspeicher durchgeführt. Die Implementation erfolgte in C++. Zur Darstellung der Datenbank wurde die Sparse-Matrix Implementation der uBLAS-Template-Bibliothek verwendet, die unter *www.boost.org* frei erhältlich ist.

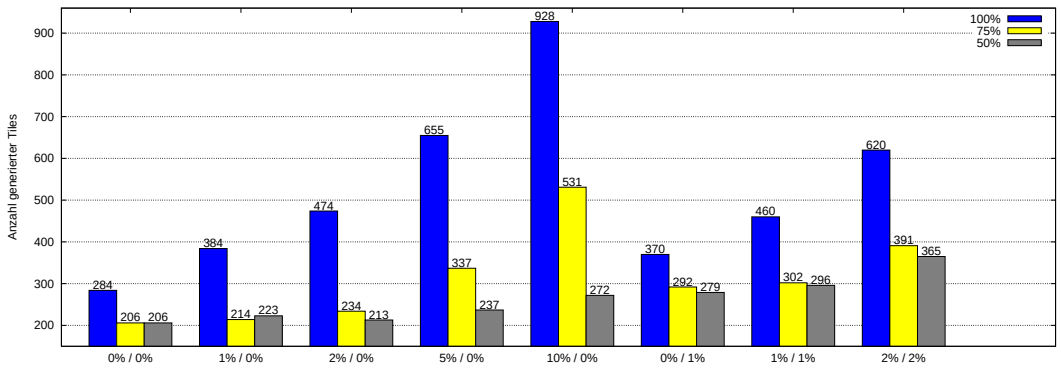
5.1 Tilings

Bisher wurden immer künstliche Datenbanken betrachtet, die vollständige Tiles, also mit einer Dichte von 100%, enthielten. In realen Daten werden allerdings wahrscheinlich keine vollständigen Tiles zu erwarten sein. Deshalb soll eine solche Situation simuliert werden, indem Rauschen in die künstlichen Datenbanken eingestreut wird. Dabei können sowohl positive als auch negative Kunden-/Produkt-Paare gestört werden. Hier wird die Wirkung

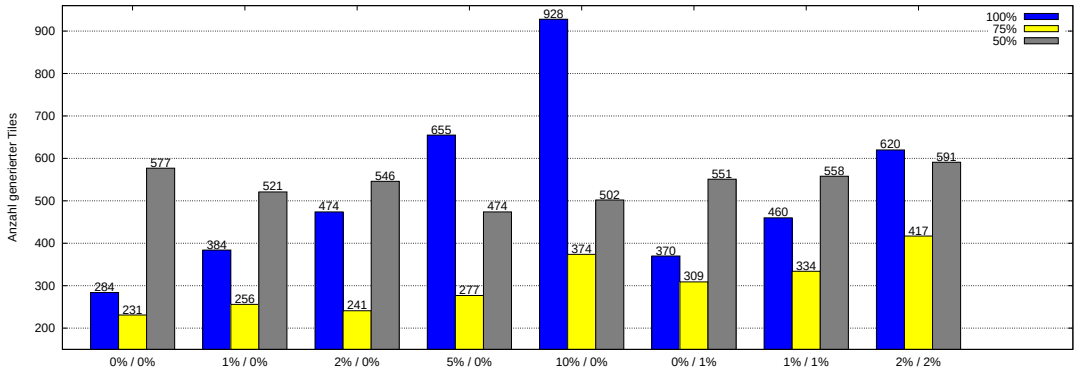
der Mindestdichte Δ deutlich. Sinkt Δ, so sollte es trotz Rauschen möglich sein, weiterhin entsprechend große Tiles zu finden. Von Interesse ist hier vor allem die durchschnittliche Fläche der Tiles, die erreicht wird, sowie die Anzahl der Tiles, die generiert werden und wie viele der ursprünglichen Tiles vom gefundenen Tiling überdeckt werden.

Wie zuvor wurde eine Datenbank mit 300 Kunden und Produkten, einer Dichte von 10% und 150 enthaltenen Tiles gewählt. In die Datenbank wurde dann zunächst nur in die positiven, anschließend auch in die negativen Kunden-/Produkt-Paare Rauschen eingefügt. Dazu wurde einfach ein fester Prozentsatz (immer gemessen an den positiven Paaren) der Paare umgedreht. Das Rauschen in den zuvor positiven Paaren sorgt dafür, dass die Dichte der enthaltenen Tiles sinkt. Für $\Delta = 1,0$ können diese Tiles dann nicht mehr gefunden werden. Auch für kleinere Werte von Δ kann es sein, dass Tiles nicht gefunden werden, weil Kunden- oder Produktprofile nicht mehr der Dichteanforderung genügen. Das Rauschen in den zuvor negativen Paaren ist insofern interessant, zu sehen, wie es gelingt, die nun hinzukommenden Paare in vorhandene Tiles einzubinden, oder ob für die Paare sehr viel mehr zusätzliche Tiles gebildet werden müssen.

Als erstes soll die Anzahl der generierten Tiles betrachtet werden. In Abbildung 5.1(a)



(a) Strikte Dichteanforderung

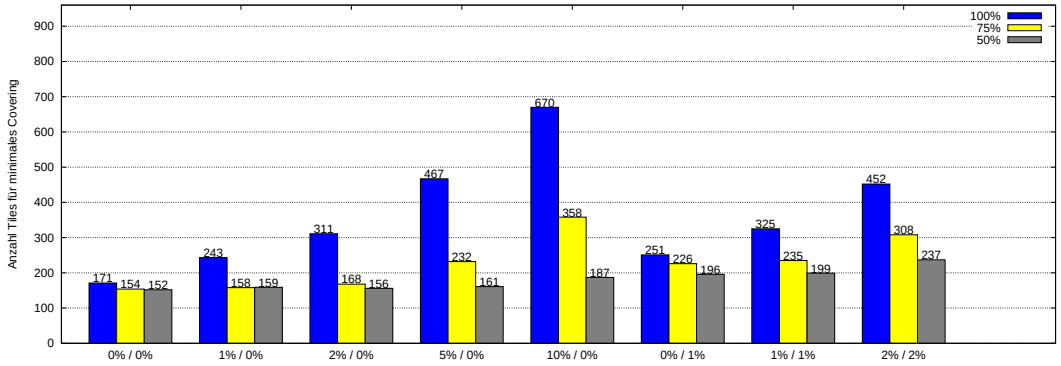


(b) Relaxierte Dichteanforderung

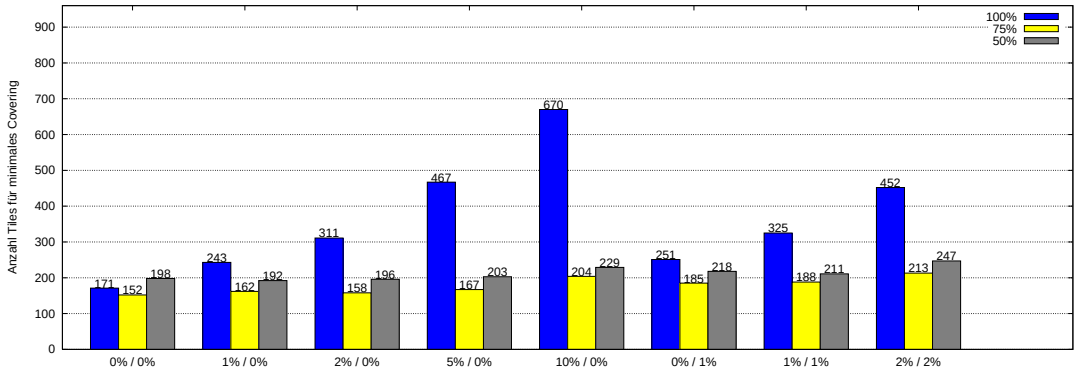
Abbildung 5.1: Anzahl generierter Tiles

bzw. 5.1(b) ist die Anzahl der Tiles für die strikte bzw. relaxierte Dichteanforderung für unterschiedliche Werte des eingefügten Rauschens aufgeführt. Auf der x-Achse der Diagramme sind die unterschiedlichen Datenbanken mit dem jeweils eingefügten Rauschen in positiven (1. Wert) und negativen Paaren (2. Wert) zu sehen. Zum Vergleich enthält die erste Datenbank kein Rauschen. Die Anzahl der generierten Tiles ist auf der y-Achse aufgetragen, wobei die Skala bei 150, der Anzahl der ursprünglich in der Datenbank enthaltenen Tiles, beginnt.

