Häufige Mustererkennung in sequenziellen Phasendatenbanken

Data-Mining und Prognoseerstellung am Beispiel von Bausparkollektiven

Inaugural-Dissertation

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Herrn Dipl-Math. Marcel Schwalb

aus Euskirchen

Köln, 2019

Berichterstatter:	Prof. Dr. Rainer Schrader
(Gutachter)	Prof. Dr. Oliver Schaudt
Tag der mündlichen Prüfung:	15.10.2018

Abstract

This thesis deals with pattern recognition in the german "bauspar" system which is a highly popular system of collective savings and crediting in Germany. It uses frequent subsequence mining and association rule mining to design and improve algorithms that predict the behavior of "bauspar" customers.

Sequential pattern mining is a classical data mining topic with a lot of practical applications. Given a database of finite sequences of products or events, frequent sequence mining tries to find all subsequences in a database that fulfill a certain support threshold. Those frequent subsequences can be used to identify crucial information about the structure of the database. They can also be used to construct association rules like "customers who buy product X will also buy product Y".

This thesis extends the standard sequential pattern mining problem to deal with phase constraints that are typically found in "bauspar" systems. Instead of treating all products independently every product belongs to a phase. Products between phases have to occur in a specific order, although an arbitrary number of phases can be skipped. The phase model is extended to include multidimensional sequence data and multidimensional data that is independent of time. Advantages and disadvantages of the model are discussed.

The present work includes 5 new algorithms to search for frequent subsequences in databases with phase constraints and 5 new algorithms to find frequent subsequences in the more complex model with multidimensional data. Test results show that the new algorithms can be an order of magnitude faster than standard algorithms if the additional phase information is used.

For the first time results of sequential pattern mining with phase constraints and a fully developed customer model are used to explore "bauspar" collectives in Germany. It can be shown that the additional information of multidimensional customer data can significantly improve the prediction quality of customer behavior in "bauspar" collectives. Practical backtesting results verify that the proposed approach via frequent association rules mining and decision trees can help to form a better understanding of the inner structure of costumer behavior.

In the future the results of this thesis may inspire more research in the field of "bauspar" pattern mining and may lead to new models of collective savings and crediting prediction.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich ausgiebig mit der häufigen Musterkennung in Bausparkollektiven und darauf aufbauend mit dem Entwurf und der Verbesserung von Prognoseverfahren zur Vorhersage des Verhaltens von Bausparern.

Die häufige Mustererkennung in Sequenzen ist ein klassisches Data-Mining-Gebiet. Ausgehend von einer Datenbank an endlichen, linear geordneten Sequenzen von Produkten oder Ereignissen werden alle häufigen Teilsequenzen dieser Menge gesucht, die eine Mindestanzahl von Supersequenzen in der Menge besitzen. Anhand dieser häufigen Teilsequenzen können grundlegende Charakteristika der Datenbank ermittelt werden. Weiterhin lassen sich Assoziationsregeln der Form: "Kunden die Verhalten X gezeigt haben, zeigen später auch Verhalten Y" formulieren, die mit Wahrscheinlichkeiten versehen werden können.

Die Dissertation erweitert das bestehende Standardmodell der Suche nach häufigen Teilsequenzen in Sequenzen um ein Phasenmodell, in dem Sequenzelemente in Phasen eingeteilt werden können. Diese Phasen müssen von allen Sequenzen in einer festen Reihenfolge durchlaufen werden, wenn auch beliebige Phasen übersprungen werden können. Zusätzlich wird das neue Phasenmodell um den Umgang mit multidimensionalen, zeitunabhängigen Attributen und multistufigen Sequenzen erweitert. Hierbei werden Eigenschaften der Modelle durchleuchtet und effiziente Unterteilungsstrategien diskutiert.

Es werden insgesamt 5 neue Algorithmen für die Suche nach häufigen Teilsequenzen in sequenziellen Phasendatenbanken und 5 neue Algorithmen für die Suche in komplexeren Modellen vorgestellt. In dieser Arbeit inkludierte Testergebnisse zeigen, dass einige dieser Algorithmen die bisherigen Standardalgorithmen im Falle von Phasendatenbanken um ein Vielfaches in der Geschwindigkeit übertreffen können.

Erstmalig werden in dieser Arbeit die theoretischen Ergebnisse auf die Daten von Bausparkollektiven angewandt. Hierzu wird ebenso erstmalig ein vollumfänglicher Kundenansatz statt des bisher üblichen Vertragsansatzes zur Modellierung von Bausparkollektiven genutzt. In dieser Arbeit konnte gezeigt werden, dass das Vorliegen von zusätzlichen Daten auf Kundenebene signifikanten Einfluss auf die Verhaltensweisen und Entscheidungen der Bausparer offenlegt.

Die Ergebnisse dieser Arbeit können zukünftig genutzt werden, um ein tieferes Verständnis für die Struktur und Dynamik in Bausparkollektiven zu gewinnen. Aufbauend auf den vorgestellten Ansätzen können neue Formen der Kollektivprognose und der Vorhersage von Individualverhalten konstruiert werden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Big Data

Der Ausdruck “Big Data” ist seit ein paar Jahren sowohl in der Wissenschaft als auch in Wirtschaft und Medien in vieler Munde. Oftmals stecken sehr unterschiedliche, konkrete Vorstellungen hinter der Verwendung des Begriffs, die zu zahlreichen Versuchen führten, eine standardisierte Definition von Big Data zu finden [WB13] [NWG]. Innerhalb dieser Arbeit wird sich an der Definition der NIST Working Group orientiert.

> Big Data consists of extensive datasets - primarily in the characteristics of volume, variety, velocity, and/or variability - that require a scalable architecture for efficient storage, manipulation, and analysis. [Gro]

Der Fokus wird hierbei auf zwei Schwerpunkte gelegt: auf die Komplexität der Daten und den effizienten Umgang mit ihnen.

1.2 Data-Mining

Im Laufe der letzten Jahrzehnte erfuhr das Forschungsgebiet des Data-Minings immer größere Bedeutung. Mit dem Aufkommen von elektronischen Verarbeitungssystemen und der massenhaften Speicherung von digitalen Daten stellte sich schnell die Frage, wie mit diesen Informationen effektiv und effizient umgegangen werden sollte. Viele traditionelle statistische Verfahren waren mit der Quantität und Komplexität der Eingabedaten überfordert. Wenn man nicht a priori wusste, nach welchem Aspekt man suchte, stand man bei vielen Datenhaushalten vor einer scheinbar endlosen Datengrube.

Data-Mining versucht die Probleme mit elektronischen Datenhaushalten zu bündeln und standardisierte Lösungsverfahren zu entwickeln. Da jedoch die Struktur der Daten und das Wesen der Beziehungen unter ihnen in der Praxis stark variieren, bleibt eine detaillierte Analyse der einzelnen Anwendungsfelder nicht aus. Charu C. Aggarwal, einer der Pioniere der Data-Mining Forschung, schreibt hierzu:

> While there is a conceptual portability of algorithms between many data types at a very high level, this is not the case from a practical perspective. The reality is that the precise data type may affect the behavior of a particular algorithm significantly. As a result, one may need to design refined variations of the basic approach for multidimensional data, so that it can be used effectively for a different data type. [Agg15]

Das deutsche Bausparsystem stellt folgerichtig standardisierte Data-Mining-Verfahren vor eine große Herausforderung. Durch die weltweit einzigartige Struktur des Bausparens in der Finanzbranche als ein Phasenmodell mit gekoppelten Spar- und Darlehensabschnitten und hohen Freiheitsgraden bei der Erfüllung der Tarifauflagen, ist es einerseits zwar stark strukturiert, aber andererseits auch sehr freizügig in den Optionen des Kunden in jeder Phase. Klassische Data-Mining-Verfahren geraten hier schnell an ihre Grenze und die Entwicklung von spezialisierten Herangehensweisen wird notwendig.

1.3 Prognosen des Kundenverhaltens in Bausparkollektiven

Modelle und Software zur Prognose von Bausparkollektiven sind in der Branche weit verbreitet. Tatsächlich verpflichtet das vor kurzer Zeit überarbeitete Bausparkassengesetz und die Bausparkassenverordnung Bausparkassen, regelmäßige Prognoseergebnisse an die Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) zu schicken.
Die üblichen Prognosemodelle in Bausparkassen basieren auf einer Fortschreibung von Bausparverträgen, sei es in Form von Schichten (NBI), extrapolierten Stichproben (BHW) oder Einzelvertragssimulationen (KOSIMO). Sie gehen in der Grundannahme davon aus, dass alle Verträge unabhängig voneinander Optionen ausüben und höchstens von einem Marktzins und der Bausparkasse beeinflusst werden. In der Realität steckt hinter jedem Bausparvertrag jedoch ein Mensch, eine Personengemeinschaft oder eine juristische Person. Der Eigentümer des Vertrages hat mitunter noch mehrere weitere Verträge, die er parallel bedient. Läuft ein Bausparvertrag aus, so kann er gewillt sein, als Ersatz einen neuen Bausparvertrag abzuschließen. Ebenso kann das Erreichen der Darlehensphase in einem Vertrag bedeuten, dass er die Besparung des nächsten Vertrages startet. Aus Sicht der Kasse ist die Prognose des Verhaltens des Kunden ein großer Mehrwert, da die Ergebnisse nicht nur zur Kollektivüberwachung sondern ebenso für Scoring-Verfahren oder Segmentierungsanalysen genutzt werden können.
Institutsübergreifende Standardlösungen für die Prognose von Bausparkollektiven auf Kundenbasis sind dem Autor nicht bekannt.

1.4 Ergebnisse der Dissertation

Die vorliegende Arbeit befasst sich mit dem Thema Data-Mining und Prognosemodellierung im Bereich des Bausparens. Hierbei konnten folgende neuwertigen Erkenntnisse gewonnen werden:

1. Die Formulierung und ausführliche Analyse zweier neuer mathematischer Modelle zur Suche nach häufigen Mustern in Phasendatenbanken: Es wird sowohl ein um Phasen angereichertes Sequenzmodell betrachtet als auch ein weit komplexeres Modell mit Unterstützung für multidimensionale Datenstrukturen und mehrstufige Sequenzsuchen aufgestellt. Hierbei werden Eigenschaften der Modelle durchleuchtet und effiziente Unterteilungsstrategien diskutiert.

2. Die Konstruktion von neuen Algorithmen zur Erfassung aller häufigen Sequenzen in Phasendatenbanken: Es wurden insgesamt 5 neue Algorithmen für die Mustererkennung in Sequenzdatenbanken und 5 neue Algorithmen für die Suche im komplexeren SMMP-Modell vorgestellt. In dieser Arbeit inkludierte Testergebnisse zeigen, dass einige dieser Algorithmen die bisherigen Standardalgorithmen im Falle von Phasendatenbanken um ein Vielfaches in der Geschwindigkeit übertreffen.

3. Ein Erkenntnisgewinn über die Struktur von Bausparkollektiven und dem Verhalten von Bausparern aus Sicht einer Kundenebene: In dieser Arbeit konnte gezeigt werden, dass das Vorliegen von zusätzlichen Daten auf Kundenebene signifikanten Einfluss auf die Verhaltensweisen und Entscheidungen der Bausparer offenlegt.

4. Vorschläge für die Verbesserung der Prognosequalität von Kollektivsimulationen: Es werden Ansätze diskutiert, Prognoseverfahren aufgestellt und Testergebnisse ermittelt, die nahelegen, dass der Einbezug von Kundenmerkmalen die Genauigkeit von Kollektivsimulationsrechnungen verbessern kann.

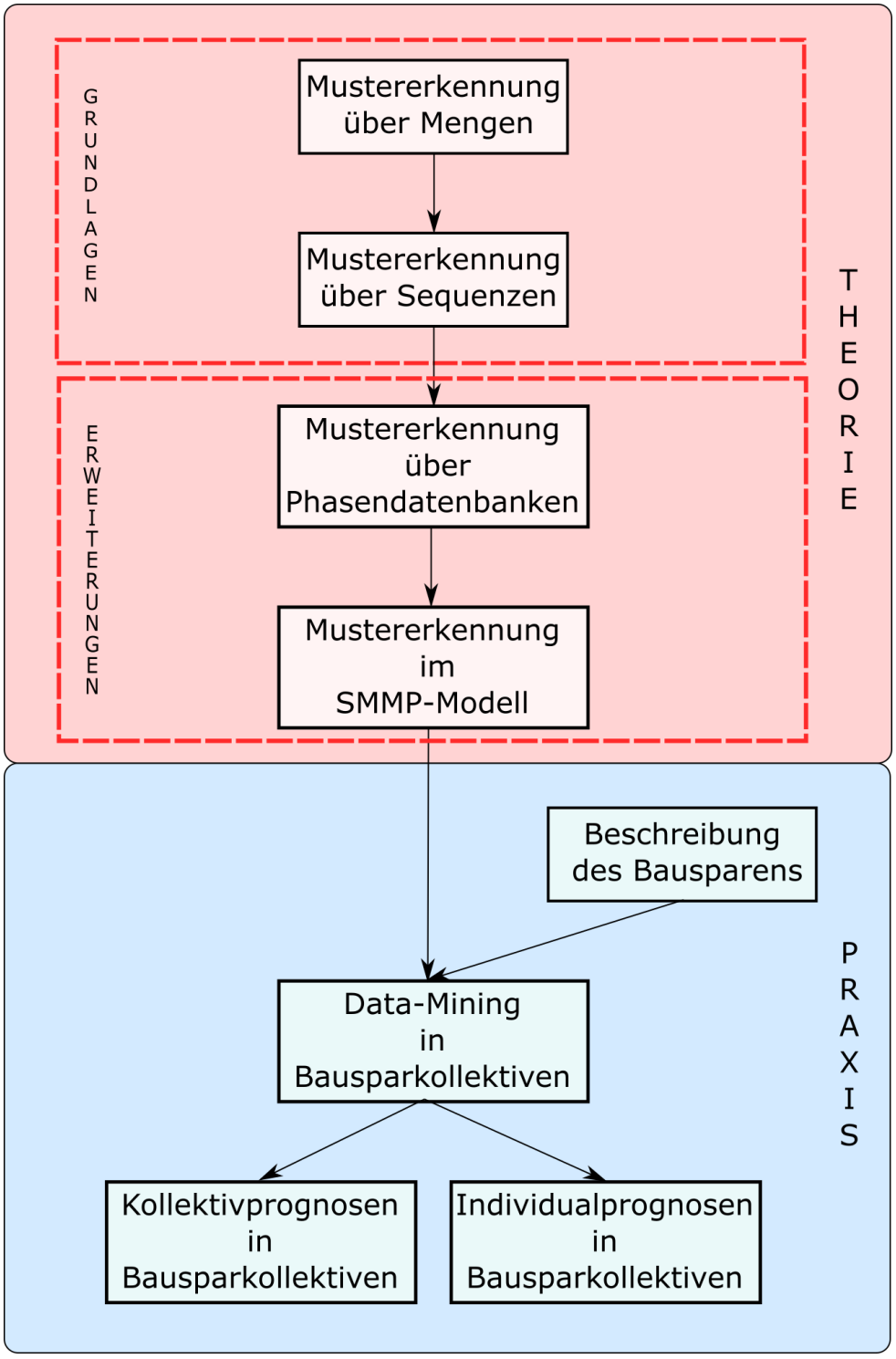
5. Die Konstruktion eines Standardprozesses zur Nutzung von Kundenattributen im Bausparwesen zur Aufstellung von Individualprognosen auf Grundlage von häufiger Mustererkennung in sequenziellen Phasendatenbanken. Die Tragfähigkeit des Modells konnte anhand von Backtests für ausgewählte Kundenverhalten bestätigt werden.

1.5 Aufbau der vorliegenden Arbeit

Die Arbeit lässt sich grob in zwei Bereiche unterteilen. In den Kapiteln 3 bis 6 wird das Gebiet der Suche nach häufigen Mustern aus mathematischer und informatischer Sicht in der Theorie beleuchtet. In den Kapiteln 3 und 4 werden hierzu die Grundlagen innerhalb der Literatur zusammengefasst, während in den Kapiteln 5 und 6 zwei neue Modelle zur Mustererkennung in

Phasendatenbanken vorgestellt werden. Neben der Herleitung und Beweise theoretischer Charakteristika werden ebenso neue Algorithmen zur Behandlung der Problemstellung konstruiert und in theoretischen Experimenten mit Standardalgorithmen aus der Literatur verglichen.

Der zweite Teil der Arbeit umfasst die Kapitel 7 bis 11 und widmet sich der praktischen Anwendung der im ersten Teil hergeleiteten Verfahren auf die spezielle Problemstellung des Data-Mining in Bausparkollektiven. Die neugewonnenen Erkenntnisse werden folgend genutzt, um die Prognosequalität von Kollektivsimulationen und Individualprognosen zu verbessern. Am Ende wird ein Ausblick über mögliche Erweiterungen des Ansatzes gegeben. Schematisch lässt sich der Inhalt der Arbeit in folgender Grafik ablesen.



Abstrakter Aufbau des Dissertationsinhalts

Teil I

Theorie

Kapitel 2

Einleitung Theorie

Der relativ neue Begriff Big Data bezeichnet die Existenz von Datensätzen, die im Allgemeinen zu groß oder zu komplex sind, um sie manuell oder mit einfachen statistischen Analysen auszuwerten. In der Regel wird deswegen auf Verfahren des Data-Minings und insbesondere der automatisierten Mustererkennung zurückgegriffen. Diese Verfahren versuchen ohne manuelle Vorgabe zu vieler Parameter, wiederkehrende Muster und auffällige Beziehungen in den Daten zu finden und diese auszugeben, wenn sie eine Signifikantsgrenze überschreiten.

Big Data stellt algorithmische Verfahren vor eine große Herausforderung. Aus theoretischer Sicht müssen sie mit komplexen Eingaben umgehen können und dabei möglichst exakte und vollständige Ergebnisse garantieren. Aus praktischer Sicht ergeben sich aufgrund der Masse und Komplexität an Daten sehr schnell Laufzeitprobleme. Die Menge der Eingabedaten kann es unmöglich machen, alle notwendigen Daten zur gleichen Zeit im Arbeitsspeicher zur Verfügung zu stellen. Hier können Divide-And-Conquer-Verfahren Abhilfe bieten.

Im Rahmen dieser Arbeit werden ausschließlich Forschungsfelder des Data-Mining betrachtet, in denen die vollständige Schürfung von häufigen Strukturen behandelt wird. Dies beinhaltet häufige Produktmengen, Sequenzen und Assoziationsregeln.

Ein Ziel dieser Arbeit ist die Nutzung von Ansätzen des Data-Minings, die auch für Anwender noch verständlich sind, die zwar fachliche Kenntnis des Problemumfelds mitbringen, aber keine ausgebildeten Mathematiker oder Informatiker sind. Dies beinhaltet Verfahren, die zu jeder Zeit transparente Zwischenergebnisse liefern und ohne aufwändige Datentransformationen ausgewertet werden können. Der Autor der Arbeit ist der Überzeugung, dass Data-Mining in konservativen Umfeldern wie der Bausparkassenbranche nur gelebt werden kann, wenn die verwendeten Algorithmen nicht wie eine Black-Box funktionieren und von den Anwendern akzeptiert werden. Weiterhin ist gerade im deutschen Umfeld eine transparente Kontrollmöglichkeit durch hausinterne Revisionen, Wirtschaftsprüfer und staatliche Aufsichten notwendig.

Diese Qualitätssicherungsmaßnahmen stoßen auf weniger Widerstand, wenn die verwendeten Algorithmen durchschaubar erklärt werden können.

Kapitel 3

Mustererkennung über Mengen

3.1 Einleitung

Die Mustererkennung in Mengensystemen gehört zu den ältesten Gebieten der Data-Mining-Forschung. Durch die minimalistische Modellierung einer Datenbank als System von Mengen können zahlreiche Schwierigkeiten komplexerer Modellierungen vermieden werden.

Der klassische Anwendungsfall betrachtet Einkäufe in einem Supermarkt. Als Grundmenge wird die Menge aller im Laden erhältlichen Produkte definiert. Ein Einkauf wird als Teilmenge dieser Produktmenge interpretiert. Ziel der Mustererkennung ist die Bestimmung von Produkten, die besonders häufig zusammengekauft werden.

Durch die Modellierung aller möglichen Objekte als Elemente einer Menge erhält man ein sehr flexibles Rahmengerüst, in dem grundsätzliche Beziehungen der Objekte untereinander durchleuchtet werden können. Man erkauft sich diesen Vorteil jedoch durch mehrere Nachteile. Zum einen werden keine bereits bekannten und teilweise zwingenden komplexeren Beziehungen der Objekte untereinander erfasst: die Ermittlung von quantitativen Mengen oder zeitlichen Abhängigkeiten wäre nur durch explizite Codierung neuer Elemente in die Grundmenge möglich. Zum anderen führen die fehlenden Einschränkungen an die möglichen Kombinationen von Objekten dazu, dass der ohnehin schon exponentiell wachsende Raum der möglichen Lösungen nicht effizient kontrolliert werden kann.

Es ist bekannt, dass viele der betrachteten Zählprobleme #P-vollständig, bzw. ihre entsprechenden Aufzählungsprobleme NP-schwer sind [Yan04]. Obgleich dieser Einschränkungen fokussiert sich ein großer Teil der Forschung auf die Entwicklung exakter Verfahren mit möglichst geringer Laufzeit. Hierzu werden neben einer optimierten Implementierung Teilmengen der Lösungsmenge gesucht, aus denen sich die Gesamtlösungsmenge (in exponentieller Laufzeit) generieren lässt.

Über die Jahre haben sich zahlreiche Forschende mit dem Thema beschäftigt, wodurch eine große Anzahl an Algorithmen erschaffen wurde, die sich aus verschiedenen Blickwinkeln und mit unterschiedlichen Methoden der Problemstellung nähern. Übersichten und Effizienzvergleiche zwischen den bekanntesten Algorithmen können unter anderem in [AAK17] [CCA14] [Agg15] [NAS+14] gefunden werden.

3.2 Häufige Produktmengen

3.2.1 Definitionen und Problemstellung

Gegeben sei eine endliche Menge $P = \{p_1, p_2, \ldots, p_n\}$ an Produkten.

Definition 3.1. *Eine **Produktmenge** $T \subseteq P$ ist eine nichtgeordnete Menge von Produkten.*

Wir schreiben hierbei formal $T = \{p_1, p_2, \ldots, p_k\}$. In erklärenden Texten und Beispielen wird aus Übersichtsgründen eine vereinfachte Schreibform angewandt. Seien $A, B, C \in P$, so ist die Schreibweise ABC äquivalent zu $\{A, B, C\}$.

Definition 3.2. *Die Menge aller zur Verfügung stehenden Produktmengen $D = \{T_1, \ldots, T_m\}$ in einer Datenbank heißt **Transaktionsdatenbank**. Produktmengen in einer Transaktionsdatenbank werden zur einfachen Unterscheidbarkeit auch **Transaktionsmengen** bezeichnet.*

Im Rahmen dieser Arbeit ist die Transaktionsdatenbank vorgegeben, endlich und nicht veränderbar. Insbesondere sind alle Transaktionen zum Zeitpunkt der Analyse vollumfänglich bekannt und werden nicht erst zur Laufzeit der Verfahren zugespielt (vergleiche Onlineverfahren).

Definition 3.3. *Sei $I \subseteq P$ eine beliebige Menge an Produkten. Die **Überdeckung** von I in D ist die Menge*

$$U_I = \{T \in D : I \subseteq T\}$$

aller Transaktionsmengen, die I enthalten.

Zu beachten ist, dass nicht notwendigerweise $I \in D$ gelten muss.

Definition 3.4. *Der **absolute Support** $abssupp_D(I)$ einer Produktmenge I in D ist die Mächtigkeit der Überdeckung*

$$abssupp_D(I) = |U_I|.$$

*Der **relative Support** einer Produktmenge I in D ist*

$$supp_D(I) = \frac{abssupp_D(I)}{|D|} = \frac{|U_I|}{|D|}.$$

Einleitende Definitionen ermöglichen es uns nun, die Aufgabenstellung der Erkennung von häufigen Produktmengen nach [AS95] zu formulieren.

Problemstellung 3.5. *Gegeben sei eine Produktmenge $P = \{p_1, p_2, \ldots, p_n\}$, eine Transaktionsdatenbank $D = \{T_1, T_2, \ldots, T_m\}$ mit $T_i \subseteq P$ und ein Schwellwert $0 \leq minsupp \leq 1$. Finde die Menge*

$$I = \left\{ I_j \subseteq P : supp_D(I_j) \geq minsupp \right\}.$$

*Mengen in I werden **häufige Produktmengen** genannt.*

Beispiel 3.6. *Gegeben sei die Menge $D = \{ABC, AB, C\}$ und $minsupp = 0,5$. Die Ergebnismenge der häufigen Produktmengen I besteht aus $I = \{\emptyset, A, B, C, AB\}$.*

Bereits in [AS94] wurde eine grundlegende Eigenschaft von häufigen Produktmengen entdeckt, die noch heute im Grundkern vieler Pattern-Mining-Algorithmen steckt.

Theorem 3.7. *Monotonie-Eigenschaft für häufige Produktmengen*
Sei $I_1 \subseteq P$ eine häufige Produktmenge einer Transaktionsdatenbank D. Jede nichtleere Teilmenge $I_2 \subseteq I_1$ ist dann ebenfalls eine häufige Produktmenge in D.

Beweis. Der Beweis folgt sofort aus der Definition. Wenn I_1 eine häufige Produktmenge von D ist, dann gilt

$$supp_D(I_1) \geq minsupp$$

und damit

$$\frac{abssupp_D(I_1)}{|D|} = \frac{|U_{I_1}|}{|D|} \geq minsupp.$$

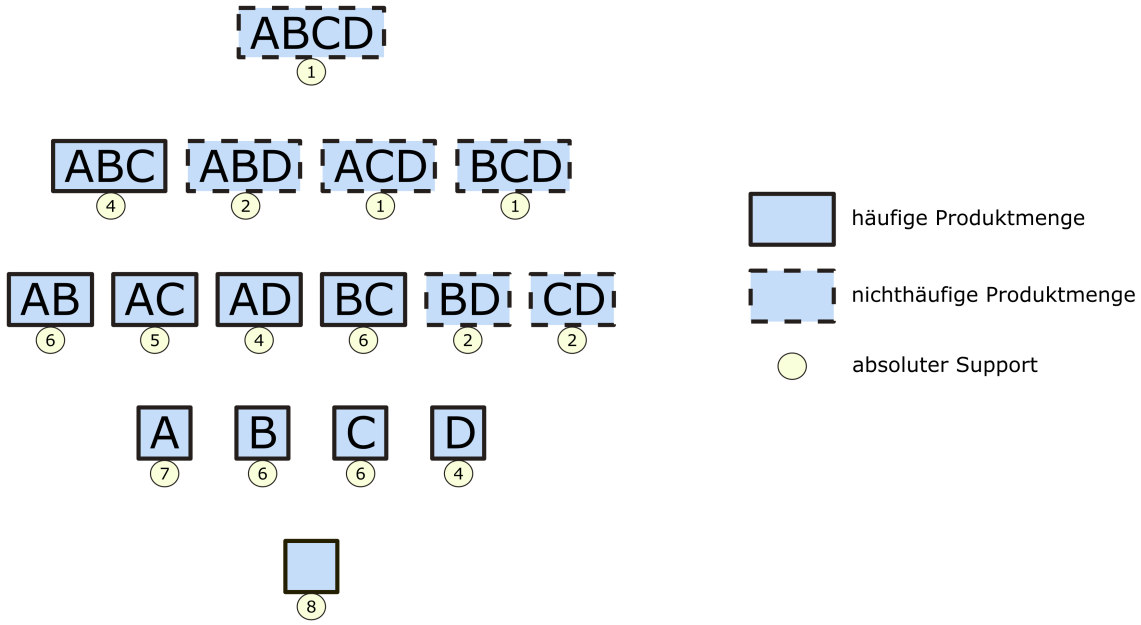
Mit $U_{I_1} = \{T \in D : I_1 \subseteq T\}$ und $I_2 \subseteq I_1$, gilt für alle $T \in U_{I_1} : I_2 \subseteq I_1 \subseteq T$.
Somit ist $|U_{I_2}| \geq |U_{i_1}|$ und damit

$$supp_D(I_2) \geq supp_D(I_1) \geq minsupp.$$

I_2 ist also eine häufige Produktmenge von D. □

Theorem 3.7 wird häufig auch **die Apriori-Eigenschaft** genannt, betitelt nach dem ersten Algorithmus zur Suche nach häufigen Produktmengen, der diese Eigenschaft nutzte.

Die Lösungsmenge der Problemstellung 3.5 kann in der Praxis sehr groß werden. Da sämtliche Teilmengen einer häufigen Produktmenge ebenso häufig sind, besitzt die Lösungsmenge mindestens 2^k Lösungen, wobei k die Länge der größten häufigen Produktmenge in ihr ist. Die Lösungsmenge wird in der Regel umso größer, je kleiner der Parameter *minsupp* gewählt wird, da hierdurch immer längere häufige Produktmengen erzeugt werden.



Beispielhafter Aufbau von häufigen und nichthäufigen Produktmengen in einer Datenbank mit absminsupp = 4

3.3 Häufige Assoziationsregeln

3.3.1 Definitionen und Problemstellung

Gegeben seien zwei Produktmengen I_1 und I_2 mit $I_1, I_2 \subseteq P$.

Definition 3.8. *Eine **Assoziationsregel** $A = (I_1 \Rightarrow I_2)$ über einer Datenbank D ist eine These*

$$(I_1 \subseteq T \Rightarrow I_2 \subseteq T) \text{ für alle } T \in D.$$

Anschaulich kann die These im Falle von Supermarkteinkäufen als "Kunden, die Produkte I_1 kauften, kauften auch Produkte I_2" formuliert werden. Die Aufstellung der These sagt jedoch noch nichts über ihren Wahrheitsgehalt oder ihre Überdeckung der Datenbank aus. Hierzu werden zwei Bewertungskriterien genutzt.

Definition 3.9. *Der **absolute Support** $abssupp_D(A)$ einer Assoziationsregel $A = (I_1 \Rightarrow I_2)$ in D ist die Mächtigkeit der Überdeckung von $I_1 \cup I_2$. Der **relative Support** einer Assoziationsregel A in D ist*

$$supp_D(A) = \frac{abssupp_D(A)}{|D|} = \frac{abssupp_D(I_1 \cup I_2)}{|D|}.$$

Der Support einer Assoziationsregel gibt die Häufigkeit an, mit der beide Produktmengen gleichzeitig in der Datenbank vorkommen. Dieser Wert kann als quantitative Wichtigkeit der Regel interpretiert werden.

Definition 3.10. *Die **Konfidenz** $conf_D(A)$ einer Assoziationsregel $A = (I_1 \Rightarrow I_2)$ in D ist die relative Häufigkeit der Gültigkeit der These. Es gilt*

$$conf_D(A) = \frac{abssupp_D(I_1 \cup I_2)}{abssupp_D(I_1)}.$$

Die Konfidenz ist ein Wert zwischen 0 und 1 und berechnet die qualitative Wertigkeit der Assoziationsregel (bedingte Wahrscheinlichkeit). Eine Konfidenz von $0,5$ bedeutet, dass die Regel in der Hälfte der Transaktionen der gesamten Datenbank D gültig ist, in der die Produkte I_1 enthalten sind. Es ist zu beachten, dass die Konfidenz nur definiert ist, wenn die Produkte I_1 in mindestens einer Transaktion in D enthalten sind.

Mit Hilfe des Definitionsgerüsts stellen wir nun die Problemstellung der Erkennung von häufigen Assoziationsregeln nach [AS95] auf.

Problemstellung 3.11. *Gegeben sei eine Produktmenge $P = \{p_1, p_2, \ldots, p_n\}$, eine Transaktionsdatenbank $D = \{T_1, T_2, \ldots, T_m\}$ mit $T_i \subseteq P$, ein Schwellwert $0 \leq minsupp \leq 1$ und ein Schwellwert $0 \leq minconf \leq 1$. Finde alle Assoziationsregeln $A_j = (I_1 \Rightarrow I_2)$ mit $I_1, I_2 \subseteq P$ und $I_1, I_2 \neq \emptyset$, für die gilt:*

$$(\exists\, T_i \in D : I_1 \subseteq T_i) \text{ und } supp_D(A_j) \geq minsupp \text{ und } conf_D(A_j) \geq minconf.$$

*Die A_j werden **häufige Assoziationsregeln** genannt.*

Beispiel 3.12. *Gegeben sei die Menge $D = \{ABC, AB, C\}$, $minsupp = 0,5$ und $minconf = 0,5$. Die Ergebnismenge der häufigen Assoziationsregeln Z mit $Z_j = (Z_j^1 \Rightarrow Z_j^2)$ und $Z_j^1, Z_j^2 \neq \emptyset$ ist $Z = \{A \Rightarrow B, B \Rightarrow A\}$.*

3.4 Maximale und geschlossene häufige Produktmengen

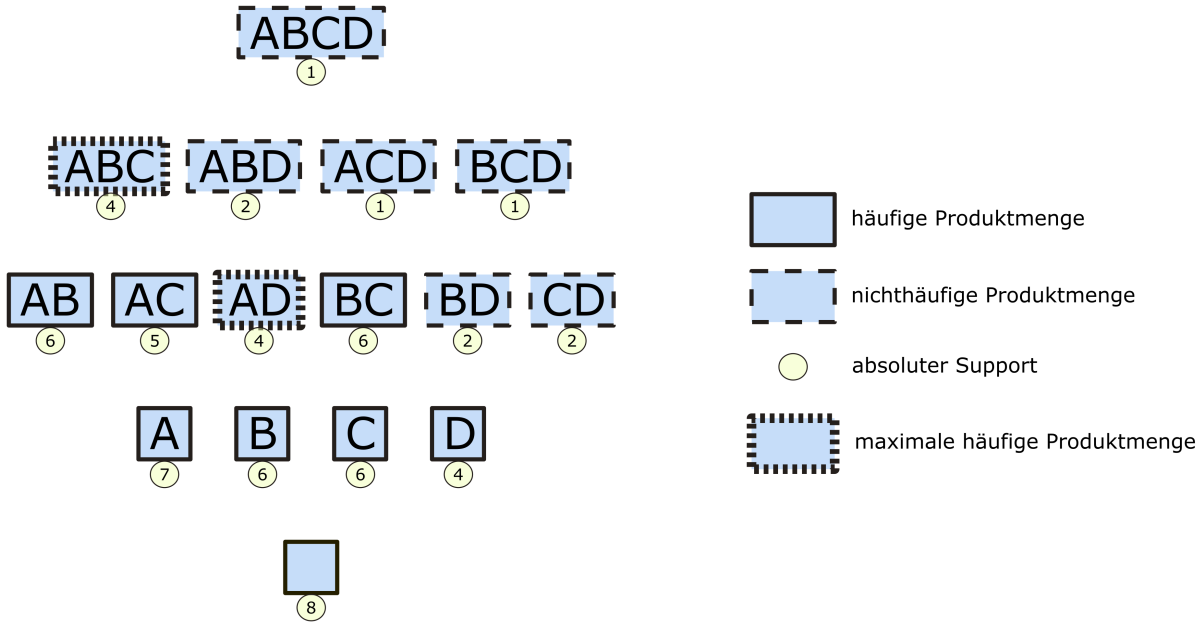
Die Erkennung von häufigen Produktmengen steht vor einem generellen Problem. Da jede Teilmenge einer häufigen Produktmenge selbst häufig ist, wird allein die Auflistung der Lösungsmenge zu einem ineffizienten Verfahren. Folglich wurde früh nach Verfahren gesucht, die die Anzahl der ausgegebenen Lösungen reduzieren, ohne zu viel Information über den gesamten Lösungsraum zu verlieren.

3.4.1 Maximale häufige Produktmengen

Definition 3.13. *Gegeben sei eine Transaktionsdatenbank D und die Menge der häufigen Produktmengen I von D. Die Menge der maximalen häufigen Produktmengen M ist*

$$M = \{A \in I : \nexists\, B \in I \text{ mit } A \subset B\}.$$

Maximale häufige Produktmengen besitzen keine Obermenge, die ebenfalls häufig ist. Anschaulich kann die Menge der maximalen häufigen Produktmengen als oberer Rand der Menge der häufigen Produktmengen verstanden werden. Insbesondere lassen sich alle häufigen Produktmengen aus ihr verlustfrei erzeugen, da I aufgrund der Monotonieeigenschaft von häufigen Produktmengen aus der Menge aller möglichen Teilmengen von Mengen in M besteht. Als Nachteil geht bei der Ausgabe der maximalen häufigen Produktmengen allerdings die Information über den Support der Teilmengen dieser Mengen verloren.



Beispielhafter Aufbau einer Menge von maximalen häufigen Produktmengen in einer Datenbank mit absminsupp = 4

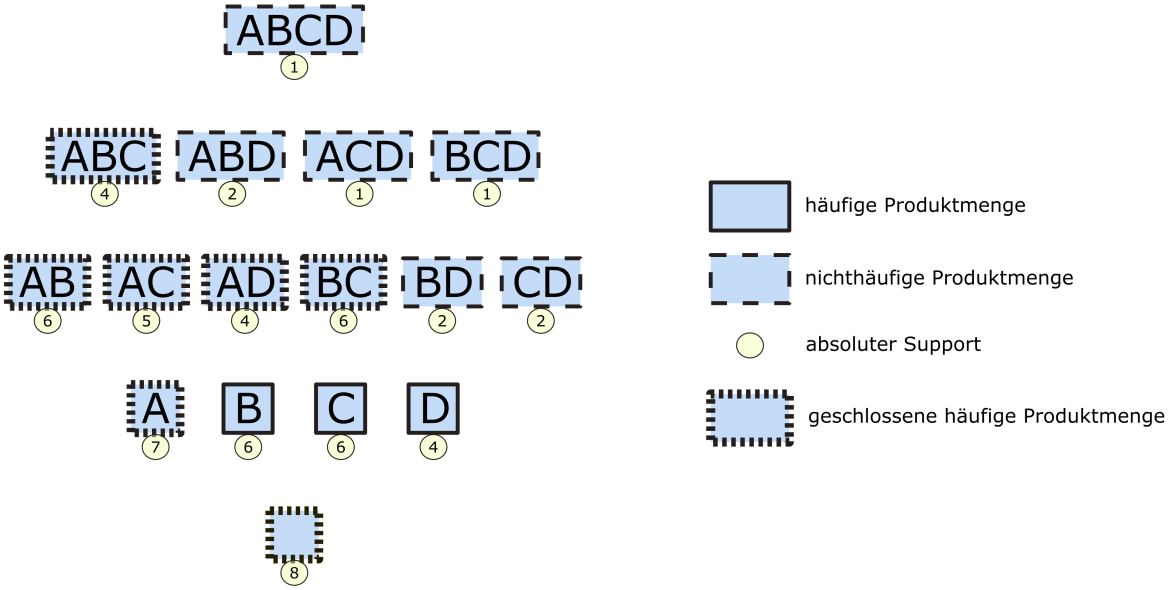
3.4.2 Geschlossene häufige Produktmengen

Definition 3.14. *Gegeben sei eine Transaktionsdatenbank D und die Menge der häufigen Produktmengen I von D. Die Menge der geschlossenen häufigen Produktmengen C ist*

$$C = \{A \in I : \nexists\, B \in I \text{ mit } (A \subset B \text{ und } supp_d(A) = supp_d(B))\}.$$

Geschlossene häufige Produktmengen können als maximale häufige Produktmengen in der Äquivalenzklasse der häufigen Produktmengen mit gleichem Support verstanden werden. Insbesondere sind alle globalen maximalen häufigen Produktmengen auch geschlossene häufige Produktmengen. Die Umkehrung gilt jedoch nicht.

Die Ausgabe der Menge der geschlossenen häufigen Produktmengen hat den Vorteil, dass sie die Information über den Support ihrer Teilmengen vollständig speichert. Zur Ermittlung des Supports einer häufigen Produktmenge reicht es aus, die kleinste geschlossene häufige Produktmenge zu suchen, die die Menge beinhaltet.



Beispielhafter Aufbau einer Menge von geschlossenen häufigen Produktmengen in einer Datenbank mit absminsupp = 4

Bekannte Algorithmen zur Bestimmung von geschlossenen häufigen Produktmengen sind zum Beispiel CHARM [ZH02] und CLOSET+ [WHP03].

3.5 Generatoren

Eine weitere Möglichkeit zur Bestimmung von häufigen Teilmengen ist die Konstruktion von Generatoren [BPT+00] und negativen [BR01] [Kry01] [KG02] bzw. positiven Rändern [LLW08]. Die Nutzung von Generatoren gibt Verfahren die Möglichkeit, statt häufiger Teilmengen nur einen geringen generierenden Kern der häufigen Teilmengen zu berechnen. Hierzu werden Generatoren so lange durch einen festgelegten Algorithmus erweitert, bis die ebenfalls bekannten negativen oder positiven Ränder der häufigen Teilmengen erreicht werden. Dies gibt dem Benutzer die Möglichkeit, nur die häufigen Teilmengen zu generieren, die durch eine Vorauswahl der Generatoren interessant erscheinen.

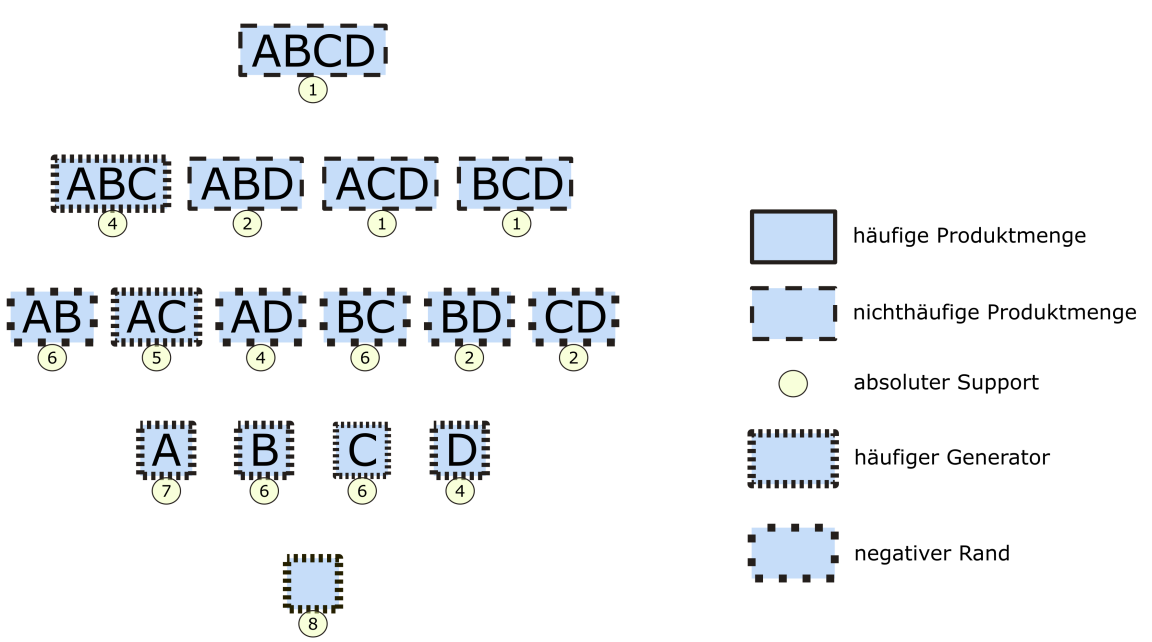
Definition 3.15. *Gegeben sei eine Transaktionsdatenbank D. Eine Produktmenge $A \subseteq P$ heißt **Generator**, wenn gilt: $\nexists\, B \subset A$ mit $supp_D(A) = supp_D(B)$. Ist A eine häufige Produktmenge, so wird A auch häufiger Generator genannt.*

Der Begriff "Generator" signalisiert, dass sich Mengen aus dieser Menge nach einem festgelegten Verfahren generieren lassen. Hierzu ist es noch notwendig, einem solchen Verfahren mitzuteilen, wann es die Generierung stoppen soll.

Definition 3.16. *Sei I die Menge aller häufigen Produktmengen einer Datenbank D über der Menge der Produkte P und $0 \leq minsupp \leq 1$ ihr minimaler Support. Sei weiterhin G die Menge ihrer häufigen Generatoren. Dann ist der **negative Rand** von G definiert als*

$$R_G^- = \{A \subseteq P : A \notin G \text{ und } (\forall\, B \subset A : B \in G)\}\,.$$

Da der negative Rand nur aus Mengen besteht, die keine Generatoren sind, folgt daraus, dass $R_G^- \cap G = \emptyset$.

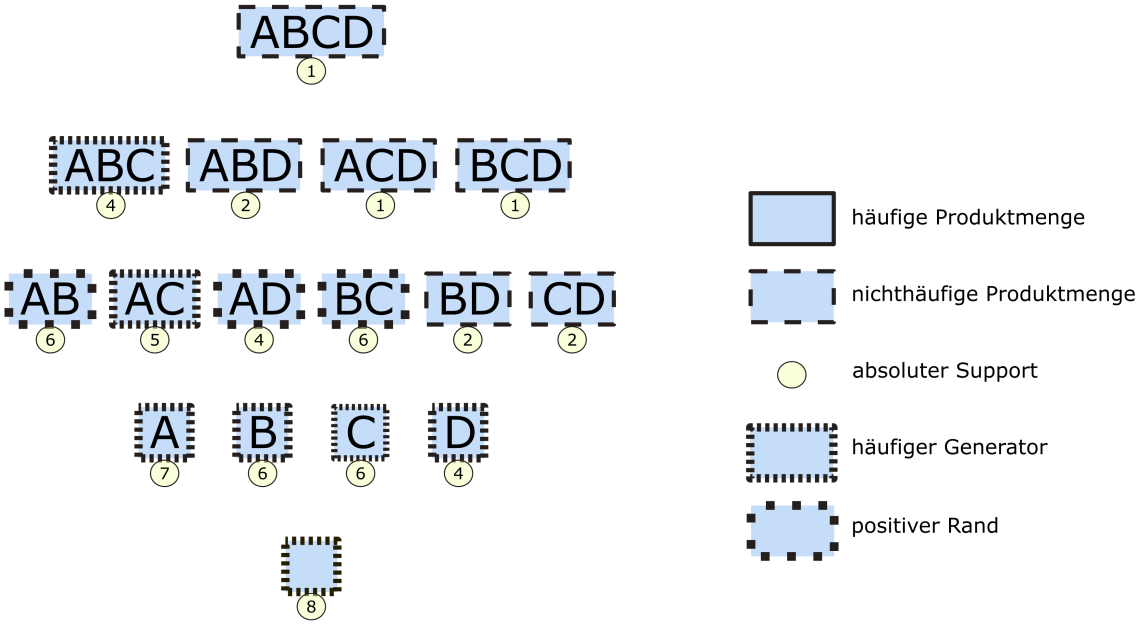


Beispielhafter Aufbau einer Menge von häufigen Generatoren und ihrem negativen Rand in einer Datenbank mit absminsupp = 4.

Definition 3.17. *Sei I die Menge aller häufigen Produktmengen einer Datenbank D und $0 \leq minsupp \leq 1$ ihr minimaler Support. Sei weiterhin G die Menge ihrer häufigen Generatoren. Dann ist der* ***positive Rand*** *von G definiert als*

$$R_G^+ = \{A \in I : A \notin G \text{ und } (\forall\, B \subset A : B \in G)\}\,.$$

Die Bezeichnung “Rand” ist bewusst bildlich gewählt, um zu visualisieren, dass es sich sowohl beim negativem als auch beim positiven Rand um Mengen handelt, die die Mengen der häufigen Produktmengen umgeben und als eine Grenze dienen, deren Überschreitung zu einem Verlassen des Gebiets der häufigen Produktmengen führt. Der Unterschied besteht darin, dass der positive Rand noch innerhalb der häufigen Produktmengen liegt, während der negative Rand diese Menge von außen umgibt.



Beispielhafter Aufbau einer Menge von häufigen Generatoren und ihrem positiven Rand in einer Datenbank mit absminsupp = 4.

Theorem 3.18. *Gegeben sei die Menge der häufigen Generatoren G und die Menge der häufigen Produktmengen I einer Datenbank D. Dann gilt*

$$|G| + |R_G^+| \leq |I|.$$

Beweis. Der Beweis folgt sofort aus den Definitionen. Wir wissen

1. $G \subseteq I$, da alle Generatoren in G häufig sind,
2. $R_G^+ \subseteq I$, da für alle $A \in R_G^+$ gilt: $A \in I$,
3. $G \cap R_G^+ = \emptyset$, da für alle $A \in R_G^+$ gilt: $A \notin G$.

Aus allen drei Aussagen folgt die Aussage des Theorems. □

Die Menge der häufigen Generatoren und ihres positiven Randes ist also nie größer als die Menge der häufigen Produktmengen. Gleiches kann man über die Menge der häufigen Generatoren und ihres negativen Randes nicht beweisen. Tatsächlich kann der negative Rand mehr Elemente besitzen als die Menge der häufigen Teilmengen, wodurch er für einen allgemeinen Algorithmus kontraproduktiv werden kann.

In der Praxis kann mit Generatoren oft eine erhebliche Reduktion des Speicherbedarfs erreicht werden, insbesondere in Verbindung mit einem positiven Rand, bei dem der Worst-Case nicht schlechter als die direkte Suche nach häufigen Produktmengen ist. Trotzdem lässt sich aus der Menge der häufigen Generatoren und ihres positiven Rands die Menge der häufigen Produktmengen verlustfrei erzeugen.

Theorem 3.19. *Gegeben sei die Menge der häufigen Generatoren G einer Datenbank D über den Produkten P, der positive Rand R_G^+ und der Support für alle Produktmengen in $G \cup R_G^+$. Dann kann für jede Menge $I \subseteq P$ bestimmt werden*

1. *ob I häufig ist,*
2. *der Support von I, falls I häufig ist.*

Für den Beweis wird hier auf [LLW08] verwiesen. Ein entsprechender Algorithmus ist im Abschnitt 3.7.3 zu finden.

3.6 Top-k-häufige Produktmengen

Die Suche nach häufigen Teilmengen musste sich lange Zeit der Kritik stellen, dass die Definition der Häufigkeit einen externen Parameter *minsupp* voraussetzt, der von den Benutzern des Algorithmus im Vorfeld festgelegt werden muss. Jedoch ist es selbst bei Expertenwissen über das Anwendungsgebiet keinesfalls klar, in welchem Rahmen sich dieser Parameter bewegt. Hierdurch müssen die Verfahren in der Regel mehrfach durchlaufen werden, bis eine zufriedenstellende Lösung eintritt. Um diese Probleme zu vermeiden, wurden die Suche nach *Top-k häufigen Teilmengen* durch einen Apriori-Ansatz eingeführt [FKT00]. Hierbei gibt der Benutzer des Verfahrens keine untere Häufigkeitsgrenze vor, sondern wählt die Anzahl der gewünschten Lösungen mit höchster Häufigkeit. Zwar wird auch in diesem Fall ein externer Parameter angegeben, jedoch lässt sich das Wachstum der Ausgabe hiermit linear steuern. Spätere Verfahren nutzen hierzu Pattern-Growth-Ansätze [CF04] [WHLT05], COFI-Bäume [NLWF05] [CXCS11] oder dimensionsreduzierende Heuristiken [SK12].

Definition 3.20. *Eine Produktmenge I heißt* ***top-k-häufig*** *in einer Transaktionsdatenbank D, wenn nicht mehr als $k-1$ Produktmengen existieren, deren Support in D größer ist als der Support von I in D.*

Die Definition impliziert, dass tatsächlich mehr als k k-häufige Produktmengen existieren können, sofern mehrere Produktmengen den identischen minimalen Support aufweisen.

Das Konzept von top-k-häufigen Produktmengen lässt sich analog auf alle engeren Definitionen erweitern. So heißt eine maximale häufige Produktmenge top-k-maximal-häufig, wenn nicht mehr als $k-1$ maximale Produktmengen existieren, deren Support größer ist.

Die Definition von Top-k-Häufigkeit stellt sicher, dass während des Schürfschritts höchstens k Endergebnisse herausgeschrieben werden müssen. Gleichermaßen führt die Definition dazu, dass die Anforderungen an die Ergebnismenge mit gefundenen Produktmengen stetig steigen können.

Theorem 3.21. *Gegeben sei eine Datenbank D und eine beliebige Menge an Produktmengen $A = \{A_1 \ldots, A_k\}$. Sei*

$$minsupp_D(A) = \min_{A_i \in A} supp_D(A_i).$$

Dann ist sichergestellt, dass für alle top-k-häufigen Produktmengen I von D gilt:

$$supp_D(I) \geq minsupp_D(A).$$

Beweis. Der Beweis folgt sofort aus der Definition. Angenommen, es gäbe eine top-k-häufige Produktmenge I von D mit $supp_D(I) < minsupp_D(A)$. Dann hätten alle k Mengen in A einen höheren Support als I. Dies steht im direkten Widerspruch zur Top-k-Häufigkeit von I. Folglich gilt für alle top-k-häufigen Produktmengen I von D

$$supp_D(I) \geq minsupp_D(A).$$

□

Aus Theorem 3.21 lässt sich ohne größere Probleme ein naiver Algorithmus zur Konstruktion aller top-k-häufigen Produktmengen formen.

Der Algorithmus legt einen leeren Speicher von Produktmengen an. Anschließend durchforstet er alle möglichen Produktmengen der Datenbank. Solange der Speicher noch nicht mit k Produktmengen gefüllt ist, fügt er die aktuelle Produktmenge dem Speicher hinzu. Ist der Speicher bereits mit k Produktmengen gefüllt, fügt er die aktuelle Produktmenge dem Speicher hinzu, wenn der Support der Produktmenge größer ist als der Support mindestens einer Produktmenge im Speicher. In diesem Fall werden die Produktmengen mit niedrigstem Support im Speicher entfernt, sofern die Größe des Speichers damit nicht unter k Produktmengen fällt.

Effizientere Algorithmen nutzen Verfahren, mit denen frühzeitig größere Teilmengen des Suchraumes ausgeschlossen werden können. Es bleibt zu bemerken, dass aufgrund der Monotonieeigenschaft von häufigen Produktmengen viele Produktmengen mit sehr wenigen Elementen in der Lösungsmenge vorkommen werden. Aus diesem Grund ist es ratsam, die Suche nach top-k-häufigen Produktmengen mit weiteren Kriterien wie einer minimalen Länge der gewünschten Produktmengen zu kombinieren. Ein solches Verfahren wird in Abschnitt 3.7.4 mit dem TFP-Algorithmus vorgestellt.

3.7 Ausgewählte Verfahren

3.7.1 Der Apriori-Algorithmus

Der Apriori-Algorithmus [RAS93] [AS94] [SA00] war einer der ersten Algorithmen zur Bestimmung von häufigen Produktmengen und häufigen Assoziationsregeln, der die Monotonie-Eigenschaften nutzte. Er lässt sich grob in zwei Phasen einteilen.

1. Finde alle häufigen Produktmengen innerhalb einer Datenbank.
2. Nutze die gefundenen häufigen Produktmengen zur Bestimmung von häufigen Assoziationsregeln.

Erkennung häufiger Produktmengen

Der Apriori-Algorithmus konstruiert häufige Produktmengen iterativ durch Vereinigung von häufigen Produktmengen kleinerer Mächtigkeit. Da jede Teilmenge einer häufigen Produktmenge ebenfalls eine häufige Produktmenge ist, bedeutet dies, dass nicht-häufige Produktmengen bei der Vereinigung zu größeren Produktmengen ausgeschlossen werden können, ohne in die Gefahr zu laufen, bei der Vereinigung eine häufige Produktmenge zu übergehen.

Zunächst werden in einer Datenbank alle häufigen Produktmengen der Mächtigkeit 1 ermittelt. Dies kann durch einen einfachen Scan der Datenbank erfolgen. Als nächstes werden in einem Vereinigungsschritt alle möglichen Produktmengen der Mächtigkeit 2 ermittelt, die sich aus der Vereinigung von häufigen Produktmengen der Mächtigkeit 1 bilden lassen. Man spricht in diesem Fall von einer **Kandidatengenerierung**. Die gebildeten Vereinigungen müssen aber keine häufigen Produktmengen sein. Aus diesem Grund wird in einem Verifikationsschritt über einen Scan der Datenbank der Support aller Kandidaten bestimmt. Alle Kandidaten, die unterhalb des minimalen Supports für häufige Produktmengen liegen, werden aus der Kandidatenmenge entfernt. Übrig bleiben die häufigen Produktmengen der Mächtigkeit 2.

Dieses Verfahren wird so lange wiederholt, bis entweder die Mächtigkeit der Grundmenge der Produkte erreicht wird oder aber keine häufige Produktmenge der Mächtigkeit k mehr gefunden werden kann. Die Lösungsmenge der Problemstellung ist nun die Menge der häufigen Produktmengen der Mächtigkeit 1 bis $k-1$.

Wenn ein Scan der Datenbank zu aufwändig ist, so reicht es für die Verifizierung einer Produktmenge der Mächtigkeit k als häufig, die Eigenschaft zu zeigen, dass alle Teilmengen der Mächtigkeit $k-1$ häufig sind. Da bereits alle häufigen Produktmengen der Mächtigkeit $k-1$ vorliegen, kann die Verifikation durch eine simple Existenzabfrage erfolgen.

Pseudocode zur Erkennung häufiger Produktmengen

Hinweis: Innerhalb des Pseudocodes wird von einer lexikographischen Ordnung von Produktmengen ausgegangen.