Zunächst sollen die Datenbanken betrachtet werden, die Rauschen in den positiven Paaren enthalten. Wie erwartet steigt die Anzahl der generierten Tiles für eine geforderte Dichte von $\Delta = 100\%$ für strikte und relaxierte Dichteanforderung stark an, da bei eingefügtem Rauschen die enthaltenen Tiles nicht mehr zulässig sind. Für $\Delta = 75\%$ kann geringes Rauschen sehr gut kompensiert werden; ein Rauschen von 5% und 10% lässt aber auch hier die Tileanzahl ansteigen. Bei einer Durchschnittsfläche der originalen Tiles von 60 ist ein Rauschen von 10% für die gegebenen Dichteanforderungen eine sehr starke Störung. Bei $\Delta = 50\%$ kann das Rauschen von Weighted Accuracy in Kombination mit der strikten Dichteanforderung noch relativ gut abgefangen werden. Verwunderlich sind auf den ersten Blick die großen Anzahlen an Tiles, die bei der relaxierten Dichteanforderung erzeugt



(a) Strikte Dichteanforderung

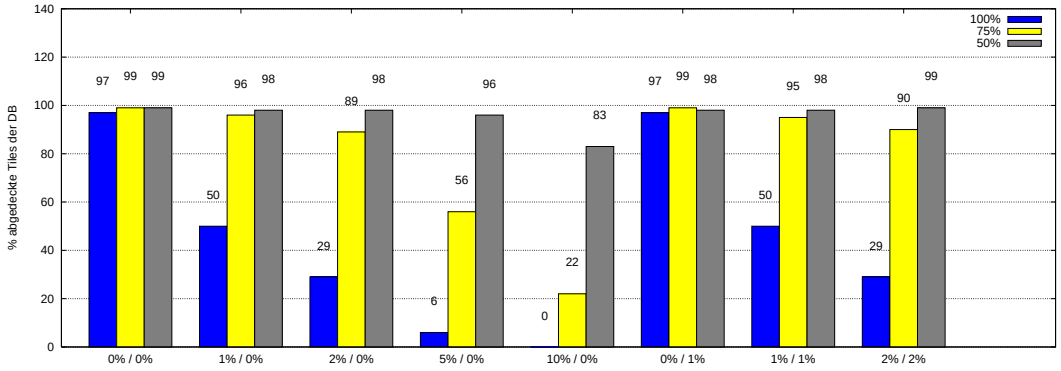


(b) Relaxierte Dichteanforderung

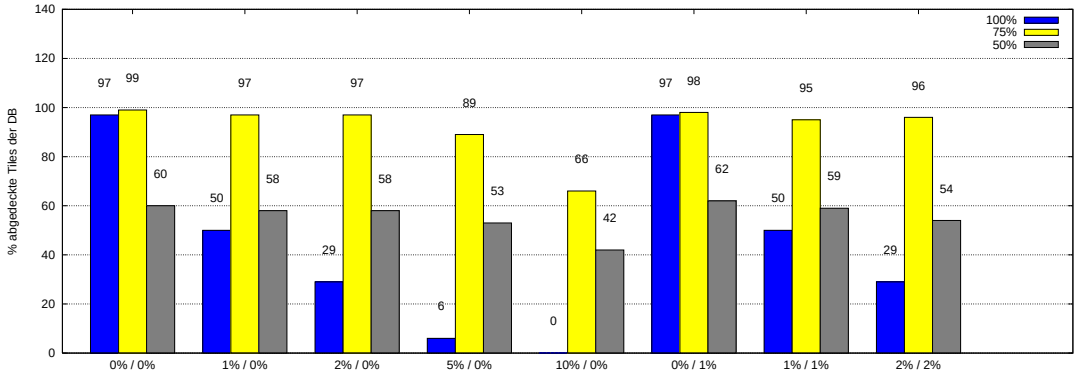
Abbildung 5.2: Anzahl Tiles im minimalen Tiling

wurden. Hier werden eigentlich, wie später noch gezeigt wird, wesentlich größere Tiles erzeugt, und damit sollten weniger Tiles genügen, um die Datenbank abzudecken. Die Ursache hierfür liegt hauptsächlich im Varianzmaß, das die Dichte der Tiles sehr nahe an Δ heranführt. Die Tiles werden zwar sehr groß, haben aber im Schnitt tatsächlich nur 50% Dichte. Die Dichte bei strikter Dichteanforderung liegt im Schnitt bei 80%. Gepaart mit vielen Überschneidungen der Tiles untereinander, führt diese niedrige Dichte zu der hohen Anzahl generierter Tiles.

Bei den Datenbanken mit Rauschen in negativen Paaren sind ähnliche Effekte festzustellen. In der Datenbank mit 0% Rauschen in positiven und 1% Rauschen in den negativen Paaren sind 90 zusätzliche positive Paare eingefügt worden, die nicht in Verbindung mit den enthaltenen Tiles stehen. An den Zahlen ist ersichtlich, dass diese erst bei geringen Werten von Δ in die anderen Tiles integriert werden können. Betrachtet man die Tiles näher, so kann man erkennen, dass die positiven Paare aus dem Rauschen als Startpunkte nur kleine Tiles ergeben. Hier müsste also noch eine Strategie entwickelt werden, die eine Überanpassung an solche vereinzelten Punkte vermeidet. Dann würde zwar nicht mehr die gesamte Datenbank überdeckt, aber vom Standpunkt des Informationsgehaltes kann auf solche Tiles verzichtet werden.



(a) Strikte Dichteanforderung

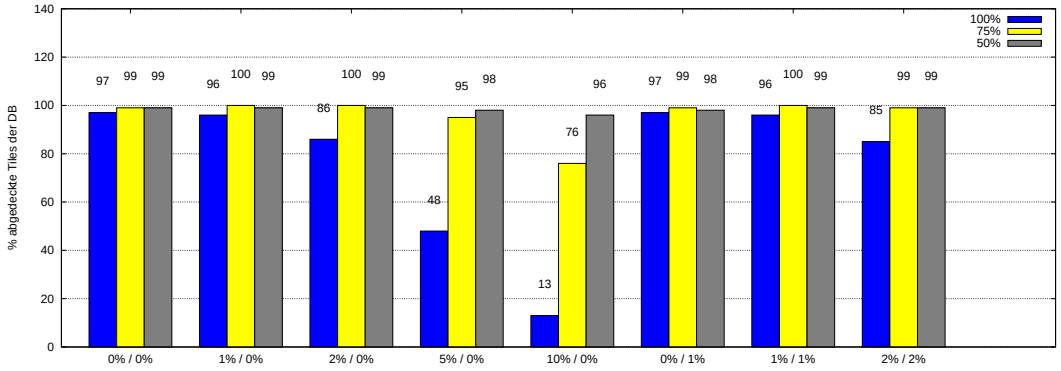


(b) Relaxierte Dichteanforderung

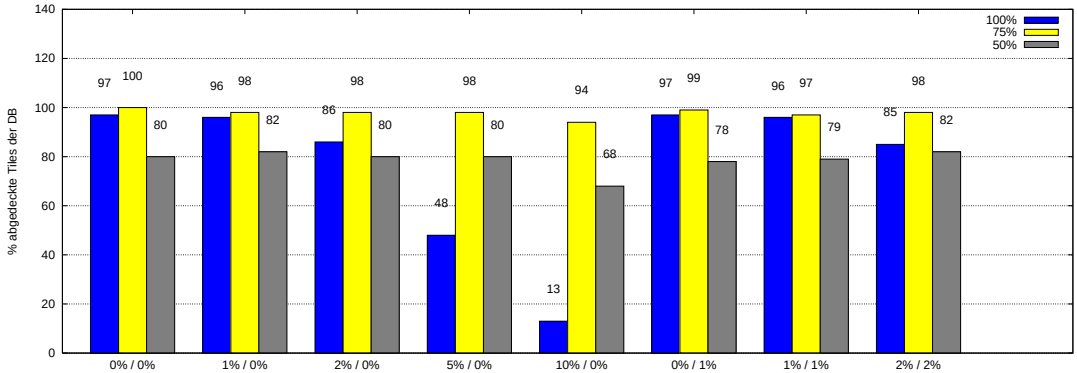
Abbildung 5.3: Prozentsatz der abgedeckten originalen Tiles bei vollständiger Überdeckung

In Abbildung 5.2(a) und 5.2(b) wird gezeigt, wie viele der generierten Tiles für ein minimales Tiling nötig sind. Fast immer sind deutlich weniger der generierten Tiles notwendig. Bei Varianzmaß und $\Delta = 50\%$ wird dies besonders deutlich. Für geringe Werte von Δ werden kaum mehr Tiles benötigt, als tatsächlich in der Datenbank vorhanden sind. Dies ist ein Zeichen dafür, dass eine verbesserte Strategie zur Auswahl der Startpunkte benötigt wird, da sehr viele fast redundante Tiles erzeugt werden. Bei Rauschen in den negativen Paaren sind erneut wesentlich mehr Tiles notwendig, was die Notwendigkeit einer Strategie zur Vermeidung von Überanpassung unterstreicht.

Der Prozentsatz der überdeckten originalen Tiles der Datenbank ist in Abbildung 5.3(a) bzw. 5.3(b) dargestellt. Gemessen wurde dabei eine Überdeckung von 100%. Ohne Rauschen werden mit Heuristik Weighted Accuracy fast alle Tiles wiedergefunden. Dann nimmt der Prozentsatz der überdeckten Tiles für $\Delta = 100\%$ und $\Delta = 75\%$ schnell ab, da aufgrund der Dichteanforderungen die originalen Tiles nicht mehr gültig sind. Auffällig ist wieder, dass sich die Tiles bei Varianzmaß und relaxierter Dichteanforderung anders 'entwickeln', als es bei der Weighted Accuracy erfolgt und so auch weniger Tiles wiedergefunden werden.



(a) Strikte Dichteanforderung



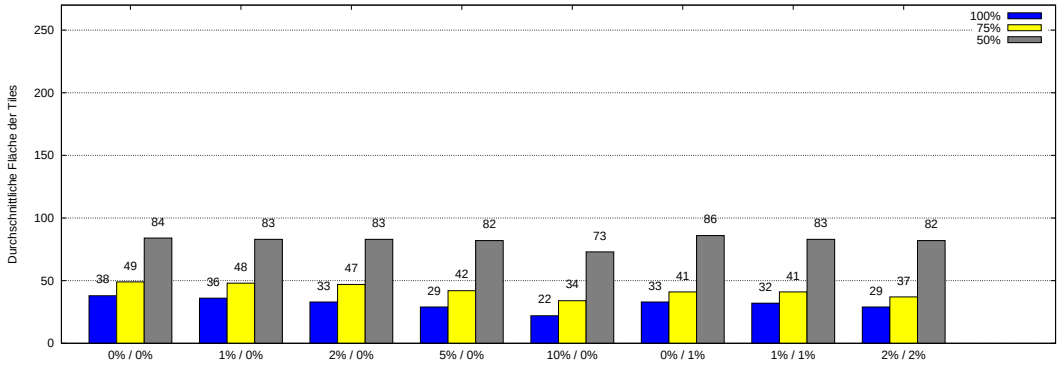
(b) Relaxierte Dichteanforderung

Abbildung 5.4: Prozentsatz der abgedeckten originalen Tiles bei 75% Überdeckung

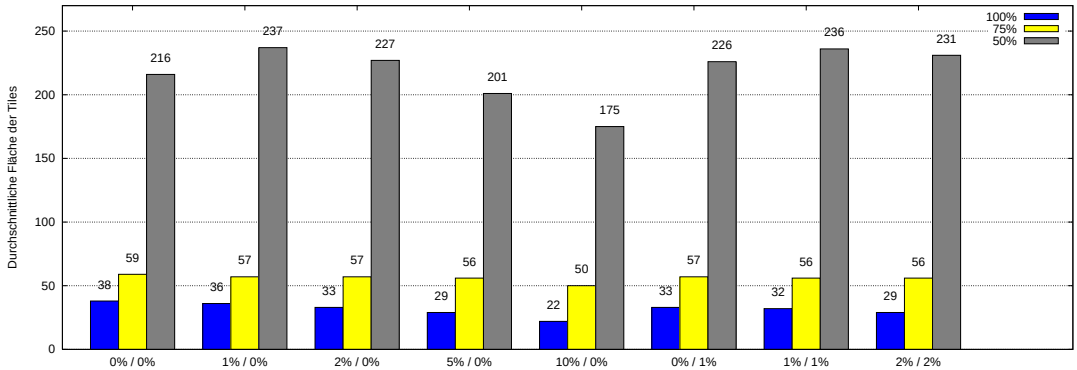
Fordert man allerdings, dass die originalen Tiles nur zu 75% überdeckt sein müssen (siehe Abbildung 5.4(a) und 5.4(b)), um als 'gefunden' zu gelten, so steigt die Anzahl der abgedeckt Tiles wieder deutlich an, und es wird deutlich, dass der Algorithmus auch bei stärkerem Rauschen in der Lage ist, die ursprüngliche Information wiederzugeben.

Betrachtet man schließlich noch in 5.5(a) und 5.5(b) die durchschnittliche Fläche der Tiles, so verhalten sich diese erwartungsgemäß. Je größer Δ gewählt wird, desto größere Flächen ergeben sich. Die original enthaltenen Tiles haben eine Fläche von 60. Mit steigendem Rauschen in den positiven Paaren fällt die Fläche langsam ab. Wenn man die Datenbank ohne Rauschen und jene mit 1% Rauschen in negativen Paaren betrachtet, so verringert sich die durchschnittliche Fläche auf der Datenbank mit Rauschen etwas, da die hinzukommenden positiven Paare nicht in große Tiles eingebunden werden können. Die durchschnittliche Fläche ist bei dem Tiling mit relaxierter Dichteanforderung bei einer Dichte nahe an Δ deutlich höher.

Anhand der Fläche der Tiles soll auch nochmal das Ergebnis der Suche BHC verglichen werden (siehe Abbildung 5.6(a) und 5.6(b)). Überall kann ein verbesserter Wert erreicht werden, allerdings ist der Flächenanstieg nicht immer besonders stark, sodass zusammen



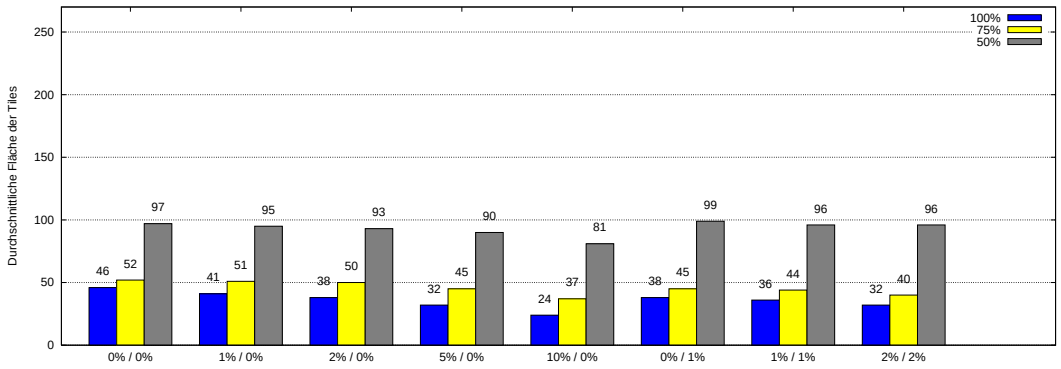
(a) Strikte Dichteanforderung



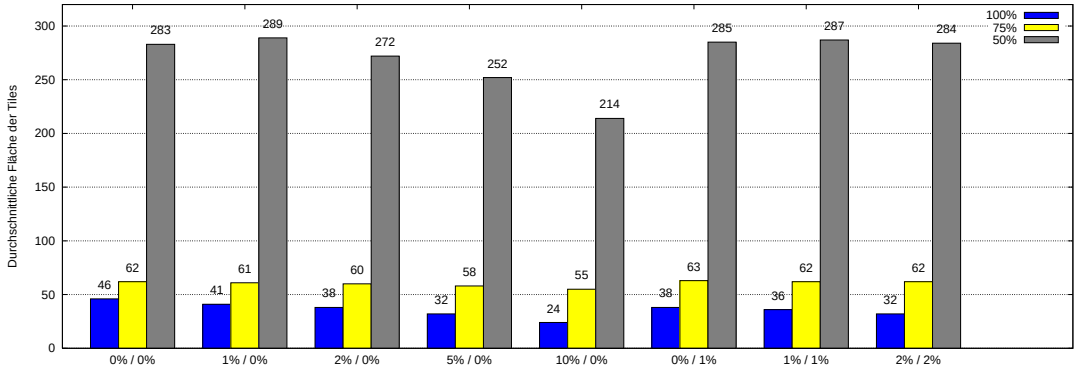
(b) Relaxierte Dichteanforderung

Abbildung 5.5: Durchschnittliche Fläche der Tiles

mit der Betrachtung der Laufzeit im nächsten Abschnitt überlegt werden muss, ob sich ihr Einsatz lohnt.