Data: Grundmenge P, Datenbank D, Häufigkeitsschranke $minsupp$
Result: Menge der häufigen Produktmengen I
$C_1 = \{\{p\} : p \in P\}$;
$k = 1$;
while $C_k \neq \emptyset$ **do**
  forall $D_j \in D$ **do**
    forall $C \in C_k$ **do**
      if $C \subseteq D_J$ **then**
        $abssupp_D(C) += 1$;
      end
    end
  end
  $I_k = \{C \in C_k : supp_D(C) \geq minsupp\}$;
  $I = I \cup I_k$;
  forall $X = \{x_1, \ldots, x_k\}, Y = \{y_1, \ldots, y_k\} \in I_k$ **do**
    if $X \setminus \{x_k\} = Y \setminus \{y_k\}$ *AND* $x_k < y_k$ **then**
      $Z = X \cup \{y_k\}$;
      if $\forall A \subset Z$ *mit* $|A| = k : A \in I_k$ **then**
        $C_{k+1} = C_{k+1} \cup Z$;
      end
    end
  end
  $k += 1$;
end

Algorithm 1: Bestimmung häufiger Produktmengen durch den Apriori-Algorithmus

Erkennung häufiger Assoziationsregeln

Mit der Kenntnis der häufigen Produktmengen können häufige Assoziationsregeln bestimmt werden. Da der Support einer Assoziationsregel $A = (I_1 \Rightarrow I_2)$ dem Support von $I_1 \cup I_2$ entspricht, wird jede häufige Produktmenge als Vereinigung der Mengen einer Assoziationsregel interpretiert. Folglich können häufige Assoziationsregeln durch Trennung einer häufigen Produktmenge in zwei beliebige Teilmengen gebildet werden. Im Falle einer Konfidenz-Vorgabe muss jedoch für jedes mögliche Paar an Teilmengen noch eine Konfidenz-Kontrolle durchgeführt werden.

Bewertung des Algorithmus

Mit Hilfe des Apriori-Algorithmus ist es grundsätzlich möglich, alle häufigen Produktmengen einer Datenbank zu ermitteln und hierzu frühzeitig alle nichtbenötigten Teilmengen auszusortieren. Jedoch steht der Algorithmus vor zwei größeren Komplexitätsproblemen.

1. Da der Apriori-Algorithmus häufige Produktmengen aus der Vereinigung

von Teilmengen aufbaut, muss er zum Finden einer häufigen Produktmenge der Mächtigkeit k zwingend vorher alle 2^k Teilmengen dieser Produktmenge ermittelt haben, die ebenso häufige Produktmengen sind.

2. Algorithmen, die alle häufigen Produktmengen einer Datenbank finden wollen, stehen grundsätzlich vor dem Problem, dass die Lösungsmenge im Worst-Case exponentiell wächst. [Yan04]

Ob der Apriori-Algorithmus eine Datenbank in der Praxis handhaben kann, hängt von der Länge der zu erwartenden häufigen Produktmengen ab. Weniger Relevanz hat die Anzahl der Transaktionen der Datenbank, die ausschließlich eine lineare Komplexitätssteigerung verursacht.

3.7.2 Der MAFIA-Algorithmus

MAFIA [BCF+05] steht für "Maximal Frequent Itemset Algorithm". Zwar war er nicht der erste Algorithmus zur Erkennung von maximalen häufigen Teilmengen [LK98] [Zak00a], aber er war einer der ersten Algorithmen, die zur Beschneidung des Suchbaumes ein Tiefenverfahren nutzten.

Der MAFIA-Algorithmus erstellt im Kern einen lexikographisch geordneten Suchbaum aller häufigen Produktmengen. Er startet mit einer Produktmenge bestehend aus dem ersten geordneten Produkt und durchläuft die Baumelemente in einem Tiefensuchverfahren. Dies bedeutet, dass er zunächst für jeden Knoten Nachfolger bildet, indem er die bestehende Produktmenge jeweils um ein weiteres Produkt erweitert. Erst wenn keine Erweiterung mehr möglich ist, kehrt er zurück zu einer höheren Stufe und erweitert eine andere Produktmenge um zusätzlich Elemente.

Sobald ein Knoten des Baumes einen Support unter $minsupp$ aufweist, wird dieser Teil des Baumes abgeschnitten, denn aufgrund der Monotonie-Eigenschaft von häufigen Mengen ist sichergestellt, dass alle Obermengen dieser Produktmenge nicht häufig sind. Ist die Monotonie-Eigenschaft erfüllt, so wird als Unterknoten dieses Knotens eine Obermenge der aktuellen Produktmenge gebildet, ergänzt um das nächste Produkt laut lexikographischer Ordnung.

Erweiterte Beschneidung des Suchbaumes

Bis hierhin wurde noch kein Gebrauch von der Definition von maximalen häufigen Produktmengen gemacht, die das Wachstum der Knoten im Suchbaum signifikant beschränken kann. MAFIA nutzt hierzu erweiterte Beschneidungstechniken.

- Steht ein Knoten für die Produktmenge I und existiert ein Element p des Unterbaumes von I mit den Überdeckungen $U_I \subseteq U_{\{p\}}$, so kann I keine maximale Produktmenge sein, da alle häufigen Produktmengen, die I enthalten ebenfalls p enthalten. In diesem Fall kann der Knoten I durch den

Knoten $J = I \cup \{p\}$ ersetzt werden und alle übrigen Teilbäume von I müssen nicht durchgetestet werden.

- Bereits in [Bay98] wird gezeigt, dass die größtmögliche im Suchbaum erzeugte Untermenge eines Knoten P das erste Baumblatt ist, welches von einem Tiefensuchalgorithmus erzeugt wird. Es enthält alle übrigen Produkte, die nach lexikographischer Ordnung noch auftreten können. Stellt sich dabei heraus, dass diese größtmögliche Untermenge häufig ist, können alle anderen Unterzweige des Knotens abgeschnitten werden, da sie in dieser Menge enthalten sind und somit nicht maximal sind. Dies gilt ebenso, wenn diese größtmögliche Untermenge bereits in einer anderen maximalen häufigen Produktmenge enthalten ist.

Bewertung des Algorithmus

Der Mafia-Algorithmus ist ein geeignetes Verfahren, um mit möglichst wenig Scans der Datenbank alle maximalen häufigen Produktmengen in ihr zu generieren, da er auf einer vertikalen Datenbankrepräsentation aufsetzt. Er vermeidet redundante Information durch den Verzicht auf die Auflistung aller möglichen Teilmengen der maximalen häufigen Produktmengen, die aus der Monotonie-Eigenschaft heraus ebenso häufig sind. Im Gegensatz zu den geschlossenen häufigen Produktmengen muss man bei den maximalen häufigen Produktmengen jedoch damit leben, dass die Information über den Support einer häufigen Produktmenge verloren geht. Man erhält nur die Garantie, dass der minimale Support nicht unterschritten wird.

In der Theorie kann nicht garantiert werden, dass die Menge der maximalen häufigen Produktmengen echt kleiner ist als die Menge der häufigen Produktmengen. In der Praxis hat die Bestimmung von maximalen häufigen Produktmengen aber in den Fällen sehr große Performancevorteile, in denen häufige Produktmengen mit sehr vielen Elementen erwartet werden.

3.7.3 Der GrGrowth-Algorithmus

Der GrGrowth-Algorithmus [LLW08] ist ein Verfahren zur Suche nach allen häufigen Produktmengen einer Datenbank, das Generatoren und einen positiven Rand zur Beschreibung der ausgegebenen Lösungsmenge nutzt. Statt die vollständige Lösungsmenge auszugeben, werden nur die notwendigsten Informationen ermittelt, um alle häufigen Produktmengen aus ihnen zu generieren.

Der Algorithmus setzt im Kern auf einem Wachstumsverfahren (Pattern-Growth) auf. In dieser Art von Verfahren wird ein Suchbaum von projizierten Datenbanken erstellt. Jeder Knoten des Suchbaumes entspricht einem Element aus der Grundmenge der Produkte. Ein Pfad in diesem Suchbaum entspricht einer Produktmenge in der lexikographischen Ordnung der Produkte. Der Support eines Knotens ergibt sich demnach als der Support der Produktmenge, die durch das Durchlaufen des Pfades von der Wurzel bis zu diesem Knoten

entsteht. Die projizierte Datenbank entspricht hierbei der Überdeckung der Produktmenge, besteht also aus allen Transaktionen der Datenbank, die diese Produktmenge beinhalten.

Durch die Konstruktion des Suchbaumes ist bereits in der Theorie ein Algorithmus zur Ausgabe aller häufigen Produktmengen implementiert. Da der Support von Produktmengen mit wachsender Anzahl gleicher Elemente immer geringer wird, muss der Baum von der Wurzel aus nur so lange durchlaufen werden, bis in einem Ast des Baumes der minimale Support unterschritten wird.

Der Suchbaum der häufigen Produktmengen wird durch ein Tiefensuchverfahren durchlaufen. In dieser Phase werden alle Generatoren für häufige Produktmengen gesucht und alle Äste des Baumes abgeschnitten, die später durch Generatoren wieder dynamisch reproduziert werden können. Hierbei kann auch auf natürliche Weise der positive Rand der Generatoren ermittelt werden, indem vor dem Abschneiden der Nichtgeneratoren der Rand in einer separaten Liste gespeichert wird.

Nach der Ausgabe der Generatoren und ihrem positiven Rand kann die Menge der häufigen Produktmengen durch einen einfachen Zusatzalgorithmus wieder erzeugt werden. Dabei werden die Generatoren solange um zusätzliche häufige Produkte erweitert, bis einer ihrer positiven Ränder erreicht wird.

Bewertung des Algorithmus

Der GrGrowth-Algorithmus versucht im Kern das exponentielle Wachstum der Lösungsmenge vom Generierungsschritt in den Reproduktionsschritt zu verlegen. Obwohl hierdurch zunächst keine Effizienz gewonnen wird, kann dieser Ansatz in der Praxis hilfreich sein. Dies ist der Fall, wenn durch die Sichtung der Generatoren bereits durch Expertenwissen Generatoren selektiert werden können, deren weiteres Wachstum interessant genug erscheint, um die erweiterte Laufzeit zu rechtfertigen. Auf diese Weise kann die Reproduktion von bereits bekannter oder als unwichtig bewerteter Information reduziert werden. Allerdings kann auch der GrGrowth-Algorithmus nicht garantieren, dass die Menge der Generatoren und ihres positiven Randes im schlechtesten Fall nicht genauso schnell wächst wie die Menge der häufigen Produktmengen.

3.7.4 Der TFP-Algorithmus

Der TFP-Algorithmus (Top Frequent Pattern Algorithm) [WHLT05] ist ein Algorithmus zur Suche nach allen top-k-häufigen geschlossenen Produktmengen einer Datenbank D, die eine vorgegebene minimale Länge besitzen. Er nutzt hierzu wie der GrGrowth-Algorithmus im Kern einen Pattern-Growth-Ansatz und konstruiert einen Baum von projizierten Teildatenbanken.

Der Algorithmus ist in der Lage mehrere Beschneidungsstrategien zu ver-

wenden:

1. Es werden nur geschlossene Produktmengen gesucht. Geschlossene Produktmengen bieten den Vorteil, dass sie maximal unter den Produktmengen mit gleichen Support sind. Kleinere Produktmengen mit gleichem Support müssen also nicht betrachtet werden und können frühzeitig aus dem Baum entfernt bzw. fusioniert werden.

2. Es werden nur geschlossene Produktmengen einer minimalen Länge l gesucht. Als Folge hieraus kann die Datenbank um alle Transaktionen bereinigt werden, die nicht mindestens l Elemente besitzen, da diese nicht den Support einer Produktmenge der minimalen Länge l erhöhen können.

3. Aufgrund der Suche nach top-k-häufigen Produktmengen ist der Algorithmus in der Lage, frühzeitig Teilbäume des Suchbaumes abzuschneiden, wenn sichergestellt ist, dass keine geschlossene Produktmenge des Teilbaumes einen Support besitzen kann, der höher ist als der geringste Support der bisher gefundenen k geschlossenen Produktmengen mit höchstem Support. Er nutzt hierzu die Erkenntnisse aus Theorem 3.21.

Bewertung des Algorithmus

Der TFP-Algorithmus nutzt den Vorteil von Verfahren zur Ermittlung von top-k-häufigen Produktmengen, die keine Vorgabe eines minimalen Supportes benötigen. Statt dessen kann intuitiv eine Anzahl von gewünschten Lösungen vorgegeben werden.

Die Wahl einer minimalen Länge von Produktmengen bietet die Möglichkeit, sich auf wesentliche Beziehungen zu fokussieren. Dieser Parameter ist in einem Top-k-Häufigkeitsansatz aus praktischer Perspektive zwingend notwendig, denn aufgrund der Monotonieeigenschaft von häufigen Produktmengen, werden längere Produktmengen ansonsten aufgrund ihres niedrigeren Supports nicht in der Lösungsmenge enthalten sein. Allerdings ist in einer komplexeren Datenbank keinesfalls im Voraus klar, welche Werte dieser Parameter sinnvollerweise annehmen sollte.

Der Algorithmus ist durch die Wahl der Eingabeparameter und die Natur von geschlossenen Produktmengen fähig, das Wachstum des Suchbaumes stark zu beschränken, was sich in einem besseren Laufzeit- und Speicherverhalten bemerkbar macht. Er kann jedoch, wie alle hier vorgestellten Algorithmen, nicht ausschließen, dass im Worst-Case ein exponentielles Laufzeitverhalten bei der Suchbaumkonstruktion erreicht wird.

Kapitel 4

Mustererkennung über Sequenzen

4.1 Einleitung

Die Mustererkennung über Sequenzen beschäftigt sich mit dem Schürfen von Mustern in Datenstrukturen, die eine geordnete, eindimensionale Folge von Elementen oder Mengen von Elementen abbilden. Häufig handelt es sich bei der grundliegenden Struktur um eine zeitliche Ordnung. So können die bereits im vorherigen Artikel angesprochenen gemeinsamen Produkte bei einem Einkauf aus dem Blickwinkel eines einzelnen Kunden über die Zeit betrachtet werden. Während der Kunde beim ersten Einkauf einen Rasierer und Rasierschaum kauft, wird er beim nächsten Einkauf vielleicht neue Rasierklingen benötigen. Obwohl Rasierer und Rasierklingen nicht im gleichen Einkauf zusammen besorgt werden, verbindet sie dennoch ein kausaler Zusammenhang, der bei einer nichtzeitlichen Betrachtung der Einkäufe verloren gehen würde.

Die praktische Anwendung einer Mustererkennung über Sequenzen beschränkt sich aber nicht auf Zeitabläufe. So können ebenso DNS-Stränge auf wiederkehrende Muster oder Textabschnitte auf wiederkehrende Ausdrücke untersucht werden.

Die Problemstellung des SPM wurde als erstes von [AS95] aufgestellt und seitdem um zahlreiche Modifikationen ergänzt. Bis zum heutigen Tage bleibt das Thema ein aktuelles Forschungsgebiet [FVLKK17] [KN17].

4.2 Häufige Sequenzen

4.2.1 Definitionen und Problemstellung

Gegeben sei eine Menge $P = \{p_1, p_2, \ldots, p_n\}$ an Produkten. Eine **Produktmenge** $T \subseteq P$ ist, wie bereits im letzten Kapitel definiert, eine nichtgeordnete Menge von Produkten.

Definition 4.1. *Eine **Sequenz** $S =< s_1, s_2, \ldots, s_m >$ mit $s_i \subseteq P$ ist eine endliche, geordnete Folge von Produktmengen. Die Länge $|S| = m$ einer Sequenz ist definiert als*

die Anzahl ihrer Sequenzglieder. Wir schreiben $a \in S$, wenn $a \in \{s_1, s_2, \ldots, s_m\}$.

In erklärenden Texten und Beispielen wird aus Übersichtsgründen eine vereinfachte Schreibform angewandt. Seien $A, B, C \in P$, so ist die Schreibweise $S = AB(BC)A$ äquivalent zu

$$S =< \{A\}, \{B\}, \{B, C\}, \{A\} > .$$

Um Muster in Sequenzen erkennen zu können, bedarf es einem Grundgerüst an Definition, die sich in ihrem Wesen stark an den entsprechenden Definitionen für die Mustersuche über Mengen orientieren.

Definition 4.2. *Eine Sequenz $S_a =< a_1, a_2, \ldots, a_n >$ ist eine **Teilsequenz** von $S_b =< b_1, b_2, \ldots, b_m >$, wenn natürliche Zahlen $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ existieren, so dass gilt $a_1 \subseteq b_{i_1} \wedge a_2 \subseteq b_{i_2} \wedge \ldots \wedge a_n \subseteq b_{i_n}$. S_b wird analog **Supersequenz** von S_a genannt und mit $S_a \subseteq S_b$ beschrieben.*

Es muss beachtet werden, dass die Definition einer Teilsequenz einige Sonderfälle ausschließt. Beispielsweise ist $A(BC)$ eine Teilsequenz von $AB(BC)A$ und $(AB)(ABC)$, aber nicht von ABC. Die Trennung in unterschiedliche Folgenglieder kann also nicht durch ein Vorkommen in einem einzelnen Glied aufgehoben werden. Ebenso kann eine beliebige Anzahl von Gliedern in der Superfolge existieren, die zwischen zwei Folgengliedern in der Teilfolge liegt. Muster mit äquidistanten Abständen sind hiermit nicht gesondert auffindbar.

Definition 4.3. *Die Menge aller zur Verfügung stehenden Transaktionssequenzen $D = \{S_1, \ldots, S_m\}$ heißt **Transaktionsdatenbank**.*

Definition 4.4. *Der **absolute Support** $abssupp_D(T)$ einer Sequenz T in D ist die Anzahl der Sequenzen S_i in D, für die gilt: T ist Teilsequenz von S_i. Der **relative Support** einer Sequenz T in D ist*

$$supp_D(T) = \frac{abssupp_D(T)}{|D|}.$$

Die Definition des Supports erfasst pro Transaktion nur das einmalige Auftreten einer Sequenz. So erhält die Sequenz A für das Auftreten in der Transaktion AAA nur den absoluten Support 1. In Beispielen und Algorithmen wird abkürzend die Schreibweise $AAA : 1$ angewandt, um eine Sequenz mitsamt ihres (absoluten) Supports anzugeben.

Einleitende Definitionen ermöglichen es uns nun, die Aufgabenstellung des SPM (Sequential Pattern Mining) nach [AS95] zu formulieren.

Problemstellung 4.5. *Gegeben sei eine Menge von Sequenzen $D = \{S_1, \ldots, S_n\}$ und ein Schwellwert $0 \leq minsupp \leq 1$. Finde alle Sequenzen $T_i =< t_1^i, \ldots, t_m^i >$ mit*

$$t_1^i, \ldots, t_m^i \neq \emptyset \text{ und } t_1^i, \ldots, t_m^i \subseteq P \text{ und } supp_D(T_i) \geq minsupp.$$

*Die Sequenzen T_i werden **häufige Sequenzen** genannt.*

Insbesondere müssen die Sequenzen T_i keine Elemente aus D sein.

Beispiel 4.6. *Gegeben sei die Menge* $D = \{AAA, (AB)(AC), (AB)C\}$ *und* $minsupp = 0,5$. *Die Menge der häufigen Sequenzen* T *besteht aus* $T = \{A, B, C, (AB), AA, AC, BC, (AB)C\}$.

Die allgemeine SPM-Problemstellung besteht im Worst-Case nicht nur aus einem exponentiell wachsenden Suchraum, sondern ebenfalls aus einer exponentiell wachsenden Anzahl an Lösungen.

Beispiel 4.7. *Es seien Produkte* $P = \{p_1, \ldots, p_n\}$, *die Sequenz* $S =< P, \ldots, P >$ *mit* $|S| = m$, *die Menge* $D = \{S\}$ *und* $minsupp = 1$ *gegeben. Die Lösungsmenge* T *der häufigen Sequenzen besteht aus allen Teilsequenzen von* S.
Die Teilsequenzen der Länge 1 *entsprechen allen Teilmengen von* P *ohne die leere Menge. Es gibt demnach* $X = 2^n - 1$ *Teilsequenzen der Länge* 1.
Die Teilsequenzen der Länge 2 *bilden sich als Verkettung aller möglichen Teilsequenzen der Länge 1. Es gibt demnach* X^2 *Teilsequenzen der Länge* 2.
Allgemein gilt damit bis zur Länge m *unter Berücksichtigung der Ausnahme der leeren Sequenz, dass die Lösungsmenge* T *aus*

$$\sum_{k=1}^{m} X^k = \frac{X^{m+1} - 1}{X - 1} - 1 = \frac{(2^n - 1)^{m+1} - 1}{2^n - 2} - 1$$

Sequenzen besteht.

Aus diesem Grund wurden im Laufe der Zeit verschiedene Verfahren zur Lösung des Problems entwickelt, die sich auf unterschiedliche Weise mit einer Beschränkung des Suchraums oder einer Eingrenzung der Lösungsmenge beschäftigen.

4.2.2 Kontrolliertes Wachstum von Sequenzen

Um den Suchraum aller Sequenzen für strukturierte Algorithmen beherrschbar zu machen, wird im Allgemeinen zunächst von einer linearen Ordnung der Grundmenge der Produkte ausgegangen. Diese lineare Ordnung wird für Produktmengen um eine lexikographische Mengenordnung erweitert. Mit der Existenz einer lexikographischen Mengenordnung kann anschließend eine lexikographische Ordnung für endliche Folgen variabler Länge erzeugt werden.

Viele Algorithmen durchschreiten den Suchraum mit einem Suchbaumansatz. Sie beginnen hierbei mit Sequenzen der Länge 1, die nur aus einem Element der Grundmenge bestehen und erweitern diese Sequenzen strukturiert. Hierbei wird auf zwei Erweiterungsmöglichkeiten zurückgegriffen.

Definition 4.8. *Sei die Sequenz* $S =< s_1, \ldots, s_n >$ *gegeben. Die Sequenz* $T =< s_1, \ldots, s_n \cup \{p\} >$ *mit* $p \in P \backslash s_n$ *wird dann eine* ***i-Erweiterung*** *von* S *genannt. Das letzte Folgenglied wird hierbei um ein weiteres Element der Grundproduktmenge ergänzt.*

Definition 4.9. *Sei die Sequenz* $S =< s_1, \ldots, s_n >$ *gegeben. Die Sequenz* $T =< s_1, \ldots, s_n, \{p\} >$ *mit* $p \in P$ *wird dann eine* ***s-Erweiterung*** *von* S *genannt. Die Sequenz wird hierbei um ein neues Folgenglied erweitert, welches aus einem Element der Grundproduktmenge besteht.*

Im Grundansatz können zwei Suchstrategien unterschieden werden.

In Breitensuchverfahren wird der Suchraum jeweils in Schichten von Sequenzen mit gleicher Länge durchsucht. Dieser Ansatz hat den Vorteil, dass eine größere Menge an möglichen Kandidaten generiert werden kann, die dann in einem einzelnen Datenbankscan auf ihren Support überprüft werden. Die ermittelten häufigen Sequenzen können anschließend durch s-Erweiterungen ergänzt werden.

In Tiefensuchverfahren wird hingegen eine einzelne Sequenz solange lexikographisch erweitert, bis sie ihren Status als häufige Sequenz verliert. Erst dann wird in einem Rückschrittverfahren eine neue Erweiterungsstelle für die gleiche Sequenz gesucht. Tiefensuchverfahren sind konzeptionell zunächst teurer, da sie eine Vielzahl von einzelnen Supportüberprüfungen in der Datenbank durchführen müssen. Jedoch erhofft man sich durch das schnellere Wachstum einer Sequenz, frühzeitige Informationen über Teilbäume in anderen Baumbereichen zu erlangen, die nicht weiter durchschritten werden müssen.

4.2.3 Monotonie-Eigenschaft von häufigen Sequenzen

Die Monotonie-Eigenschaft ist eine Aussage über häufige Sequenzen, die analog zur Monotonie-Eigenschaft von häufigen Produktmengen aufgebaut ist. Sie ist Grundlage der Algorithmen Apriori [AS94] und AprioriAll [AS95] und wird deswegen auch häufiger Apriori-Eigenschaft genannt. Viele nachfolgende Algorithmen haben sich an dieser Eigenschaft orientiert.

Theorem 4.10. *Monotonie-Eigenschaft für häufige Sequenzen*
Sei $S_1 =< s_1, \ldots, s_n >$ eine häufige Sequenz einer Transaktionsdatenbank D. Jede nichtleere Teilsequenz $S_2 \subseteq S_1$ ist dann ebenfalls eine häufige Sequenz in D.

Beweis. Wenn S_1 eine häufige Sequenz von D ist, dann gilt

$$supp_D(S_1) \geq minsupp$$

und damit

$$\frac{abssupp_D(S_1)}{|D|} = \frac{|U_{S_1}|}{|D|} \geq minsupp.$$

Mit der Überdeckung $U_{S_1} = \{T \in D : S_1 \subseteq T\}$ und $S_2 \subseteq S_1$, gilt für alle $T \in U_{S_1}$: $S_2 \subseteq S_1 \subseteq T$.
Somit ist $|U_{S_2}| \geq |U_{S_1}|$ und damit

$$supp_D(S_2) \geq supp_D(S_1) \geq minsupp.$$

S_2 ist also eine häufige Sequenz von D. □

Die aussagenlogische Negation des Lemmas besagt, dass eine Sequenz nicht häufig ist, wenn mindestens eine Teilsequenz von ihr existiert, die nicht häufig ist.

Diese Eigenschaft wird von vielen Algorithmen genutzt, um Lösungen des SPM-Problems zu konstruieren. Hierbei beginnen die Verfahren mit der Bestimmung von häufigen Sequenzen der Länge 1 und erstellen aus den Lösungen per paarweiser Verbindung potentielle Kandidaten für häufige Sequenzen der Länge 2. Diese Kandidaten werden in einem Datenbankscan bezüglich ihrer Häufigkeit kontrolliert und nichthäufige Sequenzen aus der Menge entfernt. Dieser Prozess wird iterativ durchgeführt, bis keine häufigen Sequenzen mehr erzeugt werden können.

4.2.4 Horizontale und vertikale Datenbankrepräsentationen

Im Allgemeinen liegt eine Datenbank in einer horizontalen Repräsentation vor. In dieser Art der Repräsentation werden Sequenzen in einer linearen Liste geordnet. Diese Anordnung ergibt sich meist auf natürliche Weise, wenn in der Praxis Daten zusammengestellt werden, aus denen Muster geschürft werden sollen. Im Fokus einer horizontalen Repräsentation befinden sich schließlich die Objekte, an deren Musterwissen man interessiert ist.

Sequenz-Datenbank	
SID	Sequenz
1	< (AB)C >
2	< BC >
3	< (CA) >

Tabelle 4.1: Eine horizontale Datenbankrepräsentation anhand eines Beispiels (SID = Sequenzschlüssel)

In vertikalen Datenbankrepräsentationen werden, im Kontrast zu horizontalen Datenbankrepräsentationen, nicht mehr Sequenzen von Produktmengen aus einer Grundmenge an Produkten erfasst. Stattdessen wird für jedes Produkt eine Liste von Indizes gepflegt, in der nachgeschlagen werden kann, in welcher Sequenz und welcher Produktmenge innerhalb dieser Sequenz das Produkt vorkommt.

Vertikale Datenbankrepräsentationen eignen sich besonders für Tiefensuchverfahren, da Tiefensuchverfahren aus ihrer Natur heraus keine großflächigen Kandidatenmengen nach ihrem Support scannen sondern einzelne Sequenzen. In diesem Fall ist es hilfreich eine Datenbankrepräsentation zu wählen, in der in konstanter Zeit (aus Sicht der Datenbankgröße) der Support einer Sequenz ermittelt werden kann.

Sequenz-Datenbank			
Produkt	SID/PID	SID/PID	SID/PID
A	1/1	3/1	
B	1/1	2/1	
C	1/2	2/2	3/1

Tabelle 4.2: Die entsprechende vertikale Datenbankrepräsentation (PID = Produktmengenschlüssel)

4.3 Geschlossene häufige Sequenzen

Wie bereits im Falle der Mustererkennung in Produktmengen wächst der Raum der häufigen Sequenzen mit wachsender Maximalsequenzlänge im schlechtesten Fall exponentiell stark, sodass die bloße Auflistung der Lösungsmenge bereits zu einem ineffizienten Verfahren wird. Um diese Probleme zu umgehen, wurden über die Zeit mögliche Alternativen zur Bestimmung aller häufigen Sequenzen entwickelt.

Geschlossene Sequenzsuchen konzentrieren sich darauf, nur einen Teil der Lösungsmenge des SPM-Problems zu berechnen. Aus dieser Teilmenge lassen sich jedoch alle häufigen Sequenzen samt ihrem Support verlustfrei rekonstruieren. Hierzu wird folgende Definition verwendet.

Definition 4.11. *Gegeben sei eine Datenbank D und die Menge der häufigen Sequenzen I von D. Die Menge der geschlossenen häufigen Sequenzen C ist*

$$C = \{A \in I : \nexists\, B \in I \textit{ mit } A \subset B \text{ und } supp_d(A) = supp_d(B)\}$$

Da geschlossene häufige Sequenzen ebenso häufig sind, ist die Menge der geschlossenen häufigen Sequenzen höchstens genauso groß wie die Menge der häufigen Sequenzen. In der Praxis kann mit der Konstruktion von geschlossenen häufigen Sequenzen jedoch oft eine große Speicherplatz- und Laufzeitersparnis gewonnen werden, insbesondere wenn die Maximallänge von häufigen Sequenzen als besonders groß geschätzt werden kann.

Bekannte Algorithmen zur Bestimmung von geschlossenen häufigen Sequenzen sind beispielsweise CloSpan [YHA], BIDE [WH04] und M-bBIDE [HWY13].

4.4 Maximale häufige Sequenzen

In manchen Fällen ist selbst die Menge der geschlossenen häufigen Sequenzen zu groß, um sie auszugeben oder aus ihr ohne Weiterverarbeitung sinnvolle Erkenntnisse zu gewinnen. In diesen Fällen kann es hilfreich sein, nur die Menge der maximalen häufigen Sequenzen bestimmen zu lassen.

Definition 4.12. *Gegeben sei eine Datenbank D und die Menge der häufigen Sequenzen I von D. Die Menge der maximalen häufigen Sequenzen M ist*

$$M = \{A \in I : \nexists\, B \in I \text{ mit } A \subset B\}.$$

Die Menge der maximalen häufigen Sequenzen ist immer Teilmenge der Menge der geschlossenen häufigen Sequenzen, die wiederum immer Teilmenge der Menge der häufigen Sequenzen ist. Diese Beziehung ergibt sich sofort aus den Definitionen.

Aus der Menge der maximalen häufigen Sequenzen lässt sich jederzeit die Menge der häufigen Sequenzen zurückgewinnen. Allerdings gilt dies nicht für Informationen über den Support einzelner häufiger Sequenzen. Diese gehen bei der Konstruktion der maximalen häufigen Sequenzen verloren. Wenn die Information über den Support einer Menge notwendig ist, so muss sie entweder durch einen Datenbankscan zurückgewonnen werden oder es muss auf eine geschlossene Sequenzsuche zurückgegriffen werden.

Bekannte Algorithmen zur Bestimmungen von maximalen häufigen Sequenzen sind beispielsweise MaxSP [FVWT13] oder VMSP [FVWGT14].

Beispiel 4.13. *Gegeben sei die Menge* $D = \{AAA, (AB)(AC), (AB)C\}$ *und* $minsupp = 0{,}5$. *Die Menge der häufigen Sequenzen besteht aus* $T = \{A, B, C, (AB), AA, AC, BC, (AB)C\}$. *Die Menge der maximalen häufigen Sequenzen ist* $M = \{AA, (AB)C\}$.

4.5 Generatoren

Generatoren sind eine alternative Beschreibungsart für die Menge der häufigen Sequenzen. Während bei geschlossenen Sequenzen maximal lange Sequenzen mit gleichem Support gesucht werden, beschäftigt man sich bei der Suche nach Generatoren mit minimal langen Sequenzen mit gleichem Support.

Definition 4.14. *Gegeben sei eine Produktdatenbank D. Eine Sequenz G heißt* ***Generator****, wenn gilt:* $\nexists\, A \subset G$ mit $supp_D(A) = supp_D(G)$. *Ist G eine häufige Sequenz, so wird G auch häufiger Generator genannt.*

Generatoren haben die gelegentlich gewünschte Eigenschaft, dass man aus ihnen kontrolliert häufige Sequenzen erzeugen kann, indem man sie erweitert. Dies bedeutet, dass eine selektive Vorauswahl von Generatoren genutzt werden kann, in der bereits alle Generatoren entfernt wurden, deren Muster als nicht wesentlich oder trivial erkannt wurde. Auf diese Weise lässt sich das Wachstum und die Anzahl von häufigen Sequenzen durch bereits bekanntes Fachwissen minimieren.

Generatoren zur Suche nach häufigen Sequenzen werden beispielsweise in [GWHZ08] [YZZ+11] [FVGŠH14] verwendet.

Beispiel 4.15. *Gegeben sei die Menge* $D = \{AAA, (AB)(AC), (AB)C\}$ *und* $minsupp = 0,5$*. Die Menge der häufigen Sequenzen besteht aus*

$$T = \{A : 3, B : 2, C : 2, (AB) : 2, AA : 2, AC : 2, BC : 2, (AB)C : 2\}.$$

Die Menge der Generatoren ist $M = \{A : 3, B : 2, C : 2, AA : 2\}$.

4.6 Top-k-häufige Sequenzen

Die Suche nach Top-k-häufigen Sequenzen [TYH05] [FVGG+13] ergab sich wie bereits im Falle von Produktmengen durch den Wunsch, einen Kontrollmechanismus für die Anzahl der Lösungen zu besitzen. In der Praxis ist man oft nicht daran interessiert, sämtliche Lösungen des SPM-Problems zu generieren sondern nur diejenigen, die besonders interessant sind, also besonders oft in der Datenbank vorkommen. Da man allerdings nicht im Vorfeld abschätzen kann, wie hoch hierzu der Mindestsupport sein muss, läuft man in das Problem, die gängigen Algorithmen mehrmals mit unterschiedlichem Minsupp-Wert durchlaufen lassen zu müssen.

Der Ansatz der Suche nach Top-k-häufigen Sequenzen versucht dem Problem entgegen zu steuern, indem statt des Minsupp-Parameters die Anzahl der gewünschten Lösungen mit maximalem Support vorgegeben wird. Die entsprechenden Algorithmen versuchen dann intern, den MinSupp-Wert dynamisch zu erhöhen, sodass große Bereiche des Suchraumes frühzeitig ausgeschlossen werden können.

Definition 4.16. *Eine Sequenz S heißt* ***top-k-häufig****, wenn nicht mehr als* $k-1$ *Sequenzen existieren, deren Support größer ist als der Support von S.*

Die Definition impliziert, dass tatsächlich mehr als k k-häufige Sequenzen existieren können, sofern mehrere Sequenzen den identischen minimalen Support aufweisen. Analog heißt eine maximale (geschlossene) häufige Sequenz top-k-maximal-häufig (top-k-geschlossen-häufig), wenn nicht mehr als $k-1$ maximale (geschlossene) Sequenzen existieren, deren Support größer ist.

Theorem 4.17. *Gegeben sei eine Datenbank D und eine beliebige Menge an Sequenzen* $S = \{S_1 \ldots, S_k\}$*. Sei*

$$minsupp_D(S) = \min_{S_i \in S} supp_D(S_i).$$

Dann ist sichergestellt, dass für alle top-k-häufigen Sequenzen I von D gilt:

$$supp_D(I) \geq minsupp_D(S).$$

Beweis. Der Beweis folgt sofort aus der Definition. Angenommen, es gäbe eine top-k-häufige Sequenz I von D mit $supp_D(I) < minsupp_D(S)$. Dann hätten alle k Mengen in S einen höheren Support als I. Dies steht im direkten Widerspruch

zur Top-k-Häufigkeit von I. Folglich gibt für alle top-k-häufigen Produktmengen I von D

$$supp_D(I) \geq minsupp_D(S).$$

□

Theorem 4.17 ermöglicht ein Kriterium, um Teile des Suchraumes frühzeitig abzuschneiden. Hiervon machen beispielsweise der TSP-Algorithmus [TYH05] und der TKS-Algorithmus [FVGG+13] Gebrauch.

4.7 Ausgewählte Verfahren

4.7.1 Der BIDE(+)-Algorithmus

Der BIDE-Algorithmus [WH04] ist ein Algorithmus zur Bestimmung aller geschlossenen Sequenzen innerhalb einer Datenbank. BIDE steht für "BI-Directional Extension" und bezeichnet ein damals neuartiges Verfahren zur Verifikation, ob es sich bei einer häufigen Sequenz um eine geschlossene Sequenz handelt.

Ursprünglich war der BIDE-Algorithmus für sequenzielle Datenbanken konzipiert, in denen nur einelementige Folgenglieder vorkommen. In diesem speziellen Fall kann gefolgert werden, dass eine häufige Sequenz S genau dann geschlossen ist, wenn

1. keine Superfolge mit gleichen Support existiert, die S als Präfixfolge nutzt,
2. keine Superfolge mit gleichem Support existiert, die S als Suffixfolge nutzt und
3. keine Superfolge mit gleichem Support existiert, die S um Zwischenglieder erweitert.

Aus dieser Eigenschaft lässt sich ein bidirektionales Kriterium zur Verifizierung der Geschlossenheit konstruieren, welches im BIDE-Algorithmus genutzt wird. Die Autoren weisen darauf hin, dass sich diese Eigenschaft auf Datenbanken verallgemeinern lässt, in denen beliebige Produktmengen als Folgenglieder erlaubt sind. In diesem Fall muss noch das zusätzliche Kriterium überprüft werden, dass keine Superfolge mit gleichen Support existiert, die ein Folgenglied um ein weiteres Produkt erweitert. Hierzu wurde später ein "BIDE+"-Algorithmus veröffentlicht.

Bewertung des Algorithmus

Der BIDE-Algorithmus generiert häufige geschlossene Sequenzen, ohne dabei auf eine Kandidatengenerierung zurückgreifen zu müssen oder einen Zwischenspeicher aller bisher gefundenen geschlossenen Sequenzen kleinerer Länge anlegen zu müssen. Diese Eigenschaft lässt den Algorithmus insbesondere in

Datenbanken, in denen häufige Sequenzen mit großer Länge erwartet werden, vergleichsweise effizient und mit einem überschaubaren Speicherplatzverbrauch arbeiten.

Wie bei allen Algorithmen seiner Art kämpft er allerdings mit dem Umstand, dass die Lösungsmenge der häufigen geschlossenen Sequenzen exponentiell wächst.

4.7.2 Der VSMP-Algorithmus

Der VSMP-Algorithmus [FVWGT14] ist ein Algorithmus zum Schürfen aller häufigen maximalen Sequenzen in einer Datenbank. VSMP steht hierbei für "Vertical Mining of Maximal Sequential Patterns". Der Algorithmus nutzt zur Suche nach maximalen Sequenzen ein Tiefensuchverfahren durch i-Erweiterungen und s-Erweiterungen in einem Suchbaum. Er verwendet, wie bereits der MAFIA-Algorithmus, für die Suche nach maximalen häufigen Sequenzen eine vertikale Datenbankrepräsentation.

Bewertung des Algorithmus

Der VSMP-Algorithmus ist ein relativ moderner Algorithmus für die Suche nach maximalen häufigen Sequenzen in einer Datenbank. Er kombiniert dabei viele Ansätze und Kriterien, die einzeln in bekannten Algorithmen für die Mustererkennung in Produktmengen verwendet wurden. Durch die Beschränkung auf maximale Sequenzen und ein Tiefensuchverfahren mit vertikaler Datenbankrepräsentation kann ein erheblicher Performancegewinn erreicht werden.

4.7.3 Der VGEN-Algorithmus

Der VGEN-Algorithmus ist ein Algorithmus zum Schürfen von häufigen Generatoren in Sequenz-Datenbanken. VGEN steht für "Fast Vertical Mining Sequential Generator Patterns" und beschreibt die grundlegende Strategie der Suche über eine vertikale Datenbankrepräsentation.

Der VGEN-Algorithmus ist ähnlich wie der VSMP-Algorithmus aufgebaut. Er nutzt ein Tiefensuchverfahren für Sequenzen, welches er anhand von i-Erweiterungen und s-Erweiterungen durchläuft. Nach jedem Erweiterungsschritt prüft er anhand von Kriterien, ob der entsprechende Teilsuchbaum weiter durchlaufen werden soll.

Bewertung des Algorithmus

Der VGEN-Algorithmus ist ein relativ moderner Algorithmus für die Suche nach häufigen Generatoren in einer Datenbank. Wie bereits der VSMP-Algorithmus kombiniert er dabei Ansätze und Kriterien aus bereits bekannten Verfahren. Durch die Beschränkung auf Generatoren und ein Tiefensuchverfahren mit vertikaler Datenbankrepräsentation kann ein Performancegewinn erreicht werden,

wenn das Ergebnis auch in der Regel mehr Ausgaben konstruiert als eine Suche nach maximalen Sequenzen. Im Gegensatz zur Suche nach maximalen häufigen Sequenzen ist es durch Generatoren jedoch möglich, die Nachkonstruktion aller häufigen Sequenzen zu beeinflussen und ungeeignete Generatoren frühzeitig zu entfernen.

4.7.4 Der TKS-Algorithmus

Der TKS-Algorithmus [FVGG+13] ist ein Verfahren zum Schürfen von top-k-häufigen Sequenzen. TKS steht hierbei für "Top-K Sequential Pattern Mining". Der Benutzer gibt einen Wert k für die Anzahl an häufigen Sequenzen mit höchstem Support vor, die als Ergebnis des Algorithmus ausgegeben werden sollen.

Der Algorithmus verwenden im Kern eine vertikale Datenbankrepräsentation mit Hilfe von Bit-Vektoren. Darauf aufbauend wird anhand eines Tiefensuchverfahrens und der Durchschreitung des Suchraumes mit i-Erweiterungen und s-Erweiterungen nach allen häufigen Sequenzen in der Datenbank gesucht.

Bewertung des Algorithmus

Der TSK-Algorithmus ist in der Lage die top-k-häufigen Sequenzen aus einer Datenbank zu generieren und nutzt hierzu eine Vielzahl von modernen Methoden. Er ist dabei wesentlich effizienter als der bis dahin aktuelle TSP-Algorithmus [TYH05].

Problematisch ist die Einschränkung, dass keine minimale Länge der gesuchten Sequenzen oder sonstige weitere Anforderungen an die Sequenzen vorgegeben werden können. Dies führt aufgrund der Monotonie-Eigenschaft von Sequenzen dazu, dass unter den top-k-häufigsten Sequenzen auch alle Teilsequenzen einer top-k-häufigen Sequenz fallen. Der Parameter k muss deswegen entweder besonders hoch gewählt werden oder er wird vor allem einelementige Sequenzmengen ausgegeben, deren praktischer Nutzen fraglich bleibt.

Kapitel 5

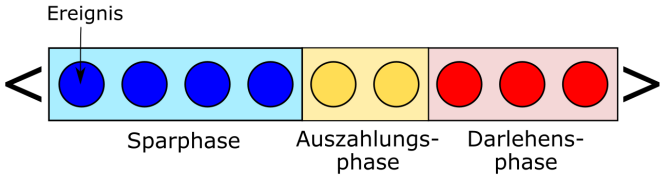
Mustererkennung in sequenziellen Phasendatenbanken

5.1 Motivation

In diesem Kapitel wird ein neues Datenbankmodell für die Suche nach häufigen Sequenzen aufgestellt, welches auf Phasen basiert. Dieses Modell wurde im Rahmen der Mustererkennung in Sequenzen nach bestem Wissen des Autors bisher in der Literatur nicht betrachtet. Es wird eine formale Definition vorgenommen und die besonderen Eigenschaften dieses Modells herausgearbeitet. Auf den Ergebnissen werden daraufhin mehrere neuwertige Algorithmen aufgebaut. Die Verfahren werden abschließend in einer Testreihe mit den gängigen und schnellsten Verfahren für die allgemeine Problemstellung verglichen und dabei Vor- und Nachteile des Ansatzes dargelegt.

5.1.1 Bausparen als Phasenmodell

Das in diesem Kapitel vorgestellte Phasenmodell ist durch ein konkretes Anwendungsgebiet motiviert. Im Rahmen des Bausparens kann ein Bausparvertrag als eine sequentielle Liste von Ereignissen verstanden werden, die nacheinander mehrere Phasen durchläuft. Auf die Details des Bausparens und die Modellierung des Bausparens als Phasenmodell wird im Praxisteil dieser Arbeit detailliert eingegangen.



Ein Bausparvertrag als lineare Liste von Ereignissen

Zu Beginn durchläuft der Bausparvertrag eine Sparphase, in der der Bausparer regelmäßige Sparzahlungen leistet und auf spezielle Optionen wie eine Erhöhung der Bausparsumme oder die Kündigung des Vertrages zurückgreifen

kann. Im Anschluss ergeben sich weitere Phasen, die teilweise übersprungen werden können. Zu jedem Zeitpunkt hat der Bausparer die Möglichkeit, den Bausparvertrag durch Kündigung oder Rückzahlung des Darlehens zu beenden.

Für jeden einzelnen Bausparvertrag sind die Anzahl und die Länge der durchlaufenen Phasen variabel. Der Vertrag kann alle Phasen von Anfang bis Ende durchlaufen, einige Phasen überspringen oder auch nur eine einzige Phase wahrnehmen. Sichergestellt ist jedoch immer, dass keine spätere Phase vor einer früheren Phase durchlaufen werden kann. Es ist nicht möglich, die Darlehensphase zu durchlaufen und anschließend in die Auszahlungssphase zu wechseln. Wenn eine Auszahlungssphase (im Datensichtbarkeitszeitraum) stattgefunden hat, so muss sie zwingend vor der Darlehensphase erfolgt sein. Die Ordnung der Phasen ist also unabhängig vom einzelnen Vertrag fest vorgegeben.

5.1.2 Formalisierung als sequentielles Datenbankmodell

Die in Kapitel 4 behandelte Problemstellung geht davon aus, dass eine allgemeine Menge an Produkten existiert, die in beliebiger Reihenfolge in Sequenzen auftauchen können. Im Rahmen vielen Anwendungsgebiete ist diese Modellierung sinnvoll. Es spricht wenig dafür, dass alle Kunden in einem Supermarkt in einer Woche nur bestimmte Produkte kaufen und in der nächsten Woche diese Produkte vollkommen meiden. Eine Phasenmodellierung wäre nur dann sinnvoll, wenn der Supermarkt sein komplettes Produktsortiment in regelmäßigen Abständen austauschen würde.

In einer Phasenmodellierung sind die Anzahl und die Reihenfolge der Phasen fest vorgegeben, nicht aber, ob und wann sie in einer einzelnen Sequenz auftauchen. Weiterhin steht fest, dass jedes Element der Produktmenge nur in genau einer Phase auftauchen kann. Dies ermöglicht eine klare Partitionierung der Produktmenge in Phasenmengen und kann die Komplexität der Sequenzsuche erheblich verringern.

Es soll nicht unerwähnt bleiben, dass jede sequenzielle Phasendatenbank als allgemeine sequenzielle Datenbank behandelt werden kann, indem man die zusätzliche Phaseninformation ignoriert. Hierdurch kann jedes Suchproblem in einer sequenziellen Phasendatenbank von einem allgemeinen Algorithmus gelöst werden. In diesen Fällen muss jedoch auf die Ausnutzung effizienterer Lösungsstrategien verzichtet werden. Gleichsam lässt sich jede allgemeine sequenzielle Datenbank als sequenzielle Phasendatenbank abbilden, indem die Anzahl der Phasen auf Eins gesetzt wird. Auch in diesem Fall hat man jedoch keine weitere Information gewonnen und spezielle Lösungsalgorithmen werden nicht effizienter sein als der allgemeine Fall.

5.1.3 Abgrenzung

Im Folgenden wird eine Abgrenzung der Modellierung einer sequenziellen Phasendatenbank von anderen in der Literatur gängigen Erweiterungen der Problemstellung aus Kapitel 4 vorgenommen.

Multidimensionale Sequenzdatenbanken

Unter multidimensionalen Sequenzdatenbanken werden sequenzielle Datenbanken verstanden, deren Sequenzen um nichtsequenzielle Informationen erweitert werden [PHP+01] . Dies können beispielsweise Informationen über das Alter oder den Wohnort des Kunden sein. Da es sich bei diesen Informationen nicht um sequenzielle Informationen handelt, sind sie für ein Phasenmodell zunächst scheinbar uninteressant. Allerdings können diese Informationen auf Wunsch als zusätzliche Phase (mit nur einem Zeitpunkt) in ein Phasenmodell eingebaut werden. Genaueres hierzu wird in Abschnitt 6.2 behandelt.

Mehrstufige Sequenzdatenbanken

Mehrstufige Sequenzdatenbanken sind Datenbanken, in denen Sequenzen von Sequenzen definiert werden [CY02] [YC05] [HZBR11]. Die Sequenzen in mehrstufigen Sequenzdatenbanken sind hierarchisch strukturiert. Eine Anwendung liegt beispielsweise in der Modellierung von mehreren parallel existierenden Supermärkten, in denen der gleiche Kunde im Zeitverlauf erfasst werden soll. Hiermit sind örtliche Wechselwirkungen unter den Einkäufen erfassbar. Ebenso ist eine Modellierung als Kunde mit mehreren gleichzeitigen Konten oder Bausparverträgen denkbar, auf die parallel eingezahlt wird. Diese Parallelität kann nicht durch Phasen abgebildet werden und gleichsam findet in Phasen keinerlei Parallelität statt. Zusätzlich werden keine getrennten Produktmengen gebildet und in allen parallelen Sequenzen werden auf die gleichen Produkte oder Ereignisse zurückgegriffen.

Da das Thema dennoch für die Modellierung von mehreren parallel betriebenen Phasendatenbanken interessant sein kann, wird es ausführlicher in Abschnitt 6.3 betrachtet.

Sequenzdatenbanken mit Zeitbeschränkungen

Sequenzdatenbanken mit Zeitbeschränkungen sind Transaktionsdatenbanken, in denen zusätzlich zu den Elementen in einer Transaktion eine Zeitinformation gespeichert wird [SA96] [MCP98] [Zak00b] [LL05] [PHW07] [LHC08]. Bei der Suche nach häufigen Sequenzen werden daraufhin Zeitfenster oder minimale bzw. maximale Toleranzgrenzen für Abstände zwischen Elementen oder die Dauer der Gesamtsequenz vorgegeben. Zusätzliche Zeitinformationen sind in Phasendatenbanken nicht vorhanden. Die zeitlichen Abhängigkeiten in Phasendatenbanken bestehen bereits bei der Grundmenge der Produkte und nicht erst als Toleranzgrenzen bei den zu suchenden Mustern.

5.2 Definition und Problemstellung

Im folgenden Abschnitt wird eine formale Definition eines Modells für sequenzielle Phasendatenbanken aufgestellt. Hierzu wird als Grundlage auf die allgemeinen Definitionen zu sequenziellen Datenbanken aus Kapitel 4 zurückgegriffen.

5.2.1 Grundlegende Definitionen

Gegeben sei eine Grundmenge an Produkten $P = \{p_1, \ldots, p_n\}$ und eine Anzahl $m \in \mathbb{N}$ an Phasen.

Definition 5.1. *Eine Abbildung $\phi : P \longrightarrow \{1, \ldots, m\}$ heißt **Phasenabbildung** und ordnet jedem Element der Menge P eine von m Phasen zu.*

Im Allgemeinen gilt $n >> m$.

Definition 5.2. *Die Menge*

$$P[i] = \{p \in P : \phi(p) = i\}$$

*wird **Phasenmenge** der Phase i genannt.*

Definition 5.3. *Eine Sequenz $S =< s_1, \ldots, s_n >$ heißt **Phasensequenz**, wenn ein $k \in \{1, \ldots, m\}$ existiert, sodass für alle $s_i \in S$ gilt: $s_i \subseteq P[k]$.*

Gegeben sei eine Transaktionsdatenbank $D = \{S_1, \ldots, S_l\}$ an Sequenzen $S_i =< s_1^i, \ldots, s_{i_k}^i >$.

Definition 5.4. *Die Menge aller zur Verfügung stehenden Transaktionssequenzen $D = \{S_1, \ldots, S_l\}$ heißt **sequenzielle Phasendatenbank**, wenn gilt:*

1. *Es existiert eine Phasenabbildung $\phi : P \longrightarrow \{1, \ldots, m\}$,*
2. *$\forall\ S_i =< s_1^i, \ldots, s_{i_k}^i > \in D\ \forall\ a, b \in \{1, \ldots, i_k\}$: Wenn $c \in s_a^i$ und $d \in s_b^i$ und $\phi(c) < \phi(d)$, dann gilt: $a < b$.*

Anschaulich wird durch Bedingung 2 sichergestellt, dass Produkte nur in der Reihenfolge der Phasenmengen in Sequenzen auftauchen können. Insbesondere folgt aus Bedingung 2, dass in einer Produktmenge s_a^i einer Sequenz nur jeweils Sequenzglieder einer einzigen Phase vorhanden sind. Diese Voraussetzung kann durch eine Ersetzung von $a < b$ in $a \leq b$ gelockert werden, allerdings erkauft man sich diese Freiheit durch eine erhöhte Komplexität bei der Konstruktion von Algorithmen.

5.2.2 Häufige Sequenzsuche

Die weiteren Definitionen zur Suche von häufigen Sequenzen in sequenziellen Phasendatenbanken verlaufen analog zum allgemeinen Fall.

Definition 5.5. *Der **absolute Support** $abssupp_D(T)$ einer Sequenz T aus einer sequenziellen Phasendatenbank D ist die Anzahl der Sequenzen S_i in D, für die gilt: T ist Teilsequenz von S_i. Der **relative Support** einer Sequenz T in D ist*

$$supp_D(T) = \frac{abssupp_D(T)}{|D|}.$$

Problemstellung 5.6. *Gegeben sei eine sequenzielle Phasendatenbank $D = \{S_1, \ldots, S_n\}$ und ein Schwellwert $0 \leq minsupp \leq 1$. Finde alle möglichen Sequenzen*

$$T_i =< t^i_1, \ldots, t^i_{i_k} > \text{ mit } t^i_1, \ldots, t^i_{i_k} \subseteq P : supp_D(T_i) \geq minsupp.$$

*Eine Sequenz T_i wird **häufige Sequenz** genannt.*

5.3 Struktur und Eigenschaften des Modells

Im Verlauf dieses Abschnittes werden grundsätzliche Aussagen über die Struktur des Modells und seiner Eigenschaften aufgezeigt und bewiesen, die im weiteren Verlauf Grundlage für die Konstruktion neuer Algorithmen werden.

5.3.1 Phasenstruktur

Aus der Definition einer Phasendatenbank (Definition 5.4) ergeben sich folgende Beobachtungen.

Lemma 5.7. *Sei $D = \{S_1, \ldots, S_l\}$ eine sequenzielle Phasendatenbank mit m Phasen. Dann gilt für jede Transaktionssequenz $S_i =< s^i_1, \ldots, s^i_{i_k} >$ in D:*

1. *S_i muss vorhandene Phasen in der Reihenfolge der Nummerierung durchlaufen,*
2. *S_i durchläuft eine beliebige Anzahl von existierenden Phasen,*
3. *S_i kann eine beliebige Anzahl von Phasen überspringen,*
4. *Für jede Produktmenge s^i_j gilt: alle Elemente aus s^i_j sind aus der gleichen Phase, das heißt*
$$\forall\, s^i_j \in S_i\ \exists n \in \{1, \ldots, m\} : s^i_j \subseteq P[n].$$
5. *alle Produktmengen einer Phase sind zusammenhängend, das heißt*
$$\forall\, s^i_j, s^i_l \in S_i \text{ mit } j < l, s^i_j \subseteq P[n] \text{ und } s^i_l \subseteq P[n] :\ \forall\, o \in \{j, \ldots, l\} : s^i_o \subseteq P[n].$$
6. *Jede Teilsequenz von S_i erfüllt ihrerseits alle Punkte des Lemmas.*

Beweis. Die Aussagen des Lemmas werden einzeln bewiesen.

1. Dies ist eine direkte Folgerung aus Definition 5.4 Bedingung 2.

2. Gilt trivialerweise, da keine maximale Begrenzung der Anzahl an Phasen in Definition 5.4 vorgenommen wird.

3. Gilt trivialerweise, da keine minimale Begrenzung der Anzahl an Phasen in Definition 5.4 vorgenommen wird.

4. Angenommen es existiere eine Produktmenge s_j^i der Sequenz s_i und zwei Elemente $c, d \in s_j^i$ mit $c \in P[n_1]$ und $d \in P[n_2]$ mit $n_1 \neq n_2$. Dann gilt entweder $\phi(c) < \phi(d)$ oder $\phi(d) < \phi(c)$. Nach Bedingung 2 aus Definition 5.4 wäre damit in beiden Fällen $i_j < i_j$. Dies ist ein Widerspruch und damit gilt die Aussage des Lemmas.

5. Angenommen es existierten zwei Produktmengen

$$s_j^i, s_l^i \in S_i \text{ mit } j < l, s_j^i \subseteq P[n_1] \text{ und } s_l^i \subseteq P[n_1]$$

und ein

$$o \in \{j, \ldots, l\} : s_o^i \not\subseteq P[n_1].$$

Aufgrund des letzten Beweispunktes folgt aus $s_o^i \not\subseteq P[n_1]$ zwingend

$$\exists\, n_2 \in \{1, \ldots, m\} \text{ mit } n_2 \neq n_1 : s_o^i \subseteq P[n_2].$$

Seien weiterhin $c \in s_j^i$, $d \in s_o^i$ und $e \in s_l^i$. Ist $n_1 < n_2$ dann gilt nach Bedingung 2 aus Definition 5.4 $j < o$ und $l < o$. Im Falle von $n_2 < n_1$ gilt $o < j$ und $o < l$. Beides führt zum Widerspruch, da nach Annahme $o \in \{i, \ldots, j\}$. Folglich ist die Annahme falsch und es gilt die Behauptung des Lemmas.