(a) Strikte Dichteanforderung



(b) Relaxierte Dichteanforderung

Abbildung 5.6: Durchschnittliche Fläche der Tiles bei Suche mit BHC

Starttiles statt Startpunkten wurden nur mit Weighted Accuracy kombiniert. In Abbildung 5.7 ist zu sehen, dass diese nur für große Δ eine Verbesserung bringen. Bei $\Delta = 50\%$ werden die durchschnittlichen Flächen wesentlich geringer im Vergleich zu denen, die mit Weighted Accuracy ohne Starttiles gebildet werden, da hier viele Generalisierungsmöglichkeiten durch das feste Starttile verloren gehen. Deshalb ist der Einsatz von Starttiles nur für große Δ angebracht.

5.2 Laufzeitbetrachtungen

In diesem Abschnitt soll das Laufzeitverhalten der Heuristiken bei strikter und relaxierter Dichteanforderung, sowie ohne und mit Suche mit BHC untersucht werden. Dafür wurde zunächst die Größe der Datenbank bei gleicher Dichte und anschließend die Dichte der Datenbank bei gleichbleibender Größe variiert.

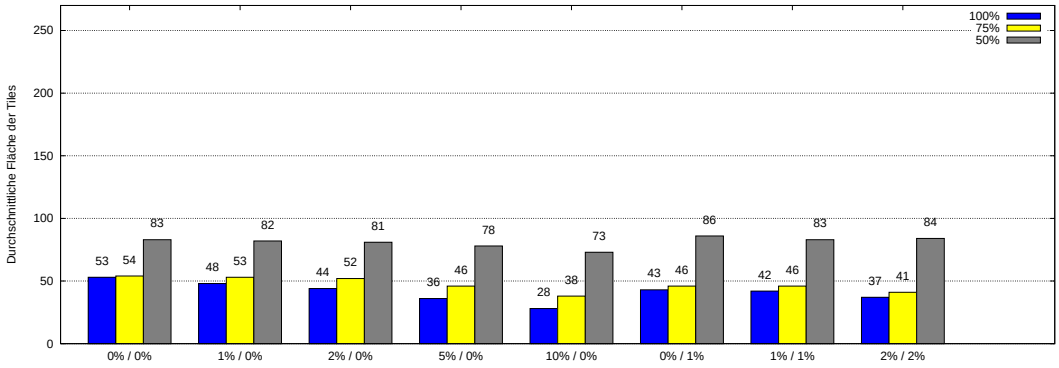


Abbildung 5.7: Durchschnittliche Fläche bei strikter Dichteanforderung mit Starttiles

5.2.1 Variation der Datenbankgröße

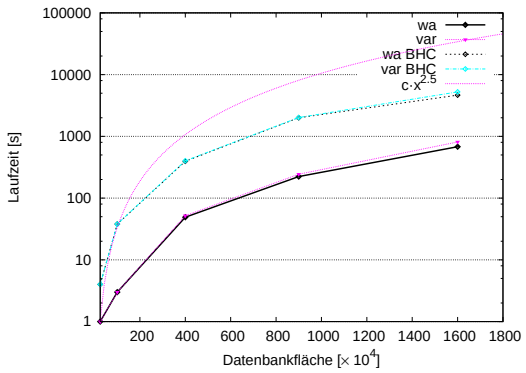
Die hier verwendeten Datenbanken haben alle eine Dichte von 2,5% und ein geringes Rauschen von 1% in den positiven Kunden-/Produkt-Paaren. Dabei haben die Datenbanken gleich viele Kunden wie Produkte und die Anzahl der eingefügten Tiles wird so erhöht, dass eine gleichbleibende Durchschnittsfläche von 60 vorliegt. In Abbildung 5.8 ist die Laufzeit des Algorithmus für verschiedene Datenbankgrößen aufgetragen. Für einen Wert von Δ lassen sich die Laufzeiten für Weighted Accuracy und Varianz vergleichen.

Um kleine Laufzeiten bei niedrigen Datenbankdichten sichtbar zu machen, wurden die y-Achsen der Diagramme logarithmisch dargestellt. Die Laufzeit des Algorithmus steigt mit wachsender Datenbankgröße stark an. Dies liegt an der großen Anzahl Kandidaten, die in jedem Schritt untersucht werden müssen. Zum Vergleich wurde in den Diagrammen zusätzlich die Funktion $x^{2,5}$ geplottet. Die Laufzeit des Algorithmus wird nach oben von dieser Funktion begrenzt. Dies entspricht den intuitiven Erwartungen an die Laufzeit des Algorithmus. Ist n die Anzahl der Kunden und Produkte, so stehen in jedem Generalisierungsschritt maximal n Kandidaten zur Verfügung. Zum Einfügen eines Kandidaten und zur Evaluation der Heuristik maxKand muss ein Vergleich des Profils des eingefügten Kandidaten mit den bereits enthaltenen Profilen bzw. mit anderen Kandidaten erfolgen, was zu quadratischer Laufzeit führt. Bei maximal n Generalisierungsschritten führt dies zu einer Laufzeit in $O(n^3)$ für das Finden eines Tiles. Schließlich muss dies potentiell für jedes positive Paar der Datenbank durchgeführt werden, was insgesamt zu der gezeigten Laufzeit in $O(n^5)$ führt.

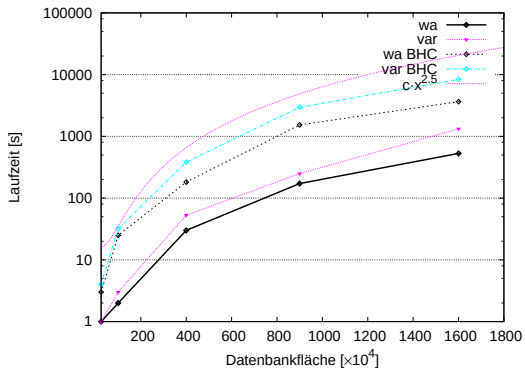
Durch das Überprüfen der relaxierten Dichteanforderung nach jedem Generalisierungsschritt ist die Laufzeit unter Verwendung der Varianz wesentlich höher als unter Weighted Accuracy. Die Suche mit BHC lässt die Laufzeit um einen konstanten Faktor ansteigen. Insgesamt steigt die Laufzeit mit fallendem Δ an, da mehr Generalisierungsschritte durchgeführt werden. Für $\Delta = 50\%$ ist der Unterschied zwischen Varianz und Weighted Accuracy am größten, da die Varianz die durchschnittliche Dichte der Tiles bis auf Δ absinken lässt und so weitaus größere Tiles erzeugt, wodurch natürlich auch die Laufzeit ansteigt.

Insgesamt lässt sich sagen, dass sich die Laufzeit zumindest unter Verwendung der Weighted Accuracy in einem akzeptablen Rahmen bewegt, sodass noch weitaus größere Datenbanken untersucht werden könnten. Sind größtmögliche Tiles erwünscht, lässt sich aber

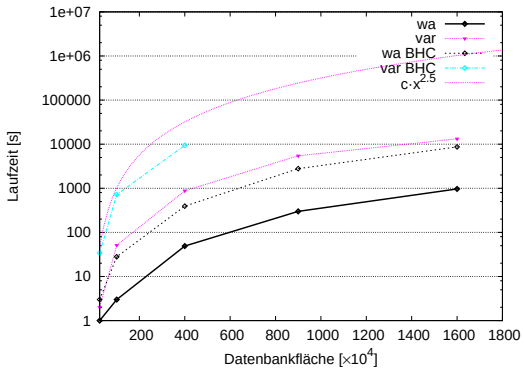
auch die Varianz durchaus einsetzen. Die Verbesserungen durch BHC sind gegenüber der Laufzeit abzuwägen.