6. Es wird gezeigt, dass jede Teilsequenz von S_i die Bedingung 2 aus Definition 5.4 erfüllt. Sei $S_j =< a_1, \ldots, a_n >$ eine Teilsequenz von $S_i =< b_1, \ldots, b_k >$, dann gilt nach Definition einer Teilsequenz (4.2): es existieren natürliche Zahlen $1 \leq o_1 < o_2 < \ldots < o_n \leq k$, so dass gilt $a_1 \subseteq b_{o_1} \wedge a_2 \subseteq b_{o_2} \wedge \ldots \wedge a_n \subseteq b_{o_n}$. Weiterhin ist S_i eine Sequenz aus einer Phasendatenbank. Damit ist gegeben :

$$\forall\, c_1, c_2 \in \{1, \ldots, k\} :$$

$$\text{Wenn } d_1 \in b_{c_1} \text{ und } d_2 \in b_{c_2} \text{ und } \phi(d_1) < \phi(d_2), \text{ dann gilt } : c_1 < c_2.$$

Da alle Elemente einer Produktmenge aus der gleichen Phase stammen, lässt sich die Bedingung umschreiben in

$$\forall\, c_1, c_2 \in \{1, \ldots, k\} :$$

$$\text{Wenn } b_{c_1} \subseteq P[e_1], b_{c_2} \subseteq P[e_2] \text{ und } e_1 < e_2, \text{ dann gilt } : c_1 < c_2.$$

Insbesondere gilt damit auch

$$\forall\, c_1, c_2 \in \{o_1, \ldots, o_n\} :$$

$$\text{Wenn } b_{c_1} \subseteq P[e_1], b_{c_2} \subseteq P[e_2] \text{ und } e_1 < e_2, \text{ dann gilt } : c_1 < c_2$$

und somit aufgrund der Teilsequenzdefinition

$$\forall\, c_1, c_2 \in \{1, \ldots, n\} :$$

$$\text{Wenn } a_{c_1} \subseteq P[e_1], a_{c_2} \subseteq P[e_2] \text{ und } e_1 < e_2, \text{ dann gilt } : c_1 < c_2.$$

Hiermit wurde gezeigt, dass S_j ebenfalls Bedingung 2 aus Lemma 5.4 erfüllt und damit alle Charakteristika einer Sequenz aus einer Phasendatenbank erbt.

□

Die Tatsache, dass Phasen in Sequenzen einer sequenziellen Phasendatenbank zusammenhängend sind und für jede Produktmenge einer Sequenz eindeutig bestimmt ist, zu welcher Phase sie gehört, ermöglicht die Definition von Phasenteilsequenzen und ihre Verkettung.

Definition 5.8. *Die **Phasenteilsequenz** einer Sequenz S der Phase i ist die Teilsequenz von S, die alle Produktmengen der Phase i beinhaltet. Sie wird mit $S[i]$ beschrieben.*

Definition 5.9. *Zwei Sequenzen $S =< s_1, \ldots, s_n >$ und $T =< t_1, \ldots, t_l >$ werden zu einer Sequenz U **verkettet**, wenn $U =< s_1, \ldots, s_n, t_1, \ldots, t_l >$ gilt. Man schreibt $U = S \circ T$.*

Lemma 5.10. *Jede Sequenz S in einer sequenziellen Phasendatenbank mit m Phasen lässt sich eindeutig in hintereinander gekettete Phasenteilsequenzen zerlegen. Dabei gilt*

$$S = S[1] \circ \ldots \circ S[m].$$

Beweis. Aus Lemma 5.7 ist bekannt, dass alle Elemente innerhalb der gleichen Produktmenge der gleichen Phase angehören. Weiterhin sind alle Produktmengen einer Phase zusammenhängend. Dies bedeutet, dass Trennungen zwischen den Phasen ausschließlich zwischen unterschiedlichen Produktmengen stattfinden können und es höchstens $m - 1$ Trennungspunkte gibt. Zusätzlich treten die vorhandenen Phasen in der Reihenfolge der Phasenordnung auf, wobei eine beliebige Anzahl von Phasen übersprungen werden kann. Mit diesem Wissen ist es möglich, einen Trennungsalgorithmus zu entwerfen, der in linearer Zeit (bezüglich der Länge der Sequenz) eine Zerlegung durchführt.

Data: Sequenz $S =< s_1, \ldots, s_l >$, Phasenmengen $P[1], \ldots, P[m]$
Result: Phasensequenzen $S[1], \ldots, S[m]$
$index = 1$;
forall $i \in \{1, \ldots, m\}$ **do**
  $S[i] =<>$;
  while $index <= l$ *und* $s_{index} \subseteq P[i]$ **do**
    $S[i] = S[i] \circ < s_{index} >$;
    $index = index + 1$;
  end
end

Algorithm 2: Algorithmus zur Konstruktion aller Phasenteilsequenzen

Nachdem die Existenz einer solchen Zerlegung als konstruktiver Algorithmus gezeigt wurde, folgt die Eindeutigkeit direkt aus der Definition einer Phasenteilsequenz. Angenommen es gäbe zwei unterschiedliche Zerlegungen einer Sequenz in Phasenteilsequenzen. Dann gibt es mindestens eine Phase, die zu zwei unterschiedlichen Phasenteilsequenzen führt. Da eine Phasenteilsequenz alle Produktmengen der entsprechenden Phase beinhaltet, können sich die unterschiedlichen Phasenteilsequenzen nicht durch einzelne Elemente innerhalb einer Produktmenge unterscheiden, denn alle Elemente innerhalb einer Produktmenge gehören zur gleichen Phase. Sie müssen sich also in der Anzahl oder der Reihenfolge der Produktmengen unterscheiden. Dies ist jedoch nach Definition einer Phasenteilsequenz ebenso ausgeschlossen, da eine Phasenteilsequenz alle Produktmengen umfasst, die zur entsprechenden Phase gehören, und weiterhin eine Teilsequenz ist, weswegen die Reihenfolge der Produktmengen in der Phasenteilsequenz der Reihenfolge der Produktmengen in der Gesamtsequenz entsprechen muss. Folglich können keine zwei unterschiedlichen Phasenteilsequenzen einer Sequenz existieren und die Zerlegung ist eindeutig. □

Phasenteilsequenzen können die gesamte Sequenz umfassen oder auch leer sein.

Beispiel 5.11. *Gegeben sei eine Sequenz $S = AB(AB)CD$ und die Phasenmengen $P[1] = \{A, B\}$ und $P[2] = \{C, D\}$. Dann ist $S[1] = AB(AB)$ und $S[2] = CD$. Es ist $S = S[1] \circ S[2]$.*

5.3.2 Häufige Phasensequenzen

Sei D eine sequenzielle Phasendatenbank mit m Phasen.

Definition 5.12. *Sei eine Phase j fixiert. Die **Phasenteildatenbank** $D[j]$ von D ist die Menge aller Phasenteilsequenzen $S_i[j]$ mit S_i aus D.*

Insbesondere hat die Phasenteildatenbank $D[j]$ immer genauso viele Elemente wie die Ursprungsdatenbank D. Sollte eine Phase in einer Sequenz aus D nicht belegt sein, so wird als Phasenteilsequenz eine leere Sequenz erfasst. Aus praktischen Gründen behält man zusätzlich die interne Indizierung von D bei, um Sequenzen leichter miteinander vergleichen zu können.

Es folgt nun das Haupttheorem des Kapitels, das eine effiziente Zerlegung des Suchraumes ermöglicht und eine Grundlage für spezialisierte Algorithmen schafft, die im weiteren Verlauf des Kapitels weiter ausgearbeitet werden.

Theorem 5.13. *Sei D eine sequenzielle Phasendatenbank. Ist eine Sequenz S eine häufige Sequenz in D, so ist jede Phasenteilsequenz $S[i]$ von S eine häufige Sequenz in der Phasenteildatenbank $D[i]$.*

Beweis. Aufgrund der Vorarbeit aus Definitionen 5.12 und 5.8 sowie Lemmata 5.7 und 5.10 ist der Beweis des Satzes übersichtlich.

Ist eine Sequenz S_j häufig in D, dann sind wegen der Monotonie-Eigenschaft

von Sequenzen (Theorem 4.10) alle Teilsequenzen von S_j ebenso häufig in D. Da sich S_j in hintereinander gekettete Phasenteilsequenzen zerlegen lässt (Lemma 5.10), die insbesondere Teilsequenzen sind, ist auch jede Phasenteilsequenz $S_j[i]$ von S häufig in D. Es bleibt zu zeigen, dass aus der Häufigkeit der $S_j[i]$ in D ebenso die Häufigkeit der $S_j[i]$ in $D[i]$ folgt.

Wenn $S_j[i]$ häufig in D ist, so liegt ihr absoluter Support *abssupp* in D über dem vorgegebenen Minimalsupport *minsupp*. Dies bedeutet, dass *abssupp* Transaktionssequenzen T_k in D existieren, für die gilt $S_j[i] \subseteq T_k$. Eine Phasenteildatenbank $D[i]$ besteht aus allen Phasenteilsequenzen $T_k[i]$ aus D (Definition 5.12). Wenn gezeigt wird, dass aus $S_j[i] \subseteq T_k$ folgt $S_j[i] \subseteq T_k[i]$, so bleibt der absolute Support erhalten und damit auch der relative Support, da $|D[I]| = |D|$. In diesem Fall wurde gezeigt, dass aus der Häufigkeit der $S_j[i]$ in D ebenso die Häufigkeit der $S_j[i]$ in $D[i]$ folgt und der Beweis ist vollständig.

Aus $S_j[i] =< s_1^j, \ldots s_l^j >$, $T_k =< t_1^k, \ldots, t_n^k >$ und $S_j[i] \subseteq T_k$ folgt per Definition: es existieren natürliche Zahlen $1 \leq o_1 < o_2 < \ldots < o_l \leq n$, so dass gilt

$$s_1^j \subseteq t_{o_1}^k \wedge s_2^j \subseteq t_{o_2}^k \wedge \ldots \wedge s_l^j \subseteq t_{o_l}^k.$$

$T_k[i]$ besteht aus allen Produktmengen von T_k der Phase i. Da $T_k[i] =< t_1^{k[i]}, \ldots, t_r^{k[i]} >$ eine Teilsequenz von T_k ist, bleibt ebenso die Ordnung unter den Sequenzgliedern erhalten. Das bedeutet, es existieren natürliche Zahlen $1 \leq p_1 < p_2 < \ldots < p_r \leq n$, so dass gilt

$$t_1^{k[i]} = t_{p_1}^k \wedge t_2^{k[i]} = t_{p_2}^k \wedge \ldots \wedge t_r^{k[i]} = t_{p_r}^k.$$

Da $S_j[i]$ aus den Produktmengen der Phase i besteht, sind alle $t_{o_1}^k$ bis $t_{o_l}^k$ auch in $T_k[i]$ enthalten. Aufgrund der Ordnung der Indizes p_1 bis p_r, lassen sich deswegen Indizes $1 \leq q_1 < q_2 < \ldots < q_l \leq r$ finden, so dass gilt

$$s_1^j \subseteq t_{q_1}^{k[i]} = t_{o_1}^k \wedge s_2^j \subseteq t_{q_2}^{k[i]} = t_{o_2}^k \wedge \ldots \wedge s_l^j \subseteq t_{q_l}^{k[i]} = t_{o_l}^k.$$

Damit ist gezeigt, dass $S_j[i] \subseteq T_k[i]$ gilt. Somit ist die Häufigkeit von $S_j[i]$ in $D[i]$ gesichert und die Aussagen des Theorems bewiesen. □

Die Umkehrung des Theorems 5.13 gilt nicht. Aus der Existenz von häufigen Phasenteilsequenzen in den Phasenteildatenbanken $D[i]$ lässt sich nicht schließen, dass die Hintereinanderverkettung dieser Phasenteilsequenzen zu einer häufigen Sequenz in D führt. Möglicherweise sinkt der Support der verketteten Sequenz unter den Mindestsupport, da in den Phasenteildatenbanken unterschiedliche Transaktionssequenzen zum Support der Phasenteilsequenzen beitragen.

Aus Theorem 5.13 ergibt sich folgende direkte Beobachtung.

Folgerung 5.14. *Ist eine Phasenteilsequenz $S[i]$ von S nicht häufig in $D[i]$, so ist sie in keiner häufigen Sequenz von D enthalten.*

Dieses Negativkriterium für häufige Sequenzen ermöglicht einen Divide-And-Conquer-Ansatz zur Ermittlung von potentiellen Kandidaten für häufige Sequenzen. Es wird im folgenden Abschnitt bei der Konstruktion der Algorithmen PhaseScan und PhaseTreeScan wieder aufgenommen.

5.4 Konstruktion von neuen Algorithmen

In diesem Abschnitt werden neue Algorithmen hergeleitet und vorgestellt, die sich die spezielle Modellierung und Eigenschaften von sequenziellen Phasendatenbanken zunutze machen, um das Problem der Suche nach allen häufigen Sequenzen in einer sequenziellen Phasendatenbank zu lösen.

5.4.1 PhaseAprioriAll

Der PhaseAprioriAll-Algorithmus ist eine Modifikation des klassischen AprioriAll-Algorithmus, um die zusätzlichen Informationen einer sequenziellen Phasendatenbank auszunutzen. Er gehört somit zur Familie der Algorithmen, die zur Lösung des Problems auf die Generierung von Kandidatenmengen zurückgreifen. Aus diesem Grund leidet er wie alle Algorithmen seiner Art häufig unter einer zu schnell wachsenden Anzahl an Kandidaten. Er soll an dieser Stelle jedoch verdeutlichen, dass auch Apriori-Algorithmen von den zusätzlichen Informationen profitieren können. In der Praxis wird jedoch empfohlen, auf die weiteren in diesem Kapitel vorgestellten neuen Algorithmen auszuweichen, die strukturiertere Konstruktionsverfahren nutzen.

Der allgemeine AprioriAll-Algorithmus

Der AprioriAll-Algorithmus [AS95] war einer der ersten Algorithmen zur Bestimmung von häufigen Sequenzen, der die Monotonie-Eigenschaften von Sequenzen nutzte. Wie bereits sein Vorgänger (siehe Abschnitt 3.7.1), der auf Produktmengen arbeitete, besteht der Apriori-All-Algorithmus aus zwei Phasen, die gemeinsam iterativ aufgerufen werden.

In einem ersten Schritt werden mögliche Kandidatensequenzen für häufige Sequenzen ermittelt. Diese Kandidaten erfüllen alle die Monotonie-Eigenschaft für Sequenzen, welche eine notwendige Bedingung für die Existenz einer häufigen Sequenz darstellt. Dies wird dadurch garantiert, dass der Algorithmus die Kandidatenmenge aus den häufigen Sequenzen kleinerer Sequenzlänge durch Verkettung erzeugt.

Im Falle von Sequenzen der Länge 1 wird mit einem anderen Verfahren gestartet. In diesem Fall findet eine Suche nach allen häufigen Produktmengen innerhalb der Transaktionsdatenbank statt. Sequenzen werden dabei als Mengensystemen von Produktmengen betrachtet. Dieses Problem kann durch eines der Verfahren aus Kapitel 3 gelöst werden. Der einzige Unterschied in der Behandlung besteht darin, dass bei der Bestimmung des Supports für jede

Sequenz nur höchstens ein Supportpunkt vergeben werden kann.

In einem zweiten Schritt werden die aus dem vorherigen Abschnitt erzeugten Kandidatensequenzen durch einen Datenbankscan auf ihren Support überprüft. Wenn eine Sequenz die minimale Supportanforderung erfüllt, so wird sie in die Menge der häufigen Sequenzen der Länge k übernommen.

Anschließend wird die Kandidatengenerierung mit der Länge $k+1$ neu gestartet. Dieses iterative Verfahren wird so lange durchgeführt, bis die Menge der häufigen Sequenzen der Länge k leer ist. Anschließend kann die Gesamtmenge der häufigen Sequenzen über alle Längen ausgegeben werden.

Hinweis: Innerhalb des Pseudocodes wird von einer lexikographischen Ordnung von Produktmengen ausgegangen.

Data: Grundmenge P, Datenbank D, Häufigkeitsschranke $minsupp$
Result: Menge der häufigen Sequenzen I
$C_1 = \{< i >: i \text{ ist häufige Produktmenge } \}$;
$k = 1$;
while $C_k \neq \emptyset$ **do**
  forall $D_j \in D$ **do**
    forall $C \in C_k$ **do**
      if $C \subseteq D_J$ **then**
        $abssupp_D(C) += 1$;
      end
    end
  end
  $I_k = \{C \in C_k : supp_D(C) \geq minsupp\}$;
  $I = I \cup I_k$;
  forall $X =< x_1, \ldots, x_k >, Y =< y_1, \ldots, y_k > \in I_k$ **do**
    if $x_1 = y_1$ AND $\ldots$ AND $x_{k-1} = y_{k-1}$ **then**
      $Z_1 =< x_1, \ldots, x_{k-1}, x_k, y_k >$;
      $Z_2 =< y_1, \ldots, y_{k-1}, y_k, x_k >$;
      if $\forall\ A \subset Z_1\ mit\ |A| = k : A \in I_k$ **then**
        $C_{k+1} = C_{k+1} \cup Z_1$;
      end
      if $\forall\ A \subset Z_2\ mit\ |A| = k : A \in I_k$ **then**
        $C_{k+1} = C_{k+1} \cup Z_2$;
      end
    end
  end
  $k += 1$;
end

Algorithm 3: Bestimmung häufiger Produktmengen durch den AprioriAll-Algorithmus samt Plus-Erweiterung

Modifikationen für sequenzielle Phasendatenbanken

Der PhaseAprioriAll-Algorithmus erweitert den klassischen AprioriAll-Algorithmus, um die Vorteile einer Phasendatenbank zu nutzen. Hierbei kann auf folgende Vorteile zurückgegriffen werden.

1. **Eingeschränkte Kandidatengenerierung:** Für Phasendatenbanken ist dank Lemma 5.7 bekannt, dass s-Erweiterungen ausschließlich innerhalb der gleichen oder einer höher geordneten Phasenmenge stattfinden können. Dies bedeutet, dass Kandidaten nur dann gebildet werden müssen, wenn die neue Produktmenge den Phasenanforderungen entspricht. Da im AprioriAll-Algorithmus immer zwei bestehende Sequenzen um das jeweils letzte Glied der anderen Sequenz erweitert werden, findet diese Erweiterung in der Regel nur in einem von beiden Fällen statt. Eine Ausnahme existiert dann, wenn beide letzten Glieder der gleichen Phase angehören.

2. **Nutzung von Phasengrenzen:** Bei der Ermittlung des Supports können die Phasengrenzen genutzt werden, um frühzeitig eine Supportüberprüfung als negativ abzubrechen, wenn durch die Phase der aktuell gescannten Produktmenge bekannt ist, dass keine Produktmengen einer früheren Phase in der gesuchten Teilsequenz mehr auftauchen können.

Die praktischen Zeiteinsparungen der Modifikationen können im nächsten Abschnitt eingesehen werden.

5.4.2 PhaseAprioriAllPlus

Der PhaseAprioriAllPlus-Algorithmus ist eine weitere Modifikation des klassischen AprioriAll-Algorithmus. Es werden alle Änderungen eingebaut, die bereits im PhaseAprioriAll-Algorithmus eingebaut wurden. Zusätzlich wird eine Modifikation genutzt, die bereits im klassischen Algorithmus als Empfehlung gegeben wurde.

Bei der Bildung der Sequenzerweiterungen der Länge n werden neue Sequenzen nur dann in die Kandidatenmenge für den nächsten Iterationsschritt übernommen, wenn für alle Teilsequenzen der Länge $n-1$ gilt, dass sie bereits in der aktuellen Menge der häufigen Sequenzen vorkommen. Da diese Menge bereits im Arbeitsspeicher zur Verfügung steht, müssen nur $n-1$ Tests auf Identität in der Menge der häufigen Sequenzen durchgeführt werden. Dies spart Berechnungszeit, da die Menge der häufigen Sequenzen in der Regel geringer ist als die Größe der Datenbank und außerdem ein Identitätstest schneller durchgeführt werden kann als ein Teilsequenztest.

5.4.3 PhasePrefixSpan

Der PhasePrefixSpan-Algorithmus ist eine modifizierte Version des beliebten PrefixSpan-Algorithmus, der in einigen Anwendungsfeldern immer noch als einer der schnellsten Algorithmen zur Suche von häufigen Sequenzen gilt. Bevor

die modifizierte Version vorgestellt wird, wird hier noch einmal die originale Version beschrieben.

Der allgemeine PrefixSpan-Algorithmus

Der PrefixSpan-Algorithmus wurde zuerst in [HP00] besprochen und später ausführlicher in [HPMA+01] bzw. [PHMA+04] vorgestellt. PrefixSpan steht für *Prefix-projected Sequential Pattern Mining* und nutzt einen Pattern-Growth-Ansatz zur Konstruktion von häufigen Sequenzen. Er ist eine Erweiterung des FreeSpan Algorithmus [HPMA+00] gleicher Autoren.

Pattern-Growth-Verfahren unterteilen den Suchraum mit einem Divide-and-Conquer-Verfahren in disjunkte Teilräume, in denen separat nach häufigen Sequenzen gesucht wird. Dies hat den Vorteil, dass bei der Konstruktion von häufigen Sequenzen nicht in jedem Berechnungsschritt die gesamte Datenbank durchsucht werden muss. Dies ist insbesondere wichtig, wenn Daten aufgrund der Größe nur auf einem langsamen Massenspeicher vorrätig sind, auf den nur linear zugegriffen werden kann.

Der PrefixSpan-Algorithmus bildet hierzu für jede i-Erweiterung (siehe Definition 4.8) und jede s-Erweiterung (siehe Definition 4.9) Projektionsdatenbanken von Sequenzen und speichert sie in einer entsprechend optimierten Datenstruktur. Ist beispielsweise eine häufige Teilsequenz ABA bekannt, so kann der Suchraum für Erweiterungen auf alle Sequenzen eingegrenzt werden, in denen die Teilsequenz ABA als Präfix vorkommt. Weiterhin müssen statt der vollständigen Sequenzen nur die jeweiligen Teilsequenzen nach dem ersten vollständigen Auftauchen der Teilsequenz ABA in der Datenstruktur abgelegt werden. Man spricht in diesem Kontext auch von Präfix- und Suffixbäumen.

Um die Effizienz weiter zu erhöhen, wird auf eine physische Konstruktion der Projektionsdatenbanken verzichtet. Stattdessen wird für jede Projektionsdatenbank ein Startpunkt in der Ausgangsdatenbank festgelegt und anhand von Zeigern und Abständen eine Liste einer virtuellen Datenbank erstellt.

Der PrefixSpan-Algorithmus versucht die Leistungsfähigkeit der Suche nach häufigen Sequenzen durch eine effizientere Datenbankorganisation und Suchraumorganisation zu steigern. Der Erfolg ist hierbei maßgeblich von der Dichte von häufigen Sequenzen in der Datenbank abhängig. Je schneller Teilbereiche der Datenbank komplett ignoriert werden können, desto schneller stellt sich in der Praxis ein Performancegewinn ein.

Allgemeiner Pseudocode

Data: Grundmenge P, Datenbank D, Häufigkeitsschranke $minsupp$
Result: Menge der häufigen Sequenzen I
Function `Start`$(\emptyset, P, D, minsupp)$
  I = RekursiverAufruf($P, D, minsupp$);
end
Function `RekursiverAufruf`$(S_{prefix}, P, D, minsupp)$
  Supp = ErmittleSupport(P, D) ;
  forall $p \in P$ mit $Supp[p] >= minsupp$ **do**
    $[S^i_{prefix}, D^i_p, S^s_{prefix}, D^s_p]$ = KonstruiereProjizierteDatenbank(D,p) ;
    I^i = Rekursiveraufruf($S^i_{prefix}, P, D^i_p, minsupp$) ;
    I^s = Rekursiveraufruf($S^s_{prefix}, P, D^s_p, minsupp$) ;
    $I = I \cup I^i \cup I^s$;
  end
  Return I ;
end

Algorithm 4: Bestimmung häufiger Sequenzen durch den PrefixSpan-Algorithmus: Teil 1

Function `ErmittleSupport`(P, D)
  forall $S \in D$ **do**
    forall $p \in P$ **do**
      if $<\{p\}> \subseteq S$ **then**
        $Supp[p] = Supp[p] + 1$;
      end
    end
  end
  Return Supp ;
end
Function `KonstruiereProjizierteDatenbank`(S_{prefix}, D, p)
  forall $S \in D$ **do**
    forall s_i in $S =< s_1, \ldots, s_l >$ **do**
      if $p \in s_i$ **then**
        $index =$ Index von p in s_i ;
        $s_{neu} = s_i \setminus \{s_i^1, \ldots, s_i^{index}\}$;
        $T =< s_{neu}, s_{i+1}, \ldots, s_l >$;
        $D_p^s = D_p^s \cup T$;
        $S_{prefix}^s =< S_{prefix}, \{p\} >$;
        break s_i ;
      end
    end
  end
  forall $S \in D$ **do**
    forall s_i in $S =< s_1, \ldots, s_l >$ **do**
      if *Letzte Produktmenge in* $S_{prefix} \cup \{p\} \subseteq s_i$ **then**
        $index =$ Index von p in s_i ;
        $s_{neu} = s_i \setminus \{s_i^1, \ldots, s_i^{index}\}$;
        $T =< s_{neu}, s_{i+1}, \ldots, s_l >$;
        $D_p^i = D_p^i \cup T$;
        $S_{prefix}^i =< s_{prefix}^1, \ldots, S_{prefix}^k \cup \{p\} >$;
        break s_i ;
      end
    end
  end
  Return $[S_{prefix}^i, D_p^i, S_{prefix}^s, D_p^s]$;
end

Algorithm 5: Bestimmung häufiger Sequenzen durch den PrefixSpan-Algorithmus: Teil 2

Modifikationen des Algorithmus

In seiner grundsätzlichen Aufbauweise ist der PrefixSpan-Algorithmus bereits gut für die Handhabung von Phasendatenbanken konstruiert. Da er mit jeder

Präfixsequenz die Datenbank auf Suffixe projiziert und in ihnen weitersucht, zerlegt er bereits intuitiv Phasendatenbanken in ihre Phasenbestandteile, indem bereits durchlaufene Phasen nicht mehr betrachtet werden. Der Algorithmus kann jedoch noch zusätzlich um einige Punkte erweitert werden.

1. **Eingeschränkte i-Erweiterungen:** Der PrefixSpan-Algorithmus kontrolliert mit jeder projizierten Datenbank den Support aller einelementigen Teilsequenzen auf i-Erweiterungen und s-Erweiterungen. Hierbei kann die aktuelle Phase des letzten Präfixelementes ermittelt werden. Für Phasendatenbanken ist dank Lemma 5.7 bekannt, dass i-Erweiterungen ausschließlich innerhalb der gleichen Phasenmenge stattfinden können. Die Supportüberprüfung kann an dieser Stelle eingeschränkt werden.

2. **Eingeschränkte s-Erweiterungen:** Für s-Erweiterungen kommen nur Sequenzen infrage, die um ein Element aus der gleichen Phasenmenge oder einer zukünftigen Phasenmenge erweitert werden. Auch in diesem Fall kann die Supportüberprüfung eingeschränkt werden.

3. **Nutzung von Phasengrenzen:** Zu Beginn des Algorithmus kann für jede Sequenz der Datenbank in linearer Zeit (bezüglich der Datenbankgröße) eine Tabelle mit den Indizes der Start- und Endproduktmengen jeder Phase ermittelt werden. Hierbei werden virtuelle Phasenteildatenbanken durch Indexverschiebung erstellt. Bei der Kontrolle des Supports eines Elementes muss nur innerhalb der entsprechenden Indizes der Phasenteildatenbanken gesucht werden. Hierdurch lässt sich die Supportberechnung und die Projektion neuer Datenbanken beschleunigen.

Praktische Ergebnisse und Vergleiche aller Algorithmen können im Abschnitt 5.5 eingesehen werden.

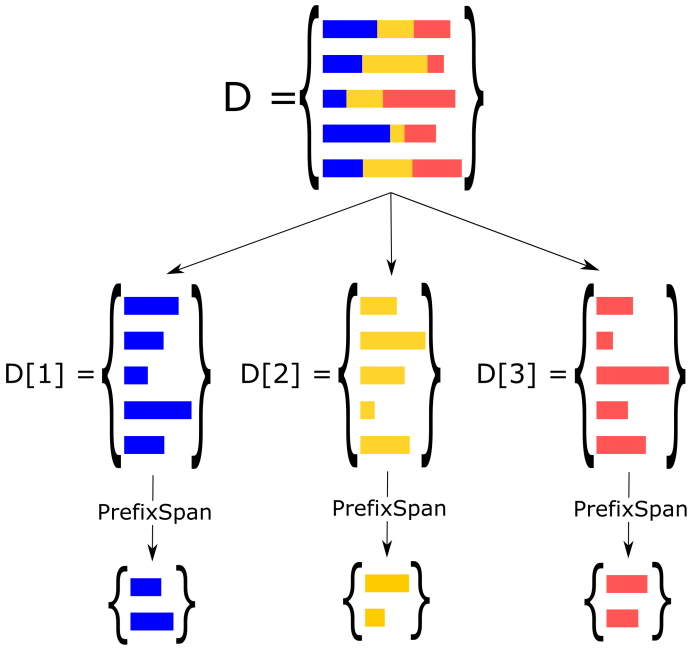
5.4.4 PhaseScan

Der PhaseScan-Algorithmus ist ein neuer Algorithmus, der als einer der Ergebnisse dieser Arbeit entwickelt wurde und die besondere Struktur von Phasendatenbanken berücksichtigt, um effizienter nach häufigen Sequenzen in einer Phasendatenbank zu suchen.

Allgemeiner Ansatz

In Theorem 5.13 und Folgerung 5.14 wurde gezeigt, dass eine häufige Sequenz einer sequenziellen Phasendatenbank nur dann häufig sein kann, wenn alle Phasenteilsequenzen in ihrer jeweiligen Phasenteildatenbank häufig sind. Dies bedeutet, dass eine nichthäufige Phasenteilsequenz nicht zu einer häufigen Sequenz in der gesamten Datenbank führen kann. Der PhaseScan-Algorithmus nutzt diese Tatsache und versucht das Problem der Sequenzsuche in der gesamten Datenbank auf die Konstruktion und Verknüpfung von häufigen Sequenzen aus häufigen Phasenteilsequenzen zu reduzieren.