(a) $\Delta = 1,0$



(b) $\Delta = 0,75$

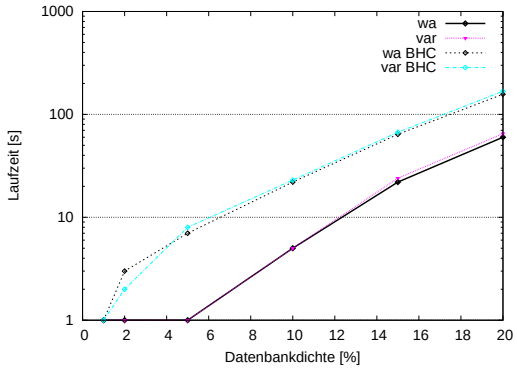


(c) $\Delta = 0,50$

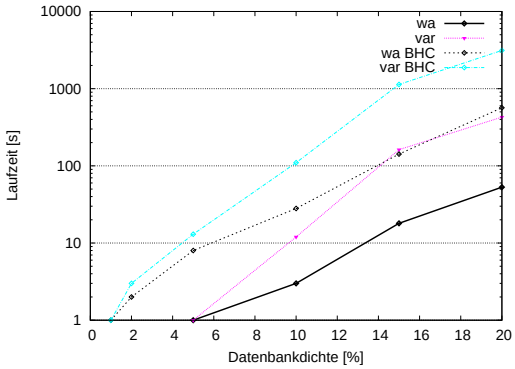
Abbildung 5.8: Laufzeit bei steigender Datenbankgröße

5.2.2 Variation der Dichte

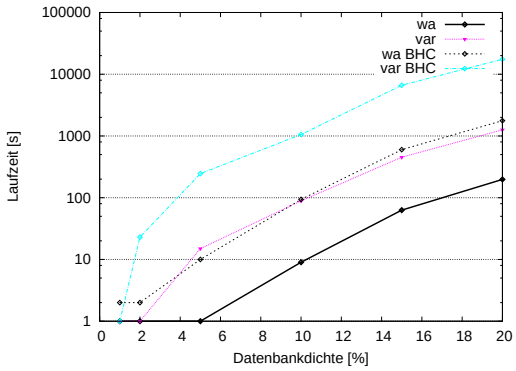
Die hier betrachtete Datenbank enthält 500 Kunden und 500 Produkte. Schrittweise wird die Dichte der Datenbank erhöht, wobei auch hier die Durchschnittsgröße der Tiles bei 60 liegt. Abbildung 5.9 dient noch einmal zur Verdeutlichung des Einflusses der Größe der Profile auf die Anzahl der Kandidaten und damit auch auf die Laufzeit des Algorithmus. Abbildung 5.10 vergleicht noch einmal die Laufzeit bei Verwendung der Weighted Accuracy mit und ohne Starttiles. Die Berechnung der Starttiles lässt die Laufzeit vor allem bei hohen Dichten stark ansteigen. Dies liegt daran, dass zunächst ein Tile aus allen Produkt- bzw. Kundennummern der Profile des Startpunktes gebildet wird. Dieses Tile ist aufgrund der Dichteanforderung meistens unzulässig. Je höher die Datenbankdichte und je höher Δ ist, desto mehr Kunden und Produkte müssen wieder aus dem Tile entfernt werden und die Laufzeit steigt rapide an.



(a) $\Delta = 1,0$

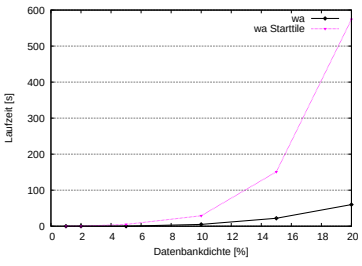


(b) $\Delta = 0,75$

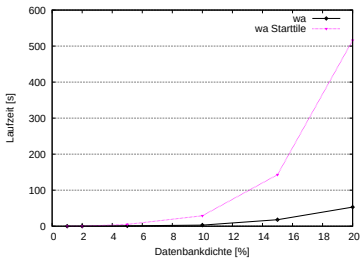


(c) $\Delta = 0,50$

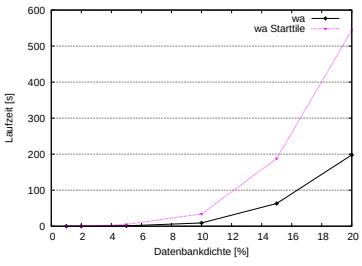
Abbildung 5.9: Laufzeit bei steigender Datenbankdichte



(a) $\Delta = 1,0$



(b) $\Delta = 0,75$



(c) $\Delta = 0,50$

Abbildung 5.10: Laufzeit bei steigender Datenbankdichte - Vergleich Starttiles

5.3 Abdeckung der Datenbank

In Abschnitt 4.2 wurde angesprochen, dass eventuell bei der Überprüfung der Profildichten im Nachbearbeitungsschritt die Startpunkte aus dem Tile entfernt werden. Werden diese dann nicht von einem anderen Tile überdeckt, so sind nicht 100% der Datenbank durch das Tiling abgedeckt. Hier soll noch einmal kurz demonstriert werden, dass nur sehr selten positive Paare der Datenbank nicht im Tiling enthalten sind. In Tabelle 5.1 und 5.2 sind die Prozentsätze der abgedeckten positiven Paare für die Datenbanken aus Abschnitt 5.1 aufgeführt. Fast immer werden nahezu 100% der positiven Paare abgedeckt, wodurch das Vorgehen in der Nachbearbeitung gerechtfertigt ist.

Δ	0%/0%	1%/0%	2%/0%	5%/0%	10%/0%	1%/1%	1%/1%	2%/2%
0,75	99.99	100.00	99.97	99.89	99.80	99.85	99.90	99.79
0,5	99.99	99.97	99.97	99.97	99.75	99.65	99.55	99.42

Tabelle 5.1: Prozentsatz der abgedeckten positiven Kunden-/Produkt-Paare der Datenbank bei Verwendung von Weighted Accuracy und strikter Dichteanforderung

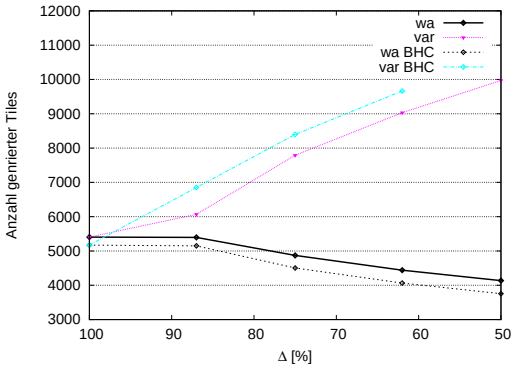
Δ	0%/0%	1%/0%	2%/0%	5%/0%	10%/0%	1%/1%	1%/1%	2%/2%
0,75	99.98	99.86	99.88	99.71	98.99	99.26	99.09	98.37
0,5	96.94	97.19	97.28	97.67	97.52	97.14	97.00	97.07

Tabelle 5.2: Prozentsatz der abgedeckten positiven Kunden-/Produkt-Paare der Datenbank bei Verwendung von Varianz und relaxierter Dichteanforderung

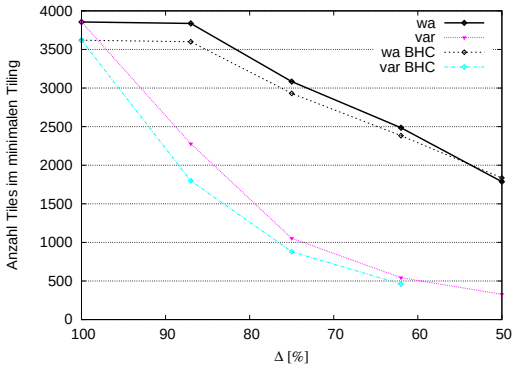
5.4 MovieLens

Abschließend wird der Algorithmus noch einmal auf einer Datenbank mit realen Daten getestet. Die in Abschnitt 2.4 eingeführte MovieLens-Datenbank enthält Bewertungen von Filmen. Insgesamt haben 943 Nutzer 1682 Filme auf einer Skala von 1 bis 5 bewertet. Nun soll herausgefunden werden, welche Filme von welchen Nutzern besonders gerne angeschaut werden. Deshalb wurden alle Bewertungen mit der höchsten Wertung von 5 herangezogen. Auf der folgenden Seite sind in Abbildung 5.11 die Ergebnisse des Algorithmus aufgeführt. Die Ergebnisse der relaxierten Dichteanforderung mit der Suche BHC fehlen aufgrund der hohen Laufzeit für $\Delta = 50\%$.