In einem ersten Schritt wird das allgemeine Problem auf m Teilprobleme aufgesplittet und es werden die häufigen Teilsequenzen für jede Phasenteildatenbank ermittelt. Für dieses Verfahren kann grundsätzlich ein beliebiger Algorithmus zur Lösung des allgemeinen Problems genutzt werden. Im Rahmen dieser Arbeit wird hierzu auf den PrefixSpan-Algorithmus zurückgegriffen.



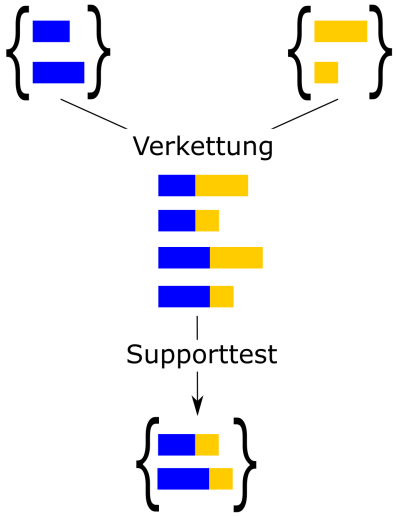
Eine Datenbank wird durch PhaseScan in ihre Phasendatenbanken aufgetrennt

In einem zweiten Schritt werden die nun ermittelten häufigen Phasenteilsequenzen Schritt für Schritt zu vollständigen häufigen Sequenzen zusammengesetzt. Es muss dabei bedacht werden, dass häufige Sequenzen nicht notwendigerweise alle Phasen durchlaufen müssen. Tatsächlich sind bereits die gefundenen Phasenteilsequenzen häufige Sequenzen der gesamten Datenbank. Insgesamt müssen deswegen alle möglichen Teilmengen der Menge der Phasen miteinander verknüpft werden. In Summe sind also bei m Phasen 2^m Phasenverknüpfungen notwendig. Dies mag zunächst kontraproduktiv wirken, da die Anzahl der Verknüpfungen mit der Anzahl der Phasen exponentiell wächst. Allerdings muss beachtet werden, dass bereits das Grundproblem der Suche nach häufigen Teilsequenzen einen exponentiell wachsenden Lösungsraum besitzt, der sich an der Länge n der längsten häufigen Sequenz orientiert. In der Regel gilt allerdings $n >> m$, sodass durch die Transformierung des Problems häufig eine erhebliche Reduzierung der Laufzeit erreicht werden kann.

Verkettung von häufigen Sequenzen aus Phasenteildatenbanken

Im PhaseScan-Algorithmus werden die häufigen Sequenzen aus Phasenteildatenbanken miteinander verkettet, indem an jede häufige Sequenz der ersten Phasenteildatenbank jede häufige Sequenz der zweiten Phasenteildatenbank

angehängt wird. Das Ergebnis bildet genau dann eine häufige Sequenz in der gesamten Datenbank, wenn der Support der verketteten Sequenz oberhalb der *minsupp*-Grenze liegt. Dieser Support muss für jede Neuverknüpfung überprüft werden.



Rekonstruktion von häufigen Sequenzen durch Verkettung von Teilsequenzen

Um die Supportüberprüfung möglichst effizient zu gestalten, bedient sich der PhaseScan-Algorithmus einer vertikalen Datenbankrepräsentation auf Basis der bisher gefundenen häufigen (Phasen-)Teilsequenzen. Dies bedeutet, dass für jede häufige (Phasen-)teilsequenz ein Bit-Vektor mit n Einträgen gespeichert wird, wobei n die Länge der gesamten Transaktionsdatenbank angibt. Ist die häufige (Phasen-)Teilsequenz in der Datenbanktransaktion i enthalten, so wird der Bit-Vektor an der Stelle i auf 1 gesetzt, ansonsten auf 0. Der Support ergibt sich damit aus der Summe der Werte im Bit-Vektor.

Supportberechnung durch effiziente Bitvektor-Logik

Möchte man den Support einer verketteten Sequenz ermitteln, so kann man die beiden Bit-Vektoren der Einzelsequenzen mit einer komponentenweisen UND-Verknüpfung zu einem neuen Bit-Vektor vereinen. Der neue Support ist durch die Summe der Werte des neuen Bit-Vektors gegeben. Dieses Verfahren arbeitet korrekt, da nur Vereinigungen zwischen Sequenzen aus unterschiedlichen Phasenabschnitten durchgeführt werden. Für diese Abschnitte ist dank der Phasenreihenfolge sichergestellt, dass es nur diese eine Verkettungsmöglichkeit gibt (siehe Eindeutigkeit der Phasenteilsequenzzerlegung). Es soll an dieser Stelle betont werden, dass dies eine Eigenschaft von sequenziellen Phasendatenbanken ist, die die Berechnung der Häufigkeit einer Verkettung signifikant beschleunigen kann.

Im allgemeinen Fall ist eine Supportberechnung durch UND-Verknüpfung

der Bit-Vektoren nicht zulässig, da das Einzelauftreten von Teilsequenzen und deren Hintereinanderverkettung nicht notwendigerweise den Support in einer Transaktionssequenz korrekt wiedergibt. So können die beiden Teilsequenzen in der Datenbank beliebig ineinander verschachtelt sein und sich gar gleiche Produktmengen teilen.

Unabhängigkeit von Datenbankscans

Im ersten Schritt wird beim PhaseScan-Algorithmus für jede Phasenteildatenbank ein separates Suchproblem durch den allgemeinen PrefixSpan-Algorithmus gelöst. Der allgemeine PrefixSpan-Algorithmus kann durch seine Natur über die Berechnung von projizierten Teildatenbanken bereits als Grundlage gemeinsam mit den häufigen Phasenteilsequenzen die projizierten Datenbanken ausgeben. Aus diesen projizierten Datenbanken lassen sich auf einfache Weise die Bit-Vektoren für den zweiten Teil des PhaseScan-Algorithmus ermitteln.

Sind die Bit-Vektoren der Teilprobleme im ersten Schritt ermittelt, so kommt der Phasescan-Algorithmus im weiteren Verlauf ohne die Informationen aus der allgemeinen Datenbank aus. Für die weiteren Verknüpfungen müssen nur noch Bit-Vektoren vereinigt und überprüft werden. Aus diesem Verzicht auf die allgemeine Datenbankinformation lassen sich mitunter signifikante Geschwindigkeitsgewinne produzieren, da alle notwendigen Informationen komprimiert auf Bit-Basis im Hauptspeicher zur Verfügung stehen.

Reduzierung des Verkettungsaufwands durch rekursiven Aufruf

Im PhaseScan-Algorithmus müssen sämtliche Kombinationsmöglichkeiten der Phasenteildatenbanken miteinander verkettet werden. Eine beliebige Anzahl von Phasenteildatenbanken wird miteinander verkettet, indem ausgehend von der ersten vorkommenden Phasenteildatenbank jeweils die nächste vorkommende Phasenteildatenbank angeschlossen wird. Die nachfolgenden Phasenteildatenbanken werden dann an das Ergebnis des letzten Schrittes angehängt. Durch dieses Verfahren müssen zu einem Zeitpunkt immer nur zwei Datenbanken und jeweils zwei Sequenzen aneinandergekettet werden. Das Ergebnis dieser Verkettung wird nur dann in die neue (Teil-)Datenbank übernommen, wenn die neue Sequenz häufig in der gesamten Datenbank ist. Auf diese Weise spart man Verkettungs- und Kontrollaufwand für alle Sequenzen, für die bekannt ist, dass bereits ein Teil der Sequenz nicht mehr häufig ist.

Um weiteren Aufwand einzusparen, werden die notwendigen 2^m Kombinationsmöglichkeiten an Phasenteildatenbanken rekursiv an einen Kombinationsbaum gehängt. Somit muss jede mögliche Teilkombination nur genau jeweils einmal berechnet werden. Details hierzu können dem Pseudocode entnommen werden.

Pseudocode des Algorithmus

```
Data: Grundmenge P, Datenbank D, Häufigkeitsschranke minsupp,
      Anzahl Phasen m, Phasenmengen Ph[1] ... Ph[m]
Result: Menge der häufigen Sequenzen I

Function Start(∅, P, D, minsupp)
    forall i ∈ {1, ..., m} do
        DP[i] = ErstellePhasendatenbank(D, Ph[i]) ;
        IP[i] = PrefixSpan(P, DP[i], minsupp) ;
    end
    I = RekursiverAufbau(IP, 0, minsupp, ∅) ;
    Return I ;
end

Function RekursiverAubau(IP, k, minsupp, I_akt)
    I_neu = ∅ ;
    if I_akt = ∅ then
        I_neu = IP[k] ;
    end
    else
        forall S ∈ I_akt do
            forall T ∈ IP[k] do
                I_neu = I_neu ∪ VerketteFallsHäufig(S, T) ;
            end
        end
    end
    // Phasenteilsequenzen mit Phase k verketten
    I_1 = RekursiverAufbau(IP, k + 1, minsupp, I_neu) ;
    // Phasenteilsequenzen ohne Phase k verketten
    I_2 = RekursiverAufbau(IP, k + 1, minsupp, I_akt) ;
    I = I_1 ∪ I_2 ;
    Return I ;
end
```

Algorithm 6: Bestimmung häufiger Sequenzen durch den PhaseScan-Algorithmus

5.4.5 PhaseTreeScan

Der PhaseTreeScan-Algorithmus ist ein neuer Algorithmus, der als einer der Ergebnisse dieser Arbeit entwickelt wurde und die besondere Struktur von Phasendatenbanken berücksichtigt, um effizienter nach häufigen Sequenzen in einer Phasendatenbank zu suchen. Er setzt im Grundkern auf dem PhaseScan-Algorithmus aus Abschnitt 5.4.4 auf, erweitert den Algorithmus jedoch um eine effizientere Verkettung der häufigen (Phasen-)Teilsequenzen.

Problem der Komplexität der Verkettungsmöglichkeiten

Wenn im PhaseScan-Algorithmus zwei Phasenteildatenbanken verkettet werden, so muss für jede Phasenteilsequenz der ersten Phasenteildatenbank und jede Phasenteilsequenz der zweiten Phasenteildatenbank eine Verkettung mit Supporttest durchgeführt werden. Der Aufwand dieser Überprüfung wächst quadratisch mit der Anzahl der gefundenen häufigen Phasenteilsequenzen.

Man könnte meinen, dass ein quadratischer Komplexitätsaufwand im Rahmen des ohnehin exponentiell wachsenden äußeren Rahmens zu vernachlässigen wäre. Doch zwei Gründe lassen eine effizientere Behandlung sinnvoll erscheinen.

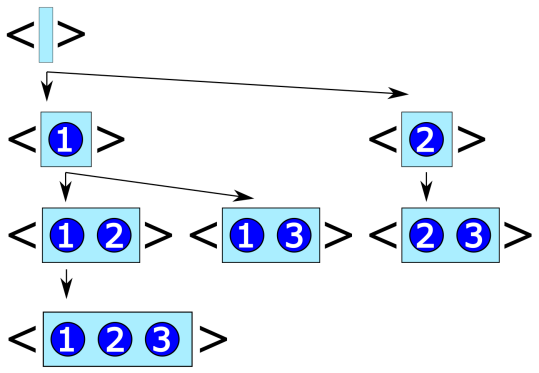
1. Die quadratisch wachsende Überprüfung muss für jede mögliche Kombination des äußeren Rahmens (Phasenkombinationen) durchgeführt werden. Die daraus resultierenden Komplexitätsaufwände multiplizieren sich statt sich unabhängig voneinander zu addieren.

2. Die Verkettungsüberprüfung wächst quadratisch mit der Anzahl der bereits gefundenen häufigen Phasenteilsequenzen. Diese Anzahl ist in der Regel wesentlich höher als die Anzahl der definierten Phasen. Dies bedeutet, dass selbst eine sehr geringe Anzahl von definierten Phasen im quadratischen Wachstum bei einer sehr hohen Anzahl an gefundenen häufigen Phasenteilsequenzen zu einer enormen Laufzeitbelastung führen kann.

Glücklicherweise kann aufgrund der Monotonie-Eigenschaft von häufigen Teilsequenzen ein effizienteres Verfahren zur Verkettung von zwei Phasenteildatenbanken gefunden werden.

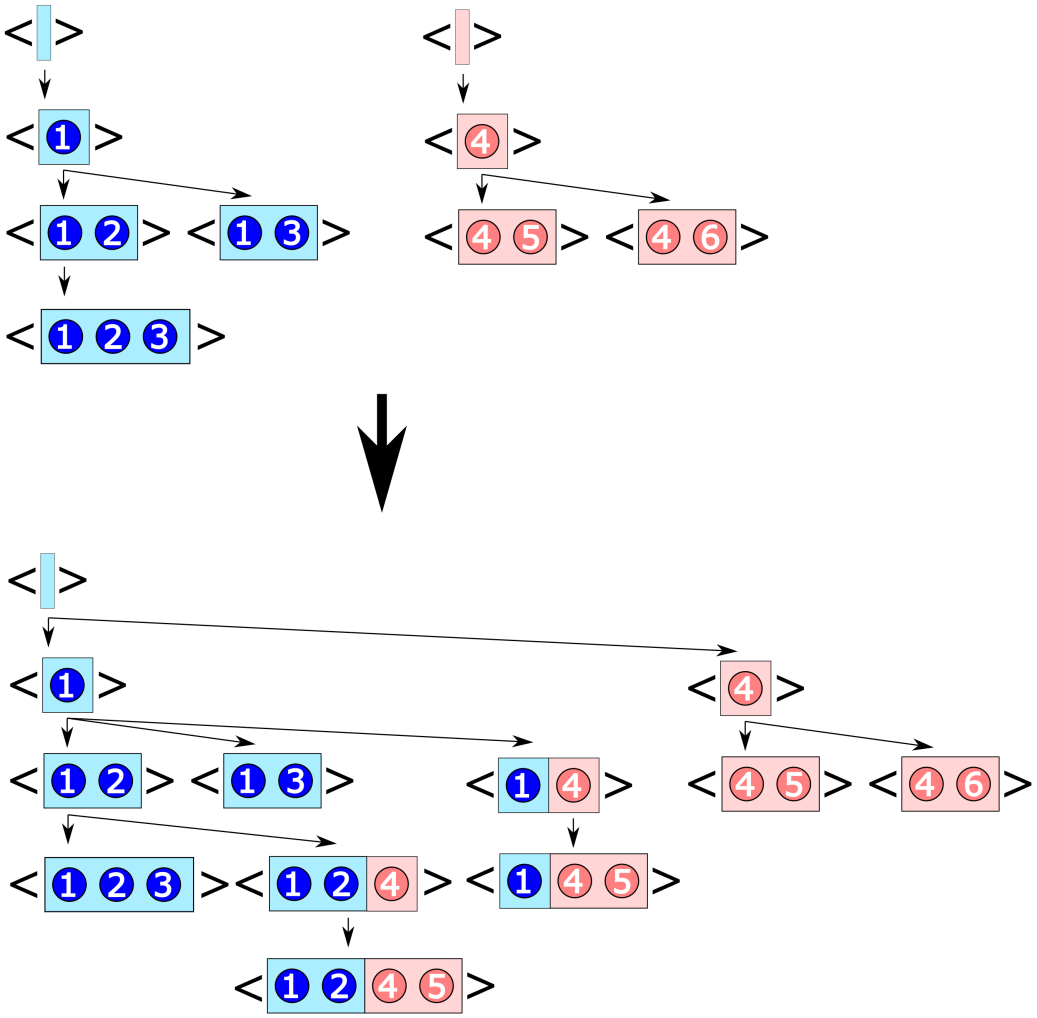
Ausnutzung von hierarchisch strukturierten Lösungsbäumen

Im PhaseScan-Algorithmus werden die bereits gefundenen häufigen Teilsequenzen in einer einfachen Menge bzw. linearen Liste erfasst. Diese Speicherung führt zu einem Verlust an Information, denn der zugrunde liegende PrefixSpan-Algorithmus, der für jede Phase im Vorfeld ausgeführt wird, kann die Ergebnisse in einer Baumstruktur ausgeben, in der die häufigen Sequenzen auf natürliche Weise gewachsen sind.



Baumstruktur von häufigen Phasenteilsequenzen als PrefixSpan-Ausgabe

Der PhaseTreeScan-Algorithmus nutzt die vorhandene Baumstruktur der häufigen Sequenzen in den Phasenteildatenbanken, um zwei Phasenteildatenbanken in eine größere Phasenteildatenbank zu vereinen, die ebenfalls eine Baumstruktur besitzt. Auf diesem Wege können effiziente Entscheidungskriterien genutzt werden, um unnötige Pfade in den Bäumen frühzeitig abzuschneiden.



Beispiel einer Baumstruktur von verketteten häufigen Phasenteilsequenzen

Wenn zwei Teilphasendatenbanken vereinigt werden, so beginnt der PhaseTreeScan-Algorithmus mit der Wurzel der linken Teilphasendatenbank. Für diesen Knoten wird geprüft, ob die Verkettung mit der Wurzel der rechten Teilphasendatenbank per Support-Vektor-UND-Verknüpfung zu einer häufigen Sequenz führt. Wenn dies der Fall ist, werden im rechten Baum alle Kinder des Knotens mit der Wurzel des linken Baumes getestet. Sobald die Verkettung mit einem Knoten des rechten Baumes zu keiner häufigen Sequenz führt, können alle Kinder dieses Knotens ignoriert werden, da sie Supersequenzen des Knotens repräsentieren, die damit ebenfalls nicht mehr häufig werden können.

Wenn alle Knoten des rechten Baumes abgearbeitet worden sind, können

die Kinder des Knoten des linken Baumes auf gleiche Weise behandelt werden. Sollte sich der Fall einstellen, dass für einen Knoten des linken Baumes nicht einmal die Wurzel des rechten Baumes in der Verkettung zu einer häufigen Sequenz führt, können sogar alle weiteren Kinder des linken Baumes für weitere Tests übergangen werden, da auch diese aufgrund der Monotonie-Eigenschaft nicht mehr häufig werden können.

Während der Algorithmus durch beide Bäume wandert, wird als Ausgabe parallel ein dritter Baum erstellt, in den alle neu verketteten Sequenzen eingehängt werden. Hierzu werden zunächst die beiden Bäume unverkettet unter eine neue gemeinsame Wurzel eingehängt, da sie zunächst aufgrund der Phasenstruktur keine Überschneidungen besitzen. Nun können beim Durchführen der Verkettungstests und beim Durchwandern des linken Baumes an gleicher Stelle im neuen Baum alle verketteten Sequenzen hinzugefügt werden, die sich durch Verkettung des Knotens mit dem rechten Baum ergeben. Im Endeffekt wird hierbei für jeden Knoten des linken Teilbaumes der rechte Teilbaum kopiert, solange der Supporttest positiv ausfällt. Am Ende ist ein neuer Sequenzbaum konstruiert worden, der in den Folgeschritten mit weiteren Phasenteildatenbanken verkettet werden kann.

Wenn alle möglichen Phasenkombinationen durchgetestet worden sind, gibt der PhaseTreeSpan-Algorithmus als Ausgabe einen Gesamtbaum aus, der in beliebiger Reihenfolge durchgegangen werden kann, um alle häufigen Sequenzen zu ermitteln.

5.5 Experimente und Testergebnisse

In diesem Abschnitt werden die im letzten Abschnitt vorgestellten neuen Algorithmen mehreren Testfällen unterzogen und deren Ergebnisse bezüglich der Laufzeit und der Anzahl von Kriterien wie Supportüberprüfung oder Datenbankscans ausgewertet. Hierzu wird ein neu entwickeltes synthetisches Datenmodell genutzt, welches flexibel durch Parameterwahl unterschiedliche Arten von sequenziellen Phasendatenbanken generieren kann. Diese Phasendatenbanken werden daraufhin als Eingabe an Algorithmen geliefert, um darin alle häufigen Sequenzen zu finden. Die neu entworfenen Algorithmen werden mit Standardalgorithmen aus der Literatur für die allgemeine Problemstellung verglichen.

5.5.1 Synthetisches Datenmodell

Das genutzte synthetische Datenmodell konstruiert eine gewünschte randomisierte sequenzielle Phasendatenbank durch die Vorgabe verschiedener Parameter. Folgende Eingaben können hierbei getätigt werden:

1. die Anzahl der Produkte in der Datenbank pro Phase,

2. die Anzahl der Phasen,
3. die maximale Länge der Phasenteilsequenzen einer Phase,
4. die maximale Länge an Elementen innerhalb einer einzelnen Produktmenge,
5. die Anzahl der Sequenzen in der Datenbank.

Als Produkte werden ohne Beschränkung der Allgemeinheit natürliche Zahlen verwendet. Die Randomisierung geschieht allein durch Wahl der Parameter. Das bedeutet, dass gleiche Parameter in jedem Fall zur gleichen sequenziellen Phasendatenbank führen. Insbesondere werden in einem Testfall alle Algorithmen mit der identischen Datenbank durchlaufen.

Beispiel einer sequenziellen Phasendatenbank durch synthetische Generierung

Im Folgenden sei beispielhaft das Ergebnis der Generierung einer sequenziellen Phasendatenbank für die nachfolgenden Parameter angegeben:

1. die Anzahl der Produkte in der Datenbank pro Phase: 10
2. die Anzahl der Phasen: 3
3. die maximale Länge der Phasenteilsequenzen einer Phase: 3
4. die maximale Länge an Elementen innerhalb einer einzelnen Produktmenge: 3
5. die Anzahl der Sequenzen in der Datenbank: 10

Generierte sequenzielle Phasendatenbank	
SID	Sequenz
1	< (5 6) 5 19 27 29 26 >
2	< 1 (2 7) 13 15 (12 18) 23 >
3	< (2 3 9) 7 (13 15) 11 (12 16) (21 25) 27 >
4	< (3 6) (2 4) (0 4) 16 (21 24) (20 29) 28 >
5	< (4 6) 5 18 11 12 21 26 25 >
6	< 9 (1 5 7 9) (16 17) 26 >
7	< 3 4 12 (28 29) >
8	< (0 6) 8 (1 6) 18 18 10 24 24 >
9	< (1 6) (5 6) 1 15 14 25 (23 27) (21 28 29) >
10	< 3 3 10 (17 18) 16 20 >

Tabelle 5.1: Eine synthetisch generierte sequenzielle Phasendatenbank

5.5.2 Vergleichsalgorithmen

Die in dieser Arbeit entwickelten Algorithmen werden Standardalgorithmen aus dem Gebiet der sequenziellen häufigen Mustererkennung gegenübergestellt. Der AprioriAll-Algorithmus und der PrefixSpan-Algorithmus wurden hierzu in einer eigenen Umsetzung implementiert, da sie die Grundlage für die modifizierten PhaseAprioriAll- und PhasePrefixSpan-Algorithmen bieten. Für die restlichen Algorithmen wurde auf die Open-Source-Bibliothek SPMF [PFV18] zurückgegriffen, die eine Vielzahl von implementierten Algorithmen zur Verfügung stellt, die auf einer gemeinsamen Datenschnittstelle aufbauen. Die Bibliothek wird von über zwei Dutzend aktiv Forschenden betreut und gepflegt.

1. **Der AprioriAll-Algorithmus:** Der Algorithmus wurde ausführlich in Abschnitt 5.4.1 vorgestellt.

2. **Der AprioriAllPlus-Algorithmus:** Der Algorithmus wurde ausführlich in Abschnitt 5.4.2 vorgestellt.

3. **Der PrefixSpan-Algorithmus:** Der Algorithmus wurde ausführlich in Abschnitt 5.4.3 vorgestellt.

4. **Der GSP-Algorithmus:** Der GSP-Algorithmus ist eine Weiterführung des AprioriAll-Algorithmus, der wesentlich besser mit der Größe der Datenbank skaliert. [SA96]

5. **Der Fast-Algorithmus:** Der FAST-Algorithmus wurde in [SFMH11] vorgestellt und nutzt indizierte dünnbesiedelte Listen zur Suche nach häufigen Sequenzen.

6. **Der SPAM-Algorithmus:** SPAM steht für "Sequential PAttern Mining" und gilt als einer der schnellsten Algorithmen zur Lösung des allgemeinen Problems. [AFGY02]

5.5.3 Testergebnisse

Alle folgenden Testergebnisse wurden auf einem Surface Pro 5 unter Debian 9 Linux ermittelt. Als Prozessor wurde ein Intel i5-7300U mit 4 Kernen genutzt und es standen rund 5 GB an freiem Speicher zur Verfügung. Es wurden keine parallelisierten Algorithmen verwendet, damit die Ergebnisse vergleichbar bleiben.

Testergebnisse für die AprioriAll- und AprioriAllPlus-Algorithmen

Die AprioriAll-Varianten werden in diesem Abschnitt unabhängig von den restlichen Algorithmen getestet. Da die AprioriAll-Algorithmen im Grundkern eine Kandidatengenerierung nutzen, die alle möglichen Kombinationen an Sequenzerweiterung bildet, leiden sie ohne Modifizierungen unter starken

Performanceeinbußen. Es kann jedoch gezeigt werden, dass im Falle von angepassten Phasenalgorithmen spürbare Performancegewinne erzielt werden können.

Als Testergebnisse werden neben der Laufzeit ebenso die generierten Kandidaten pro Iterationsschritt gezählt. Die Kandidatenmenge im Iterationsschritt n umfasst hierbei alle Sequenzen der Länge $n + 1$.

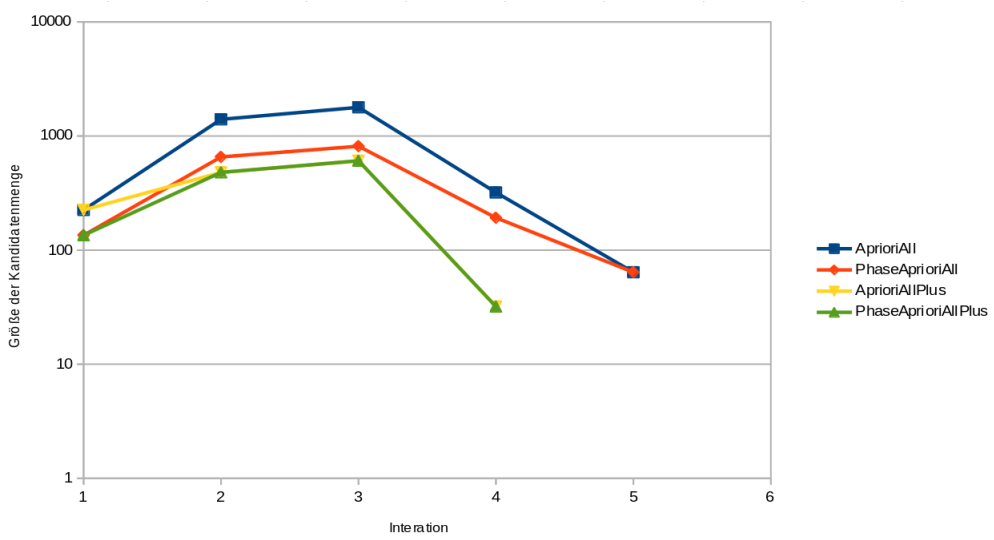
In diesem Test werden nachfolgende Parameter fixiert:

1. Anzahl der Produkte pro Phase: 10,
2. Anzahl der Phasen: 5,
3. maximale Länge einer Phasenteilsequenz: 3,
4. maximale Größe an Elementen innerhalb einer Produktmenge: 3,
5. Anzahl der Sequenzen in der Datenbank: 1000.

Der Parameter "minimaler Support" wird variabel gehalten und kann Werte zwischen 0.25 und 0.15 annehmen.