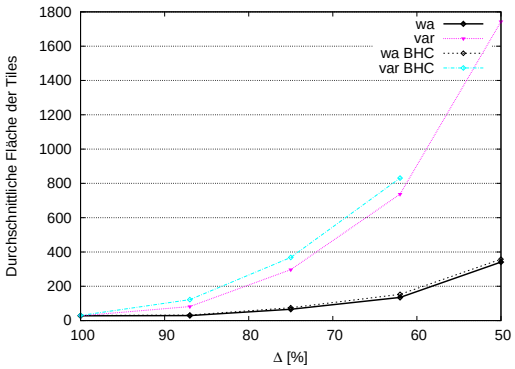
Die Beobachtungen auf den künstlichen Datenbanken lassen sich auch auf die MovieLens-Datenbank übertragen. Auffallend ist die große Anzahl an Tiles, die bei relaxierter Dichteanforderung mit fallendem Δ generiert werden. Hier scheinen sehr viele große Überschneidungen der Tiles aufzutreten, da im minimalen Tiling dagegen nur sehr wenige Tiles benötigt werden, um die Datenbank abzudecken. Die kleinste erreichte Anzahl an Tiles im minimalen Tiling liegt bei 328 Tiles. Dieser Wert wurde bei relaxierter Dichteanforderung ohne Suche und $\Delta = 50\%$ erzielt. Die durchschnittliche Fläche der Tiles liegt auch hier bei der relaxierten Dichteanforderung deutlich über der bei strikter Dichteanforderung, dafür sinkt die durchschnittliche Dichte wesentlich stärker. Die Laufzeiten schwanken mit Δ und der Anzahl der generierten Tiles.



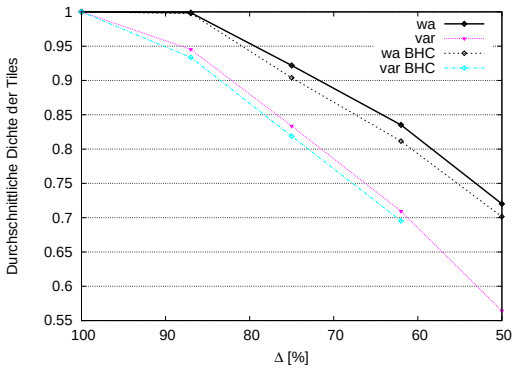
(a) Anzahl der generierten Tiles



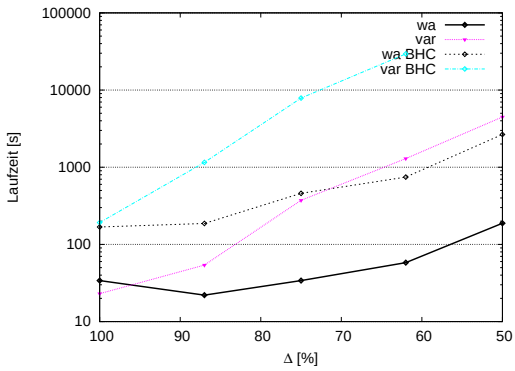
(b) Anzahl der Tiles im minimalen Tiling



(c) Durchschnittliche Fläche der Tiles



(d) Durchschnittliche Dichte der Tiles



(e) Laufzeit

Abbildung 5.11: Evaluation der Heuristiken auf der MovieLens-Datenbank

Kapitel 6

Literaturüberblick

Es wurden Probleme aus unterschiedlichen Anwendungsbereichen untersucht und mit dem Problem des Findens eines Tilings für Collaborative Filtering Daten verglichen.

6.1 Clustering

Clustering-Methoden werden standardmäßig zur Unterstützung von Recommender-Systemen eingesetzt. Dabei kann eine Clusterbildung der Nutzer nach Produkten oder umgekehrt erfolgen. [4] und [7] benutzen Varianten von k-Means, um Nutzer zu gruppieren, wogegen in [14] verschiedene Clustering-Verfahren wie hierarchisches Clustering eingesetzt werden, um Produkte zu gruppieren. [5] benutzt wiederum k-Means um Produkte zu clustern; anschließend werden die Cluster nach den Eigenschaften der enthaltenen Objekte analysiert, um eine Beschreibung der Datenbank zu erhalten. Es gibt aber auch einige Arbeiten, die sich die Beziehungen zwischen Produkt- und Kundenclustern zunutze machen. In [20] wird zum Beispiel zunächst ein Clustering der Kunden und Produkte getrennt vorgenommen und anschließend die Kundencluster anhand der Produktcluster neu berechnet und umgekehrt, bis sich das Clustering stabilisiert hat.

6.2 Formale Begriffsanalyse

Im Rahmen der Formalen Begriffsanalyse, die sich aus der Theorie der Verbände entwickelt hat [21], werden Strukturen betrachtet, die mit Tiles vergleichbar sind. Dort werden sogenannte Gegenstände und Merkmale betrachtet. Ein Begriff ist ein Tupel aus einer Menge von Gegenständen und einer Menge von Merkmalen. Die Menge der Merkmale wird auch als Intension des Begriffs bezeichnet. Die Menge der Gegenstände ist die Extension des Begriffes. Die beiden Mengen bedingen sich gegenseitig. Die Intension umfasst alle Merkmale, die die Gegenstände der Extension gemeinsam haben. Umgekehrt muss die Extension alle Gegenstände beinhalten, die alle Merkmale der Intension aufweisen. Aus der Menge der Begriffe lässt sich ein vollständiger Verband aufbauen. Eine genauere Einführung in die Formale Begriffsanalyse findet sich in [11]. Bei einem Begriff handelt sich also quasi um ein Tile mit 100% Dichte. Es wurden Algorithmen entwickelt [17, 18, 19], die die Menge aller

Begriffe ermitteln. Die Theorie findet viele Anwendungen im Data Mining, zum Beispiel zur Visualisierung von Konzepthierarchien oder zum Entdecken von Assoziationsregeln. Sie lässt sich allerdings nicht vollständig auf die aktuelle Problemstellung übertragen, da eine Lockerung des Konzeptes 'Begriff' nicht möglich ist in dem Sinne, dass die Extension eines Begriffes Gegenstände enthalten könnte, die nicht alle Merkmale der Intension besitzen. Es ist also nicht möglich Tiles mit einer Dichte, die weniger als 100% beträgt, zu bilden.

6.3 Frequent Itemset Mining

Auch beim Erstellen von Assoziationsregeln tritt ein Teilproblem auf, das dem des Tilings ähnelt. Hier liegen ebenfalls 0/1-Datenbanken vor. Eine Assoziationsregel ist von der Form 'Wenn ein Kunde Produkt A und B kauft, so kauft er auch (wahrscheinlich) Produkt C'. Um Assoziationsregeln zu bilden, werden zunächst alle *Frequent Itemsets* gebildet. Bei einem Frequent Itemset handelt es sich um eine Menge von Produkten, die von mindestens k Kunden gekauft wurden, bzw. – in den Begriffen des Frequent Itemset Minings – um eine Menge von Items, die in mindestens k Transaktionen enthalten sind. k nennt man den Support des Itemsets. Assoziationsregeln werden dann aus diesen Frequent Itemsets in einem zweiten Schritt gebildet. Ein solches Itemset lässt sich als Tile interpretieren, das als Produkte alle Items des Itemsets enthält und als Kunden die Transaktionen, die den Support des Itemsets bilden. Zum Finden der Frequent Itemsets stehen sehr effiziente Algorithmen zur Verfügung. Der bekannte Apriori-Algorithmus, auf dessen Prinzip fast alle anderen beruhen, wurde 1993 von Agrawal et al. entwickelt [1, 2]. Er beruht auf der Eigenschaft der Antimonotonie bzw. des 'nach unten Abgeschlossenseins' der Frequent Itemsets: 'Wenn ein Itemset nicht $k-1$-frequent ist, so kann er auch nicht k-frequent sein'. Der Algorithmus führt eine Breitensuche durch und generiert aus allen Itemsets, die k-frequent sind, Itemsets die $k+1$-frequent sind. Er liefert alle Frequent Itemsets für Werte von k zwischen eins und der Anzahl der Items. Andere Algorithmen wie FP-Growth [13] führen eine Tiefensuche durch.

Alle Frequent Itemsets bzw. Tiles zu enumerieren, ist für eine Beschreibung der Datenbank sicher nicht nötig. Manche Algorithmen spezialisieren sich auf das Finden bestimmter Itemsets, anstatt alle zu enumerieren. [3] findet maximale Frequent Itemsets, also solche, für die kein weiteres Itemset existiert, in dem das Itemset komplett enthalten ist und das ebenfalls frequent ist. Hier werden weitaus weniger Itemsets generiert, als es bei den ersten Verfahren der Fall ist. Aus Sicht der Tiles ist dies allerdings noch keine befriedigende Lösung, da der Support eines maximalen Itemsets nicht betrachtet wird. Tiles sollten möglichst groß sein, also möglichst viele Kunden und Produkte umfassen. Deshalb kann es sinnvoll sein, nicht unbedingt ein maximales Itemset zu wählen, sondern aus diesem gegebenenfalls Items herauszunehmen, da dann der Support des Itemsets und damit die Größe des korrespondierenden Tiles wesentlich größer werden kann. Dies wird bei der CHARM-Implementation von Zaki [27] berücksichtigt. Hier werden Closed Frequent Itemsets berechnet. Ein Itemset heißt abgeschlossen, wenn es keine Obermenge des Itemsets gibt, die mindestens genauso hohen Support hat. Das heißt also, dass alle größten Tiles der Datenbank berechnet würden und entspricht den Begriffen aus der Formalen Begriffsanalyse.