Anzahl der Kandidaten pro Iterationsschritt (It) bei $minsupp = 0.25$							
Algorithmus	Laufzeit (s)	It 1	It 2	It 3	It 4	It 5	It 6
AprioriAll	1.18	225	1400	1784	320	64	0
PhaseAprioriAll	0.46	135	656	814	192	64	0
AprioriAllPlus	0.68	225	480	607	32	0	0
PhaseAprioriAllPlus	0.44	135	480	607	32	0	0

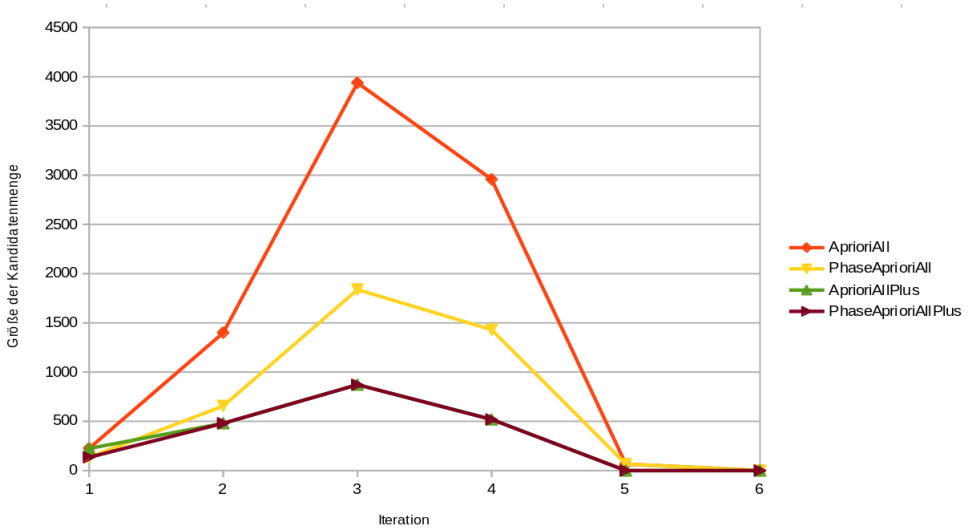
Tabelle 5.2: Ergebnisse der AprioriAll-Testreihe 1



Größe der Kandidatenmenge pro Iterationsschritt

Anzahl der Kandidaten pro Iterationsschritt (It) bei $minsupp = 0.20$							
Algorithmus	Laufzeit (s)	It 1	It 2	It 3	It 4	It 5	It 6
AprioriAll	12.66	225	1400	3938	2958	66	0
PhaseAprioriAll	4.50	135	658	1838	1430	66	0
AprioriAllPlus	5.55	225	480	872	519	0	0
PhaseAprioriAllPlus	3.64	135	480	872	519	0	0

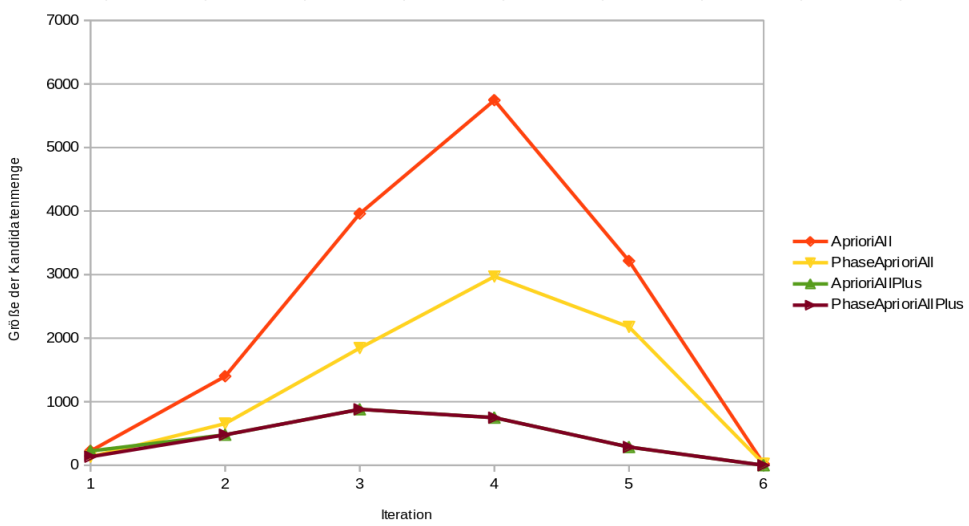
Tabelle 5.3: Ergebnisse der AprioriAll-Testreihe 2



Größe der Kandidatenmenge pro Iterationsschritt

Anzahl der Kandidaten pro Iterationsschritt (It) bei $minsupp = 0.15$							
Algorithmus	Laufzeit (s)	It 1	It 2	It 3	It 4	It 5	It 6
AprioriAll	77.30	225	1400	3960	5744	3215	24
PhaseAprioriAll	27.60	135	656	1844	2970	2177	24
AprioriAllPlus	18.99	225	480	880	749	287	0
PhaseAprioriAllPlus	14.78	135	480	880	749	287	0

Tabelle 5.4: Ergebnisse der AprioriAll-Testreihe 3



Größe der Kandidatenmenge pro Iterationsschritt

Das Ergebnis der Testreihe zeigt klar, dass die Formulierung einer sequenziellen Phasendatenbank bei einem angepassten Algorithmus erhebliche Laufzeit- und Speichergewinne bringen kann. Aufgrund der Phasenstruktur müssen nicht nur weniger Kandidatenmengen gebildet werden sondern die Supportüberprüfungen können auch effizienter durchgeführt werden.

Es zeigt sich aber auch, dass die Vorteile der geringeren Kandidatengenerierung bei der Erweiterung AprioriAllPlus bereits so effizient sind, dass eine weitere Aufteilung in Phasen hier kaum Wirkung zeigt. Dies liegt daran, dass die Überprüfung auf Identität mit Sequenzen der Länge $n-1$ bereits implizit die Phasenstruktur berücksichtigt. Trotzdem kann weiterhin ein Performancegewinn in der Laufzeit erreicht werden, der allerdings umso geringer wird, je wichtiger die Kandidatengenerierung wird.

Im Weiteren folgen ausführliche Testreihen unter Vergleich der Algorithmen, die auf einem Pattern-Growth-Ansatz aufbauen und damit mit erheblich längeren Transaktionsdatenbanken umgehen können.

Testergebnisse bei variabler Sequenzgröße mit durchschnittlicher Sequenzlänge

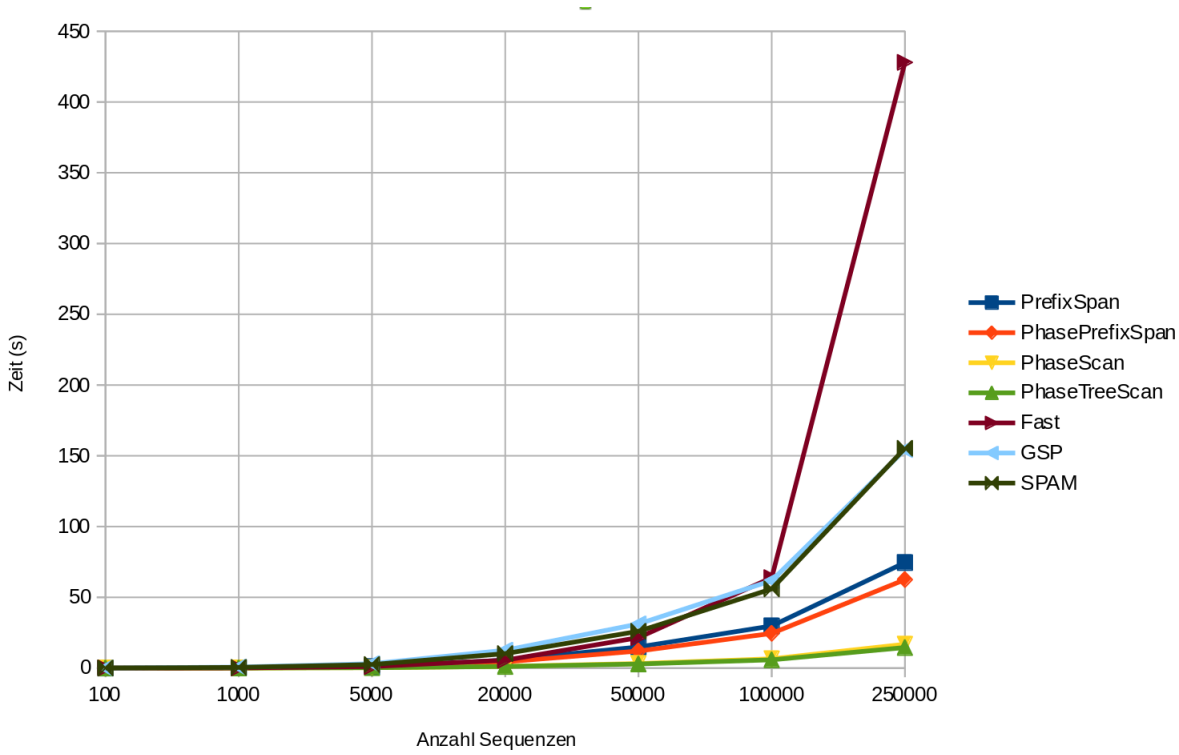
In diesem Test werdend nachfolgende Parameter fixiert:

1. Anzahl der Produkte pro Phase: 10,
2. Anzahl der Phasen: 5,
3. maximale Länge einer Phasenteilsequenz: 5,
4. maximale Größe an Elementen innerhalb einer Produktmenge: 3,
5. minimaler Support: 10%.

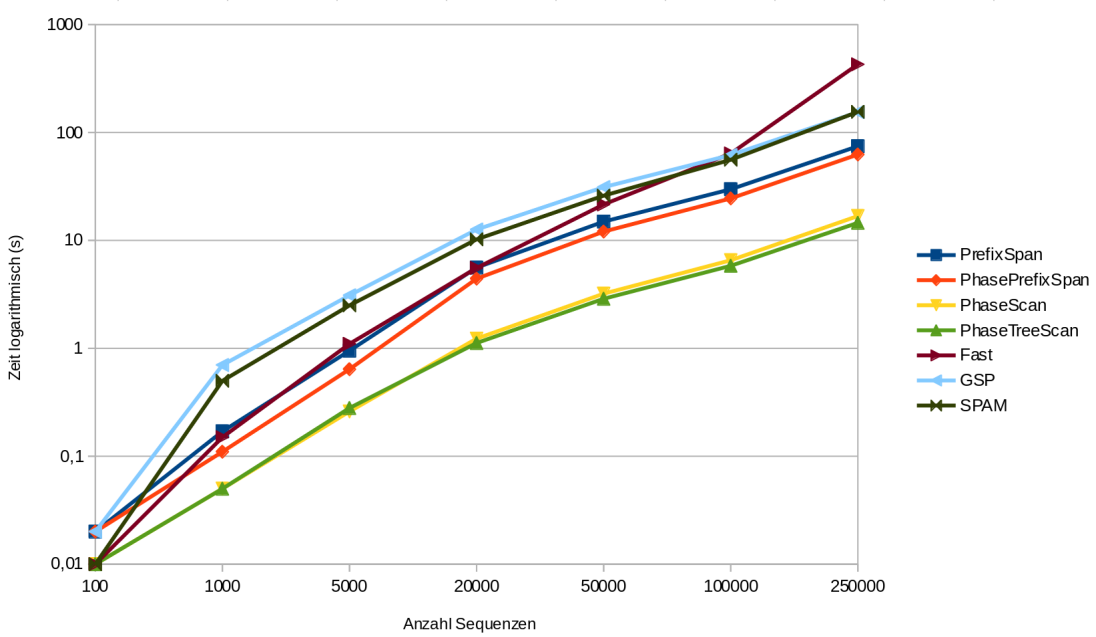
Der Parameter "Anzahl der Sequenzen in der Datenbank" wird variabel gehalten und kann Werte zwischen 100 und 250.000 annehmen.

Laufzeiten in Sekunden nach Sequenzgröße der Datenbank							
Anzahl Sequenzen	100	1000	5000	20000	50000	100000	250000
PrefixSpan	0.02	0.17	0.95	5,60	14,93	29,79	74,55
PhasePrefixSpan	0.02	0.11	0.64	4.42	12.07	24.51	62.52
PhaseScan	0.01	0.05	0.26	1.23	3.21	6.55	16.83
PhaseTreeScan	0.01	0.05	0.28	1.12	2.88	5.82	14.55
Fast	0.01	0.15	1.10	5.56	21.53	63.88	428.00
GSP	0.02	0.70	3.10	12.61	31.18	61.84	154.36
SPAM	0.01	0.50	2.50	10.23	26.00	55.93	155.31
Anzahl häufige Sequenzen	1182	1047	1050	1050	1050	1050	1050

Tabelle 5.5: Ergebnisse der Testreihe 1



Zeitdauer bei variabler Sequenzgröße der Datenbank



Logarithmische Zeitdauer bei variabler Sequenzgröße der Datenbank

Die Testreihe zeigt, dass der steigende Aufwand zur Berechnung aller häufigen Sequenzen von den Algorithmen sehr unterschiedlich verarbeitet wird. Während die Standardalgorithmen bei einer Sequenzgröße von 250.000 Sequenzen bereits über eine Minute reine Rechenzeit benötigen, kommen die neuen Algorithmen PhaseScan und PhaseTreeScan mit 17 bzw. 15 Sekunden aus. Anhand der Darstellung des zweiten Diagramms ist erkennbar, dass der logarithmische Abstand zwischen den Algorithmen im hinteren Bereich relativ konstant bleibt.

Die Ergebnisse sind nicht überraschend, denn während die Standardalgorithmen jede zusätzliche Sequenz in der Datenbank in ihrem Verfahren zusätzlich berücksichtigen, müssen die neuen Verfahren PhaseScan und PhaseTreeScan ausschließlich die Phasenteilprobleme mit einer gestiegenen Datenbankgröße lösen. Sobald dieser Schritt vollzogen ist, agieren sie nur noch auf der Anzahl der gefundenen häufigen Sequenzen. Diese Sequenzanzahl bleibt allerdings relativ konstant, da sich der minimale Support auf die Datenbanklänge bezieht und sich dementsprechend anpasst.

Testergebnisse bei variabler Sequenzgröße mit langer Sequenzlänge

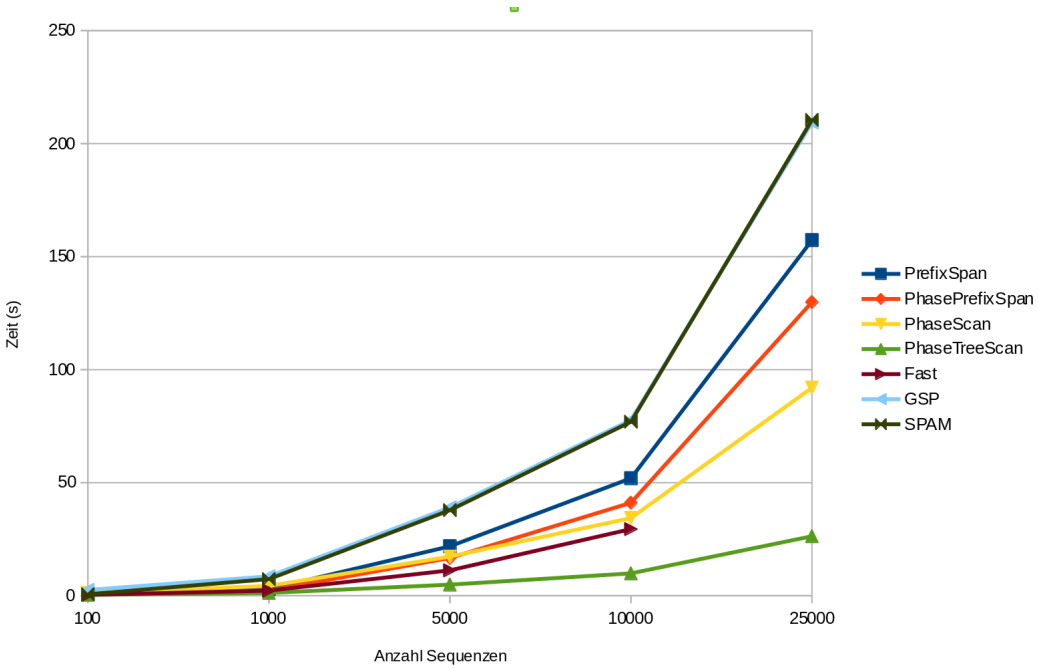
In diesem Test werdend nachfolgende Parameter fixiert:

1. Anzahl der Produkte pro Phase: 10,
2. Anzahl der Phasen: 5,
3. maximale Länge einer Phasenteilsequenz: 10,
4. maximale Größe an Elementen innerhalb einer Produktmenge: 3,
5. minimaler Support: 10%.

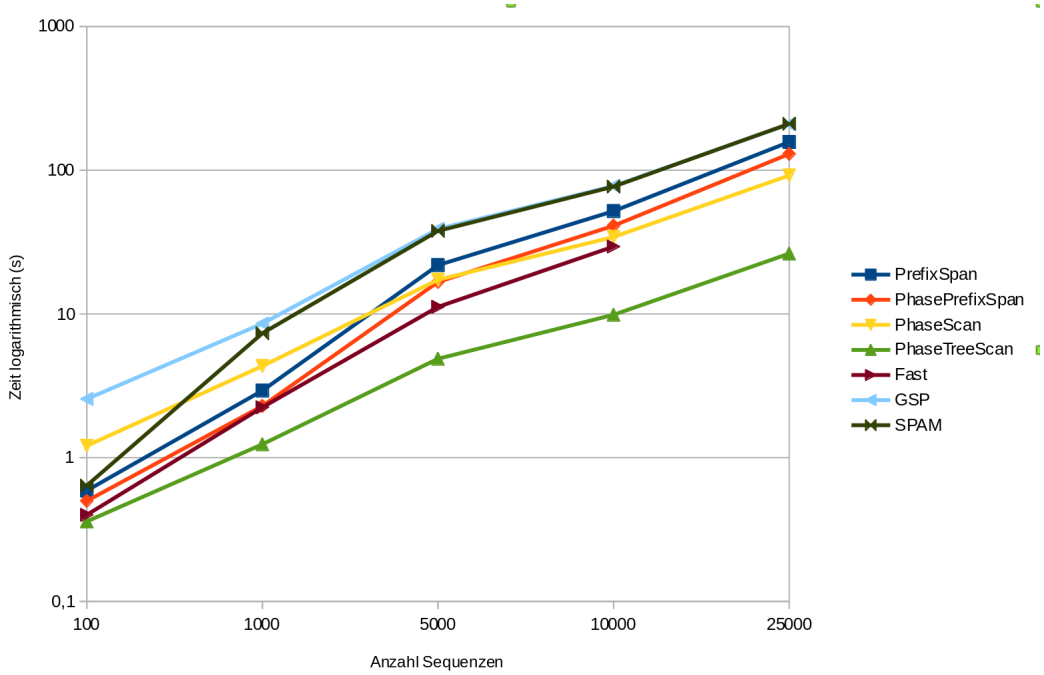
Im Gegensatz zum letzten Testfall wurde bei diesem Testfall die maximale Länge einer Phasenteilsequenz verdoppelt. Über alle Phasen können Sequenzen eine Länge von bis zu 50 Gliedern erreichen. Der Parameter "Anzahl der Sequenzen in der Datenbank" wird variabel gehalten und kann Werte zwischen 100 und 25.000 annehmen.

Laufzeiten in Sekunden nach Sequenzgröße der Datenbank					
Anzahl Sequenzen	100	1000	5000	10000	25000
PrefixSpan	0.59	2.93	21.86	51.95	157.30
PhasePrefixSpan	0.50	2.29	16.69	41.10	129.86
PhaseScan	1.21	4.34	17.29	34.33	92.00
PhaseTreeScan	0.36	1.24	4.88	9.87	26.30
Fast	0.40	2.24	11.21	29.46	-
GSP	2.56	8.57	39.05	77.73	209.30
SPAM	0.64	7.34	37.83	76.94	210.48
Anzahl häufige Sequenzen	53071	22984	21801	21869	23013

Tabelle 5.6: Ergebnisse der Testreihe 2



Zeitdauer bei variabler Sequenzgröße der Datenbank



Logarithmische Zeitdauer bei variabler Sequenzgröße der Datenbank

Auch in dieser Testreihe wird deutlich, dass die PhaseScan- und PhaseTreeScan-Algorithmen den anderen Algorithmen im geringeren Wachstum klar überlegen sind. Während im letzten Testfall noch Datenbankgrößen bis 250.000 betrachtet werden konnten, haben die anderen Algorithmen nun bereits bei 25.000 Sequenzen erhebliche Probleme. Der Fast-Algorithmus verweigerte in diesem Fall sogar aufgrund von zu wenig Speicherplatz den Dienst.

Testergebnisse bei variabler Phasenanzahl

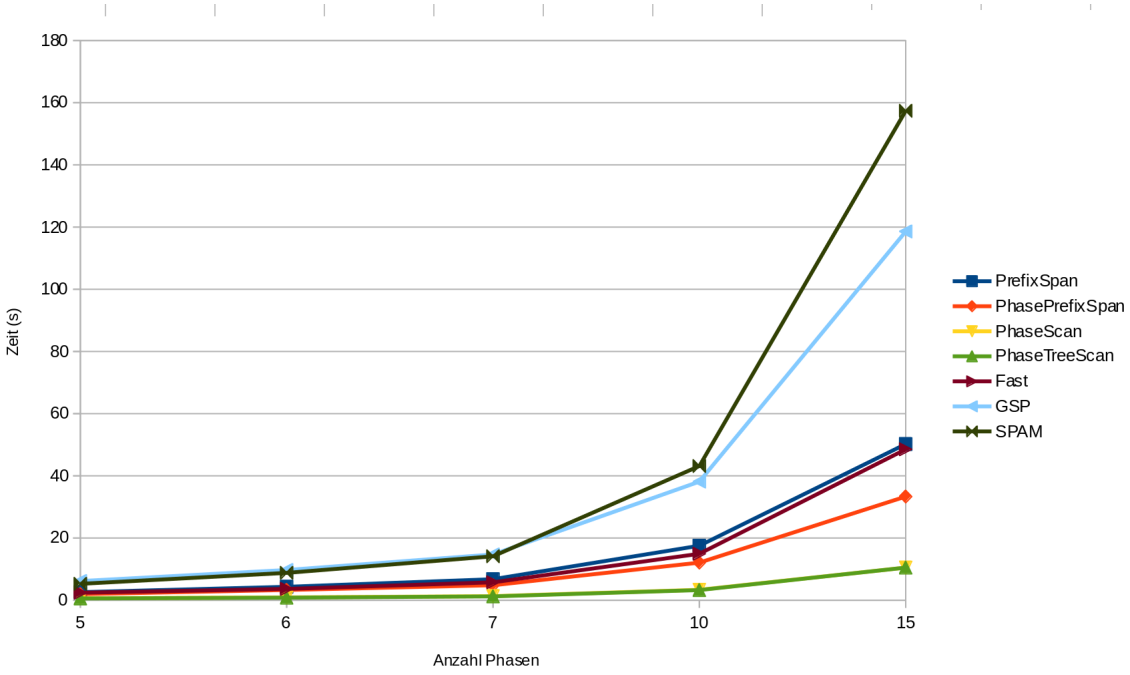
In diesem Test werdend nachfolgende Parameter fixiert:

1. Anzahl der Produkte pro Phase: 10,
2. Anzahl der Sequenzen: 10.000,
3. maximale Länge einer Phasenteilsequenz: 5,
4. maximale Größe an Elementen innerhalb einer Produktmenge: 3,
5. minimaler Support: 10%.

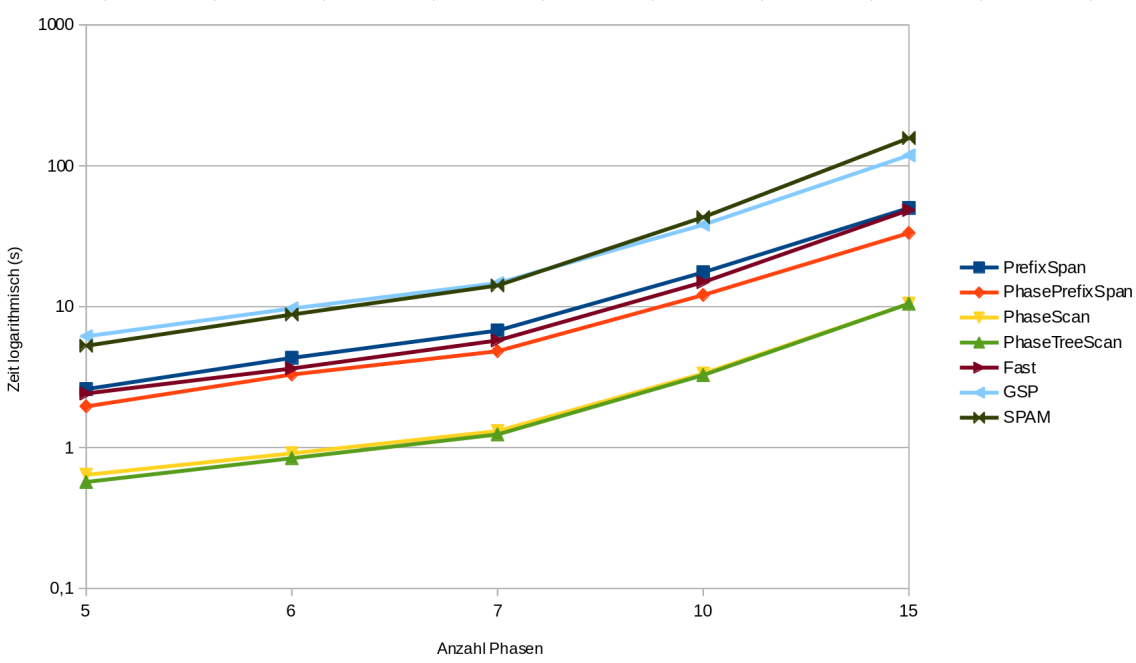
Der Parameter ”Anzahl der Phasen“ wird variabel gehalten und kann Werte zwischen 5 und 15 annehmen.

Laufzeiten in Sekunden nach Anzahl der Phasen					
Anzahl der Phasen	5	6	7	10	15
PrefixSpan	2.60	4.33	6.77	17.53	50.21
PhasePrefixSpan	1.96	3.30	4.83	12.10	33.34
PhaseScan	0.64	0.91	1.31	3.35	10.50
PhaseTreeScan	0.57	0.84	1.24	3.27	10.50
Fast	2.42	3.64	5.76	14.94	48.53
GSP	6.16	9.72	14.72	38.14	118.58
SPAM	5.28	8.80	14.13	43.16	157.40
Anzahl häufige Sequenzen	1050	1560	2170	4600	10650

Tabelle 5.7: Ergebnisse der Testreihe 3



Zeitdauer bei variabler Anzahl Phasen



Logarithmische Zeitdauer bei variabler Anzahl Phasen

Für die allgemeinen Algorithmen hat die Erhöhung der Anzahl der Phasen den ausschließlichen Einfluss, dass sich die Länge der Sequenzen erhöht, da die Länge der Phasensequenzen konstant bleibt. Für die neuen Algorithmen bedeutet die Erhöhung jedoch, dass sie zwar das Problem in eine größere Anzahl an Teilproblemen zerlegen können, am Ende jedoch eine größere Anzahl an Phasen wieder zusammenfügen müssen. Die Testreihe zeigt jedoch, dass eine Erhöhung der Anzahl der Phasen durch die neuen Algorithmen gut verkraftet wird. Tatsächlich sind sie immer noch um den Faktor 5-10 schneller als die allgemeinen Algorithmen, obwohl sich durch die Erhöhung der Phasen ebenso die Anzahl der häufigen Sequenzen erhöht.

Testergebnisse bei variabler maximaler Sequenzlänge

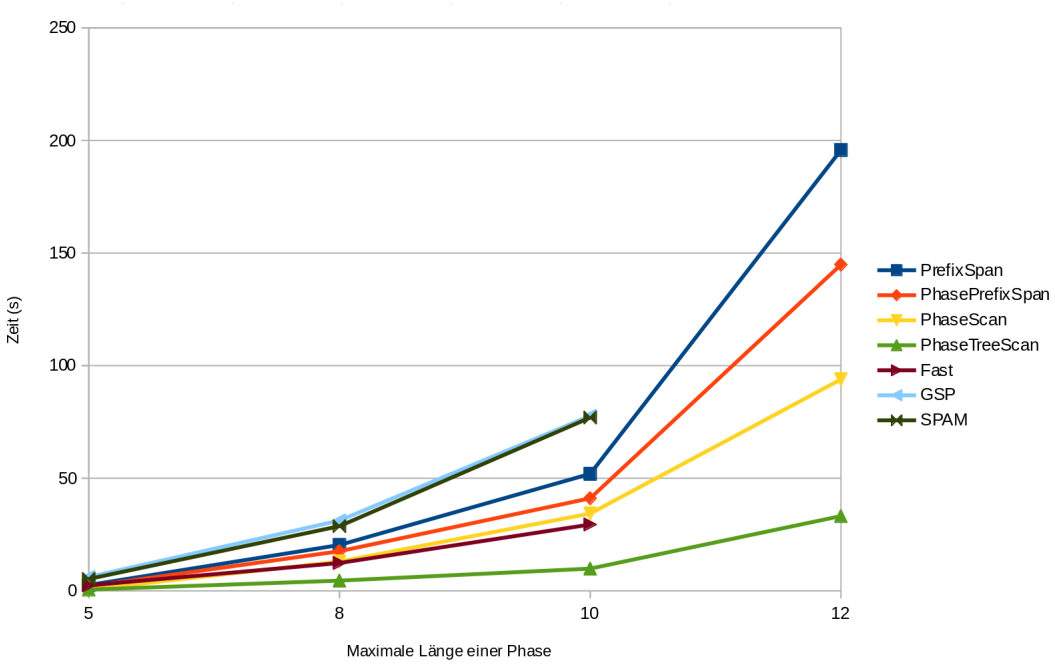
In diesem Test werdend nachfolgende Parameter fixiert:

1. Anzahl der Produkte pro Phase: 10,
2. Anzahl der Phasen: 5,
3. Anzahl der Sequenzen: 10.000,
4. maximale Größe an Elementen innerhalb einer Produktmenge: 3,
5. minimaler Support: 10%.