Bei den Closed Frequent Itemsets werden durch die Forderung nach Abgeschlossenheit der Itemsets indirekt die Kunden durch den Vergleich des Supports mitberücksichtigt. In [12] werden nicht nur Itemsets, sondern direkt Tiles generiert. Als Maß für ein Tile wird hier auch seine Fläche herangezogen. Der Algorithmus generiert alle Tiles der Datenbank mit 100% Dichte, aber es existiert auch eine Variante, um die n größten Tiles zu ermitteln. Zum Ermitteln der Tiles wird ebenfalls ein Apriori-ähnlicher Ansatz gewählt.

Abgesehen davon, dass nur die Variante des Tilings [12] Kunden und Produkte gleichberechtigt in den Suchprozess einbezieht, ist allen vorgestellten Algorithmen gemeinsam, dass sie nur vollständige Tiles mit einer Dichte von 100% erzeugen. Dies liegt an der Grundannahme des Apriori-Algorithmus, dass eine Menge von Items nicht mehr k-frequent sein kann, sobald eine Untermenge nicht k-frequent ist. Diese gilt nicht mehr, sobald man nicht mehr fordert, dass eine Transaktion alle Items des Itemsets enthalten muss, um zum Support zu zählen. Fordert man beispielsweise, dass eine Transaktion 50% der Items enthalten muss, so kann durch Hinzufügen eines Items zum Itemset der Support wieder ansteigen; und ein Itemset ist eventuell k-frequent, das Teilmengen enthält, die es nicht sind. Der Algorithmus verliert damit seine effiziente Pruning-Strategie und wird in dieser Form unbrauchbar.

Unter dem Aspekt des fehlertoleranten Frequent Itemset Minings [15, 26] wurden erste Ansätze entwickelt, um dieses Problem zu umgehen. Yang et al. führen in [26] den Begriff des starken und schwachen fehlertoleranten Itemsets ein. Für einen gegebenen Schwellenwert ist ein fehlertolerantes Frequent Itemset stark, wenn die Dichte jeder Transaktion im Support über dem Schwellenwert liegt, und es ist schwach, wenn die Dichte des von Itemset und Supporttransaktionen aufgespannten Tiles über dem Schwellenwert liegt. Hier lassen sich einige schwache Pruning-Strategien anwenden. Mit diesen Definitionen allein lassen sich 'Trittbrettfahrer' allerdings nicht vermeiden, wie leicht ersichtlich ist. Pei et al. [15] wählen einen sehr ähnlichen Ansatz, indem sie nicht einen Prozentsatz, sondern eine feste Anzahl an Items vorgeben, die in einer Transaktion fehlen dürfen, damit sie noch zum Support gezählt wird. Um einen Extremfall der Trittbrettfahrer wie in Abbildung 2.2 zu vermeiden, wird gefordert, dass jedes Item für sich einen gewissen Support, also eine Mindestdichte, hat. Diese Dichte kann frei gewählt werden. Durch die Annahme einer festen Anzahl fehlender Items in einer Transaktion ergibt sich wieder ein antimonotones Konzept und die Pruning-Strategien greifen erneut. Allerdings findet hier keine Skalierung der Anzahl fehlender Items im Vergleich zur Größe des Itemsets statt, sodass mit steigender Größe des aufgespannten Tiles auch seine Dichte ansteigen muss.

Ein dritter Ansatz in diesem Bereich wird von Seppänen und Mannila mit 'Dense Itemsets' [16] unternommen. Auch hier soll es möglich sein, dass Transaktionen zum Support eines Itemsets zählen, wenn sie nicht alle Items enthalten. Es wird eine Schranke δ in Prozent und ein Support σ absolut vorgegeben. Ein Itemset hat schwache Dichte δ, wenn es eine Menge von σ Transaktionen gibt, sodass diese zusammen mindestens δ% Items des Itemsets enthalten, dies entspricht also der Dichte eines Tiles, wie es in dieser Arbeit definiert wurde. Um Trittbrettfahrer, ein Ausdruck, der der Arbeit von Seppänen entnommen wurde, zu vermeiden, wird zusätzlich der Begriff der starken Dichte eines Itemsets eingeführt. Ein Itemset hat eine starke Dichte δ, wenn jede Teilmenge des Itemsets mindestens eine schwache Dichte von δ hat. Hier liegt also jetzt eine variable Schranke der zulässigen, fehlenden Items vor und gleichzeitig werden Trittbrettfahrer vermieden. Damit ist bezüglich der Itemsets bzw. der Tiles ein sehr ähnlicher Rahmen aufgespannt worden.

Im Unterschied zur vorliegenden Arbeit werden bei 'Dense Itemsets' bei den Dichtebetrachtungen jedoch nur Items herangezogen. Transaktionen spielen nur für den Support eine Rolle; ihre Dichte wird nicht überprüft. Dementsprechend wird auch nicht im Sinne der Überdeckung der Datenbank überprüft, ob jede Transaktion im Support eines Itemsets enthalten und damit überdeckt ist. Der Algorithmus enumeriert nach Vorgabe von δ und σ alle Itemsets, die starke Dichte δ bei Support σ besitzen. Im hier vorgestellten Ansatz wird auf eine Enumeration aller Tiles verzichtet und lediglich so viele Tiles erzeugt, wie zur Überdeckung der Datenbank notwendig sind.

Kapitel 7

Zusammenfassung und Ausblick

Es wurde ein Algorithmus entwickelt, der Techniken des Regellernens verwendet, um Tilings in Collaborative Filtering Daten zu finden. Wie auf Tests mit künstlichen Datenbanken deutlich wurde, ist er in der Lage, vorhandene Strukturen in den Daten zu rekonstruieren. Die dabei entstehenden Tiles erreichen mit sinkender geforderter Dichte eine immer größere Fläche, wobei die tatsächliche Dichte des Tiles stets über dem geforderten Schwellenwert liegt. Um Trittbrettfahrer zu vermeiden, wurde eine Dichteanforderung an Kunden- und Produktprofile gestellt, die sehr strikt ist. Dies verhindert oft die Entstehung potentiell größerer Tiles. Allerdings wird so die Güte hinsichtlich der Ähnlichkeit und Zusammengehörigkeit der enthaltenen Kunden und Produkte gewährleistet. Eine leichte Relaxierung dieser Dichteanforderung erbringt schließlich noch wesentlich größere Tiles, deren Dichte kurz über dem Schwellenwert liegt. Auch hier wird durch Dichteanforderung und Minimierung der Varianz eine optimale Verteilung positiver und negativer Kunden-/Produkt-Paare in den Tiles erbracht.

Vergleicht man auf künstlichen Datenbanken die Größe der gefundenen Tiles mit der Größe der original in die Datenbank eingestreuten Tiles, so sind die gefundenen Tiles für große geforderte Dichten zunächst deutlich kleiner. Allerdings kann trotzdem ein sehr großer Anteil der Tiles wiedergefunden werden. Durch Variation des Schwellenwertes der Dichte kann ein Rauschen in den Datenbanken kompensiert werden.

Ob die strikte oder relaxierte Dichteanforderung zum Einsatz kommt, sollte anhand der Laufzeit, die investiert werden kann, entschieden werden. Bei relaxierter Dichteanforderung ist diese deutlich größer. Insgesamt sind die Laufzeiten für die betrachteten Instanzen bei strikter Dichteanforderung zufriedenstellend und auch bei deren Relaxierung noch akzeptabel, wenn bedacht wird, dass effiziente Algorithmen, die bisher für ähnliche Fragestellungen verwendet wurden, sich nicht auf dieses Problem übertragen lassen [16]. Die Verwendung der Suche Beam-Search mit Hill-Climbing lässt die Laufzeit noch einmal um einen konstanten Faktor ansteigen. Es wird eine kleine Verbesserung beispielsweise hinsichtlich der Fläche der Tiles erzielt. Hier muss abgewogen werden, ob eine Verbesserung auf Kosten der Laufzeit erstrebenswert ist.

Es lassen sich zwei Kernpunkte zur Verbesserung des Algorithmus ausmachen. Erstens sollte aus Sicht des Tilings vor allem eine Version des Separate-&-Conquer-Ansatzes zum Covering der Datenbank entwickelt werden, die von Beginn an eine bessere Verteilung der

Tiles anstrebt. Oft weisen Tiles sehr starke Überlappungen auf, und für das Tiling wird kaum zusätzliche Information gewonnen. Durch geschicktere Auswahl der Startpunkte lässt sich dieser Effekt eventuell vermeiden. Dazu könnten bereits bei der Auswahl eines Startpunktes dessen Profile betrachtet werden. Da dies jedoch zusätzliche Laufzeit kosten wird, muss evaluiert werden, ob diese durch die erwartete geringere Anzahl an generierten Tiles kompensiert werden kann. Alternativ lässt sich auch das minimale Tiling verwenden. Erstrebenswert wäre jedoch ein Tiling, dessen Anzahl an Tiles sich nur wenig von dem des minimalen Tiling unterscheidet.

Anhand der Anzahl der generierten Tiles lässt sich auch diskutieren, ob wirklich versucht werden sollte, die gesamte Datenbank abzudecken. Für vereinzelte Punkte wird jeweils ein neues Tile generiert, wie durch Einfügen von Rauschen in künstlichen Datenbanken festgestellt wurde. Da eigentlich Häufungspunkte in den Daten gesucht werden, stellt sich die Frage, ob dies nicht zu einer unnötigen Überanpassung des Tilings an die Daten führt; eine Problematik, wie sie so auch im Regellernen auftritt. Beim verwendeten Round-Robin-Verfahren zur Auswahl der Startpunkte könnte zum Beispiel ein Profil als vollständig abgearbeitet gelten, wenn 95% der enthaltenen Produkt- bzw. Kundennummern in Kombination mit der Profilnummer abgedeckt werden.

Zweitens muss aus Sicht der Tiles eine Heuristik und Suche entwickelt werden, die Kandidaten differenzierter beurteilen kann, d.h. die weniger Kandidaten mit gleichem Wert belegt bzw. Kandidaten mit gleichem Heuristikwert besser verarbeiten kann. Ein Versuch wurde bereits mit der Einführung von Starttiles unternommen, der sich leider nicht als erfolgreich erwiesen hat, da die Generalisierungsmöglichkeiten der Tiles zu sehr eingeschränkt wurden. Eventuell wäre auch die Untersuchung eines Top-Down-Ansatz sinnvoll, der ausgehend von der gesamten Datenbank diese auf einzelne Tiles reduziert. Hier lassen sich vielleicht durch die umgedrehte Sichtweise auf das Problem weitere wertvolle Einsichten gewinnen.

Um die Laufzeit des Algorithmus zu reduzieren, könnte der Entwurf von Heuristiken zum Pruning von Kandidatenlisten sinnvoll sein, um deren Anzahl bzw. die Anzahl der Kunden und Produkte zu verringern, die überprüft werden, ob sie als Kandidat in Frage kommen. Bisher werden die Kunden oder Produkte neu überprüft, die im Profil des zuletzt eingefügten Kandidaten enthalten sind. Hier könnte beispielsweise überlegt werden, ob ein Kunde, der schon oft überprüft und nie in die Kandidatenliste aufgenommen wurde, überhaupt noch einmal betrachtet werden sollte.

Ebenfalls könnte eine Modifizierung der Beam-Search erfolgen, sodass auch Tiles untersucht werden, die zu einem bestimmten Zeitpunkt scheinbar schlechtere Alternativen darstellen. Dazu müsste ein Gütemaß für Tiles entwickelt werden, das sowohl Dichte als auch Fläche eines Tiles sowie potentielle weitere Generalisierungsmöglichkeiten in Betracht zieht. Eine solche Suche könnte bei entsprechender Rechenleistung wahrscheinlich wesentlich größere Tiles auch bei den gegebenen Dichteanforderungen finden. Hier ist einiges Potential zu erwarten.

Bisher wurden hauptsächlich künstliche Datenbanken untersucht, die gleich viele Kunden und Produkte enthielten. Dies ist bei der MovieLens-Datenbank nicht der Fall. Eine Auswirkung dieser unterschiedlichen Anzahl von Kunden und Produkten muss noch näher untersucht werden.

Durch die erzeugten Tilings werden leicht verständliche Beschreibungen der Datenbanken gewonnen. Um letztendlich über die Tauglichkeit der Tilings zur Generierung von Empfehlungen zu schließen, sollte ein Recommender-System aufgesetzt werden, das wie bei Verwendung von Clustering-Modellen neue Nutzer und Produkte anhand ihrer Profile Tiles zuordnet und auf dieser Zuordnung basierend Empfehlungen generiert.

Danksagung

Ich danke der GroupLens Research Group für ihre Veröffentlichung der MovieLens Daten.

Literaturverzeichnis

[1] R. Agrawal, T. Imielinski und A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, S. 207-216. Washington, D.C.. 1993.

[2] R. Agrawal und R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th International Conference on Very Large Databases*. Santiago. 1994.

[3] R. J. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proc. ACM SIGMOD '98 International Conference on Management of Data*, S. 85-93. Seattle. 1998.

[4] S.H.S. Chee, J.Han und K. Wang. RecTree: An Efficient Collaborative Filtering Method. In *Lecture Notes in Computer Science, 2114*, S. 141-151. 2001.

[5] P. Clerkin, P. Cunningham und C. Hayes. Concept Discovery in Collaborative Recommender Systems. *Technical Report*. Dublin. 2003.

[6] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. *Tech report HPL-2003-4*. HP Laboratories, Palo Alto, USA. 2003.

[7] D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas und R. Vuduc. Swami: A Framework for Collaborative Filtering Algorithm Development and Evaluation. In *Proc. 23rd Conference on Research and Development in Information Retrieval*. Athen. 2000.

[8] P.A. Flach. The Geometry of ROC Space: Understanding Machine Learning Metrics Through ROC Isometrics. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, S. 194-201. AAAI Press. 2003.

[9] J. Fürnkranz und P. A. Flach. ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms. In *Machine Learning 58(1)*, S. 39-77. 2005.

[10] J. Fürnkranz und P. A. Flach. An Analysis of Rule Evaluation Metrics. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, S. 202-209. AAAI Press. 2003.

[11] B. Ganter und R. Wille. Formal Concept Analysis: Mathmatical Foundations. Springer. 1999.

[12] F. Geerts, B. Goethals und T. Mielikäinen. Tiling Databases. In *The 7th International Conference on Discovery Science (DS'04)*, S. 278-289. Springer. 2004.

[13] J. Han, J. Pei und Y. Yin. Mining Frequent Patterns Without Candidate Generation. *ACM SIGMOD International Conference on Management of Data*, S. 1-12. Dallas. 2000.

[14] M. O'Connor und J. Herlocker. Clustering Items for Collaborative Filtering. In *ACM SIGIR 99 Workshop on Recommender Systems: Algorithms and Evaluation*. 1999.

[15] J. Pei, A. K. Tung und J. Han. Fault-Tolerant Frequent Pattern Mining: Problems and Challanges. In *Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD-2001)*. Santa Barabara. 2001.

[16] J. Seppänen und H. Mannila. Dense Itemsets. In *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, S. 683-688. ACM Press. 2004.

[17] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier und L. Lakhal. Fast Computation of Concept Lattices Using Data Mining Techniques. In *Proc. 7th International Workshop on Knowlegdge Representation Meets Databases*. Berlin. 2000.

[18] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier und L. Lakhal. Computing Iceberg Concept Lattices with Titanic. In *Journal on Knowledge and Data Engineering 42(2)*. S. 189-222. 2002.

[19] G. Stumme. Efficient Data Mining Based on Formal Concept Analysis. In *Proc. 13th International Conference on Database and Expert Systems Applications*. S. 534-546. 2002.

[20] L.H. Ungar und D.P. Foster. Clustering Methods for Collaborative Filtering. In *Workshop on Recommendation Systems at the Fifteenth National Conference on Artificial Intelligence*. 1998.

[21] R. Wille. Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts. In *Ordered Sets*. Reidel. 1982.

[22] http://www.amazon.de/exec/obidos/tg/browse/-/911764/ref=cs_nav_bn_2F_2/028-6818894-5585317, Stand 10.05.2005.

[23] http://www.cio.com/archive/080100/bezos.html, Stand 10.05.2005.

[24] http://www.groupLens.org, Stand 21.05.2005.

[25] http://www.movieLens.org, Stand 21.05.2005.

[26] C. Yang, U. Fayyad und P.S. Bradley. Efficient Discovery of Error-Tolerant Frequent Itemsets in High Dimensions. In *Proc. KDD-2001*, S. 194-203. ACM Press. 2001.

[27] M.J. Zaki und C.-J. Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining. In *2nd SIAM International Conference on Data Mining*. Arlington. 2002.