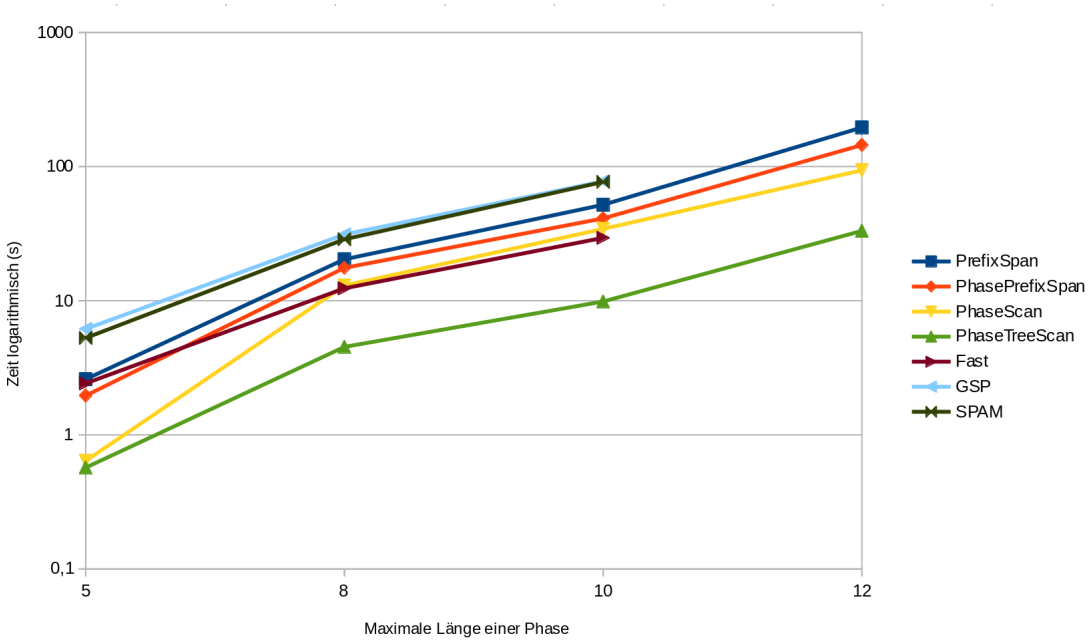
Der Parameter ”maximale Länge einer Phasenteilsequenz“ wird variabel gehalten und kann Werte zwischen 5 und 12 annehmen.

Laufzeiten in Sekunden nach maximaler Phasenlänge				
maximale Phasenlänge	5	8	10	12
PrefixSpan	2.60	20.37	51.95	195.75
PhasePrefixSpan	1.96	17.59	41.10	144.84
PhaseScan	0.64	12.99	34.33	93.93
PhaseTreeScan	0.57	4.53	9.87	33.20
Fast	2.42	12.38	29.46	-
GSP	6.16	31.22	77.73	-
SPAM	5.28	28.72	76.94	-
Anzahl häufige Sequenzen	1050	9412	21869	81548

Tabelle 5.8: Ergebnisse der Testreihe 4



Zeitdauer bei variabler maximaler Phasensequenzlänge



Logarithmische Zeitdauer bei variabler maximaler Phasensequenzlänge

Die Erhöhung der maximalen Länge der Phasensequenzen führt für die allgemeinen Algorithmen ausschließlich zu einer Verlängerung der Gesamtsequenz, ohne dass die zusätzliche Information genutzt werden kann. Die neuen Algorithmen können den Suchraum effizienter unterteilen.

In dieser Testreihe zeigt sich zum ersten Mal deutlich der Unterschied zwischen dem PhaseScan- und dem PhaseTreeScan-Algorithmus. Obwohl beide Algorithmen den Suchraum einschränken können, müssen sie am Ende die gefundenen häufigen Sequenzen wieder zusammenfügen. Während der PhaseScan-Algorithmus durch seine quadratische Laufzeit bei bis zu 81.548 häufigen Sequenzen langsam an seine Grenzen stößt, kann der PhaseTreeScan-Algorithmus von seiner Baumstruktur profitieren und hierdurch seinen Platz an der Spitze behaupten.

Testergebnisse bei variablem minimalen Support

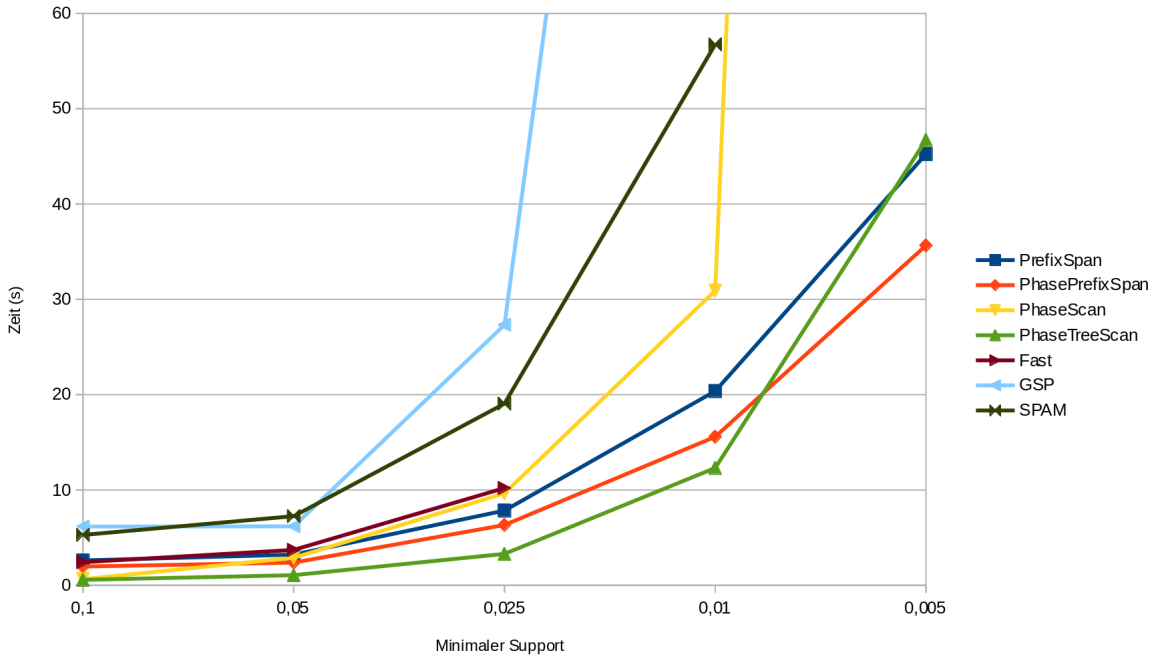
In diesem Test werdend nachfolgende Parameter fixiert:

1. Anzahl der Produkte pro Phase: 10,
2. Anzahl der Phasen: 5,
3. maximale Länge einer Phasenteilsequenz: 5,
4. maximale Größe an Elementen innerhalb einer Produktmenge: 3,
5. Anzahl der Sequenzen in der Datenbank: 10.000.

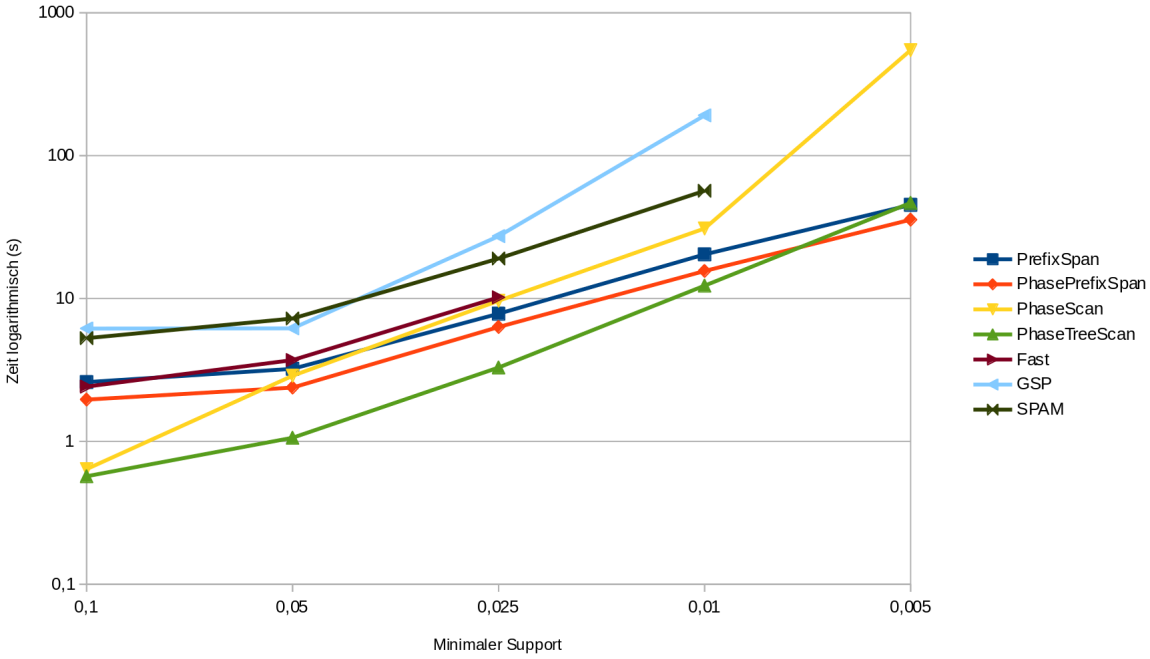
Der Parameter ”minimaler Support“ wird variabel gehalten und kann Werte zwischen 10% und 0.5% annehmen.

Laufzeiten in Sekunden nach minimalem Support					
minimaler Support	0.1	0.05	0.025	0.01	0.005
PrefixSpan	2.60	3.22	7.83	20.37	45.24
PhasePrefixSpan	1.96	2.38	6.32	15.58	35.65
PhaseScan	0.64	2.88	9.65	30.91	-
PhaseTreeScan	0.57	1.06	3.29	12.31	46.73
Fast	2.42	3.71	10.20	-	-
GSP	6.16	6.18	27.32	191.24	-
SPAM	5.28	7.25	19.06	56.75	-
Anzahl häufige Sequenzen	1050	1569	13222	96775	519867

Tabelle 5.9: Ergebnisse der Testreihe 5



Zeitdauer bei variablem minimalen Support



Logarithmische Zeitdauer bei variablem minimalen Support

Die Testreihe zeigt die Grenzen des neuen Ansatzes über die Verknüpfung von häufigen Phasenteilsequenzen auf und kann als Stresstest verstanden werden. Wenn der minimale Support einer Suche gesenkt wird, so führt dies in der Praxis dazu, dass immer längere häufige Sequenzen gefunden werden. Da wiederum alle Teilsequenzen dieser häufigen Sequenzen ebenfalls häufig sind, führt dies zu einer Vervielfachung der Lösungsmenge. Innerhalb der Testreihe müssen bei kleinstem minimalen Support 519.867 häufige Sequenzen gefunden und ausgegeben werden.

Der PhaseScan-Algorithmus gibt (wie drei der Vergleichsalgorithmen) bei dieser Datenkomplexität auf und liefert innerhalb von 5 Minuten keine Ausgabe. Die einzigen drei Algorithmen mit einer Ausgabe sind der PrefixSpan-, der PhasePrefixSpan- und der PhaseTreeScan-Algorithmus. Trotz seines effizienten Baumansatzes kostet das Zusammensetzen der häufigen Sequenzen im PhaseTreeScan-Algorithmus ungefähr so viel Zeit wie die Zerlegung des Suchraumes anfangs an Zeit gewinnt. Es kann davon ausgegangen werden, dass sich dieses Verhältnis bei einer noch höheren Anzahl an gefundenen häufigen Sequenzen weiter zum Negativen auswirkt. In dieser Testreihe zeigt der modifizierte PhasePrefixSpan-Algorithmus seine Stärken. Bei ihm ist garantiert, dass er immer ein besseres Ergebnis als der Prefix-Span-Algorithmus liefert wird, solange mehr als eine Phase betrachtet wird.

5.6 Fazit

In diesem Kapitel wurde ein neues Datenbankmodell in Form einer sequenziellen Phasendatenbank eingeführt. Es wurde gezeigt, dass effiziente Möglichkeiten

existieren, um den Suchraum in disjunkte Teilsuchräume zu zerlegen und es wurden neue Verfahren aufgezeigt, mit denen die Teilsuchräume anschließend korrekt und vollständig in kombinierte Lösungsmengen verarbeitet werden konnten.

In der Praxis lässt sich sowohl mit AprioriAll-modifizierten Algorithmen als auch mit PrefixSpan-modifizierten Algorithmen ein deutlicher Performancegewinn erzielen. Dies bedeutet, dass bereits kleinere Anpassungen an bestehenden Algorithmen ausreichen, um die Vorteile der Phasenstruktur zu nutzen.

Den deutlichsten Geschwindigkeitszuwachs liefert jedoch eine neuartige Klasse von Verfahren, die alle Vorteile der Phasenstruktur nutzen. Sie zerlegen nicht nur das Problem in disjunkte Teilprobleme, sondern nutzen auch fortgeschrittene und effiziente Baumstrukturen, um die Lösungsmengen anschließend zu verketten und zu vereinen. Dabei generieren sie einen Geschwindigkeitszuwachs, der in der Praxis von entscheidender Bedeutung ist.

Kapitel 6

Das SMMP-Modell zur effizienten Mustererkennung in komplexen Datenbanken

6.1 Motivation

In Kapitel 5 wurde ein neues Datenbankmodell für die Suche nach Mustern in sequenziellen Phasendatenbanken vorgestellt. Dieses Modell schafft die Grundlage für neue Algorithmen wie PhaseTreeScan (siehe 5.4.5), welche einen erheblichen Performancegewinn bei der Suche nach häufigen Sequenzen ermöglichen und damit das Schürfen in größeren oder komplexeren Strukturen erleichtern.

Obwohl die Suche nach häufigen Mustern in sequenziellen Datenbanken hilfreich ist und viele Anwendungsfelder hat, möchte man in der Praxis oft weitere oder komplexere Informationen in die Datenbank spielen, die sich durch den allgemeinen Ansatz so nicht direkt abbilden lassen. Hierzu zählen beispielsweise sequenzunabhängige Attribute oder die verknüpfte Behandlung mehrerer Sequenzen.

In diesem Kapitel wird ein Modell vorgestellt, welches die Vorteile einer sequenziellen Phasendatenbank mit weiteren aus der Literatur bekannten Erweiterungen zur sequentiellen Mustererkennung verbindet. Das Modell trägt den Namen **SMMP** und steht für "Sequenzielle Multidimensionale Mehrstufige Phasendatenbank".

In den folgenden Abschnitten werden zunächst die einzelnen Erweiterungen getrennt betrachtet und anschließend in ein gemeinsames Modell zusammengeführt. Es wird gezeigt, dass unter bestimmten Bedingungen die Vorteile von Phasendatenbanken weiterhin genutzt werden können, um flexibel und schnell nach komplexeren Mustern zu suchen.

6.2 Multidimensionale Sequenzsuche

In der klassischen Problemstellung der Mustererkennung in Sequenzen wird davon ausgegangen, dass die wichtigen Informationen bereits in der Existenz oder Nichtexistenz der Produkte innerhalb der Sequenz codiert sind. Im Falle des Einkaufs in einem Supermarkt möchte man schließlich Informationen über die gekauften Produkte und ihren Zusammenhang herstellen. Hierdurch lassen sich beispielweise Regalordnungen und Laufwege der Kunden optimieren.

Die meisten Unternehmen werden daran interessiert sein, ihren Umsatz durch Werbemaßnahmen zu steigern. Diese Maßnahmen richten sich naturgemäß an Menschen, die wiederum unterschiedliche Charakteristika aufweisen. Das Marketing möchte gezielt einen Teil der Kunden ansprechen, der sich in der Vergangenheit vermehrt für eine bestimmte Kombination von Produkten entschieden hat. Je mehr Informationen über einen Kunden vorliegen, desto genauer können diese in Zielgruppen segmentiert werden. So kann es sinnvoll sein, junge Kunden eher über E-Mail und Soziale Netzwerke anzusprechen, während für ältere Zielgruppen eher Radio- und Fernsehwerbung priorisiert wird.

Der Begriff der multidimensionalen Sequenzsuche wird in der Literatur nicht einheitlich angewandt. Grundsätzlich werden hierunter alle Verfahren verstanden, die die Sequenzsuche um zusätzliche Datendimensionen erweitern. Dies kann auf mehrere Arten erfolgen.

1. Für jede Sequenz werden weitere Daten betrachtet, die unabhängig von der Länge oder den Elementen der Sequenz sind. Diese Daten können beispielsweise der Wohnort oder die Berufsgruppe des Kunden sein. Im Gegensatz zu den Sequenzen sind diese Daten ungeordnet. Aus diesem Grund ist die multidimensionale Suche eng mit Pattern Mining in Mengensystemen verbunden [AS94]. In [PHP+01] werden Verfahren aus dem Pattern Mining in Mengensystemen und dem SPM-Problem zusammengeführt, um Lösungsmengen zu generieren.

2. Für jedes Produkt einer Sequenz werden weitere Daten betrachtet. Dies kann beispielsweise der Ort sein, an dem dieses Produkt gekauft wurde. Hierdurch ist es möglich, genauere Aussagen über die Beziehungen der Produkte untereinander zu tätigen. Ungeordnete Dimensionen werden unter anderem in [PCL+05] und [PLT06] behandelt. Hierbei werden zwei Algorithmen M^2SP und $HYPE$ vorgestellt, die einen Apriori-Ansatz verfolgen. Weiterhin werden in jeder Analysedimension baumartige Hierarchien betrachtet, die einer Generalisierung von Sequenzen erlauben.

3. In [CY02] und [YC05] wird die eindimensionale Sequenzsuche auf mehrere Dimensionen erweitert und ein Algorithmus $PrefixMDSpan$ zur Lösung vorgeschlagen. Auf diesen Ansatz wird näher in Abschnitt 6.3 eingegangen.

6.2.1 Definitionen und Problemstellung

Im Folgenden werden multidimensionale Merkmale als Kreuzprodukt von kategorischen Variablen betrachtet. Die Überführung von kontinuierlichen Variablen in kategorische Variablen ist außerhalb des Spektrums dieser Arbeit.

Definition 6.1. *Gegeben seien n unabhängige endliche Mengen $A_1, \ldots, A_n$ an Attributen. Eine Menge an n-Tupeln $D = \{D_1, \ldots, D_m\}$ mit $D_i \in A_1 \times \ldots \times A_n$ wird* ***multidimensionale Datenbank*** *genannt.*

Es ist möglich, die Erkennung von häufigen Mustern auf einer multidimensionalen Datenbank durchzuführen. Hierzu kann man entweder eine multidimensionale Datenbank als eine Produktmengendatenbank über die Vereinigung aller Attributmengen definieren oder speziell angepasste Algorithmen verwenden. Im ersteren Fall muss beachtet werden, dass der entstehende Suchraum durch die Vereinigung der Attributmengen zu einer Gesamtmenge in der Theorie unnötig vergrößert wird, da auch mehrere Ausprägungen aus einer Attributmenge gleichzeitig existieren können. Da der Suchraum in den genutzten Mustererkennungsalgorithmen aber sehr oft aus Kombinationen der auftretenden Datenbankmengen konstruiert wird, werden diese Fälle frühzeitig nicht weiterverfolgt.

Diese Arbeit beschäftigt sich hauptsächlich mit der Mustererkennung in multidimensionalen Sequenzdatenbanken.

Definition 6.2. *Gegeben seien n unabhängige endliche Mengen $A_1, \ldots, A_n$ an Attributen und eine Sequenzdatenbank $S = \{S_1, \ldots, S_m\}$. Eine Menge an Tupeln $D = \{D_1, \ldots, D_m\}$ mit $D_i = (d_i, S_i)$ und $d_i \in A_1 \times \ldots \times A_n$ wird* ***multidimensionale Sequenzdatenbank*** *genannt. Ein Tupel $D_i = (d_i, S_i)$ wird auch als multidimensionale Sequenz bezeichnet.*

Multidimensionale Sequenzdatenbanken speichern die multidimensonale Information also direkt als Eigenschaft ihrer Sequenzen.

Multidimensionale Sequenzdatenbank		
SID	Attribute	Sequenz
1	(21-30 Jahre, ledig, Angestellter)	< (Kaffee, Wasser), Milch >
2	(31-40 Jahre, verheiratet, Beamter)	< Kaffee , Zucker, Milch >
3	(61-80 Jahre, verwitwet, Rentner)	< Wasser, Zucker, Schokolade >

Tabelle 6.1: Ein Beispiel einer multidimensionalen Sequenzdatenbank (SID = Sequenzschlüssel)

Zur Bestimmung von häufigen Sequenzen in multidimensionalen Sequenzdatenbanken bedient man sich eines zusätzlichen Metasymbols. Dieses Symbol soll verhindern, dass der Support eines multidimensionalen Sequenzmusters zu schnell sinkt, weil die multidimensionale Datenbank zu viele Ausprägungen

annehmen kann. Aus diesem Grund wird die Definition für den Support in einer multidimensionalen Sequenzdatenbank leicht abgewandelt.

Definition 6.3. *Eine multidimensionale Sequenz* $M_a = (D_a, S_a)$ *mit* $D_a = (d_1^a, \ldots, d_n^a)$, $d_i^a \in (A_i \cup \{*\})$ *für* $i = 1, \ldots, n$ *und* $S_a =< a_1, a_2, \ldots, a_k >$ *ist eine* ***multidimensionale Teilsequenz*** *von* $M_b = (D_b, S_b)$ *mit* $D_b = (d_1^b, \ldots, d_n^b)$, $d_i^b \in A_i$ *für* $i = 1, \ldots, n$ *und* $S_b =< b_1, b_2, \ldots, b_m >$, *wenn*

1. *für* $i = 1, \ldots, n$ *gilt:* $d_i^a = d_i^b$ *oder* $d_i^a = \,''*''$,
2. *natürliche Zahlen* $1 \leq i_1 < i_2 < \ldots < i_k \leq m$ *existieren, so dass gilt* $a_1 \subseteq b_{i_1} \wedge a_2 \subseteq b_{i_2} \wedge \ldots \wedge a_k \subseteq b_{i_k}$.

M_b *wird* ***multidimensionale Supersequenz*** *von* M_a *genannt und mit* $M_a \subseteq M_b$ *beschrieben.*

Aufbauend auf dieser Definition lässt sich der Support definieren.

Definition 6.4. *Der* ***absolute Support*** $abssupp_D(M)$ *einer multidimensionalen Sequenz M in D ist die Anzahl der multidimensionalen Sequenzen* T_i *in D, für die gilt: M ist eine multidimensionale Teilsequenz von* T_i. *Der* ***relative Support*** *einer multidimensionalen Sequenz M in D ist*

$$supp_D(M) = \frac{abssupp_D(M)}{|D|}.$$

Aus den Definition ergibt sich auf bekannte Weise die Formulierung der Problemstellung des Schürfens nach häufigen multidimensionalen Sequenzen in einer Datenbank.

Problemstellung 6.5. *Gegeben sei eine Datenbank von multidimensionalen Sequenzen* $D = \{M_1, \ldots, M_m\}$ *und ein Schwellwert* $0 \leq minsupp \leq 1$. *Finde alle möglichen multidimensionalen Sequenzen* $T_i = (D_i, S_i)$ *mit*

$$D_i = (d_1^i, \ldots, d_n^i) \text{ mit } d_j^i \in (A_j \cup \{*\}) \text{ für } j = 1, \ldots, n$$

und einer Sequenz S_i, *für die gilt*

$$supp_D(T_i) \geq minsupp.$$

Die multidimensionalen Sequenzen T_i *werden* ***häufige multidimensionale Sequenzen*** *genannt.*

Beispiel 6.6. *Gegeben sei die multidimensionale Sequenzdatenbank in Tabelle 6.2. Mit einem minimalen Support von* $0,5$ *lässt sich hieraus die Liste der häufigen multidimensionalen Sequenzen in Tabelle 6.3 generieren.*

Multidimensionale Sequenzdatenbank				
SID	Alter	Familienstand	Berufstyp	Sequenz
1	21-30 Jahre	verheiratet	Angestellter	< (Kaffee, Wasser), Milch >
2	21-30 Jahre	ledig	Angestellter	< Kaffee, Milch >
3	31-40 Jahre	verheiratet	Selbständiger	< Wasser, Butter >
4	31-40 Jahre	verheiratet	Beamter	< Wasser, Milch >

Tabelle 6.2: Ein Beispiel einer multidimensionalen Sequenzdatenbank

Liste von häufigen multidimensionalen Sequenzen				
Häufigkeit	Alter	Familienstand	Berufstyp	Sequenz
0,75	*	*	*	< Wasser >
0,75	*	*	*	< Milch >
0,75	*	verheiratet	*	< Wasser >
0,5	*	*	*	< Kaffee >
0,5	*	*	*	< Kaffee, Milch >
0,5	*	*	*	< Wasser, Milch >
0,5	31-40 Jahre	*	*	< Wasser >
0,5	31-40 Jahre	verheiratet	*	< Wasser >
0,5	21-30 Jahre	*	*	< Milch >
0,5	*	*	Angestellter	< Milch >
0,5	21-30 Jahre	*	Angestellter	< Milch >
0,5	*	verheiratet	*	< Milch >
0,5	21-30 Jahre	*	*	< Kaffee >
0,5	*	*	Angestellter	< Kaffee >
0,5	21-30 Jahre	*	Angestellter	< Kaffee >
0,5	21-30 Jahre	*	*	< Kaffee, Milch >
0,5	*	*	Angestellter	< Kaffee, Milch >
0,5	21-30 Jahre	*	Angestellter	< Kaffee, Milch >
0,5	*	verheiratet	*	< Wasser, Milch >

Tabelle 6.3: Häufige multidimensionale Sequenzen mit minsupp = 0,5

6.2.2 Eigenschaften von häufigen multidimensionalen Sequenzen

Die Monotonie-Eigenschaft für häufige Sequenzen gilt auch im Falle von multidimensionalen Sequenzen.

Theorem 6.7. *Monotonie-Eigenschaft für häufige multidimensionale Sequenzen*
Sei $M_1 = (D_1, S_1)$ eine häufige multidimensionale Sequenz einer Transaktionsdatenbank D. Jede multidimensionale Teilsequenz $M_2 = (D_2, S_2) \subseteq M_1$ ist dann ebenfalls eine häufige multidimensionale Sequenz in D.

Beweis. Zum Beweis zeigen wir, dass für jede multidimensionale Transaktion $T = (D_T, S_T)$ der Datenbank D gilt: Wenn $M_1 \subseteq T$, so gilt auch $M_2 \subseteq T$. Damit wäre gezeigt, dass $abssupp_D(M_2) \geq abssupp_D(M_1)$ und somit $supp_D(M_2) \geq supp_D(M1)$. Aus dieser Aussage folgt die Korrektheit des Satzes.

Um zu zeigen, dass $M_2 \subseteq T$ gilt, beweisen wir die beiden Teilaussagen der Definition über multidimensionale Teilsequenzen.

1. Behauptung: Für $i = 1, \ldots, n$ gilt: $d_i^{M_2} = d_i^T$ oder $d_i^{M_2} = \text{''}*\text{''}$.
   Beweis: Für $i = 1, \ldots, n$ gilt: ($d_i^{M_1} = d_i^T$ oder $d_i^{M_1} = \text{''}*\text{''}$) und ($d_i^{M_2} = d_i^{M_1}$ oder $d_i^{M_2} = \text{''}*\text{''}$). Im Falle von $d_i^{M_2} = \text{''}*\text{''}$ gilt die Behauptung trivialerweise. Im Falle von $d_i^{M_2} = d_i^{M_1}$ ist $d_i^{M_2} = d_i^{M_1} = d_i^T$, womit ebenfalls die Behauptung gilt.

2. Sei $|S_T| = m$, $|S_1| = o$ und $|S_2| = l$.
   Behauptung: natürliche Zahlen $1 \leq n_1 < n_2 < \ldots < n_l \leq m$ existieren, so dass gilt $s_1^{S_2} \subseteq s_{n_1}^T \wedge s_2^{S_2} \subseteq s_{n_2}^T \wedge \ldots \wedge s_l^{S_2} \subseteq s_{n_l}^T$.
   Beweis: Da $S_1 \subseteq T$, existieren natürliche Zahlen $1 \leq j_1 < j_2 < \ldots < j_o \leq m$, so dass gilt $s_1^{S_1} \subseteq s_{j_1}^T \wedge s_2^{S_1} \subseteq s_{j_2}^T \wedge \ldots \wedge s_o^{S_1} \subseteq s_{j_o}^T$. Weiterhin gilt $S_2 \subseteq S_1$ und damit existieren natürliche Zahlen $1 \leq k_1 < k_2 < \ldots < k_l \leq o$, so dass gilt $s_1^{S_2} \subseteq s_{k_1}^{S_1} \wedge s_2^{S_2} \subseteq s_{k_2}^{S_1} \wedge \ldots \wedge s_l^{S_2} \subseteq s_{k_l}^{S_1}$. Hieraus folgt für alle $s_i^{S_2}$ mit $i = 1, \ldots, l$: $s_i^{S_2} \subseteq s_{k_i}^{S_1} \subseteq s_{j_{k_i}}^T$. Demnach ist die Behauptung mit $n_i = j_{k_i}$ gezeigt und $S_2 \subseteq T$.

□

6.2.3 Der UniSeq-Algorithmus

In [PHP+01] wird der Algorithmus UniSeq zur Erkennung aller häufigen multidimensionalen Sequenzen in einer multidimensionalen Datenbank erstmalig vorgestellt.

Im Grundkern besteht der Algorithmus aus einer Transformation der multidimensionalen Sequenzdatenbank in eine gewöhnliche Sequenzdatenbank. Hierzu wird für jede Ausprägung jeder Dimension der multidimensionalen Datenbank ein neues Element zur Grundmenge der Produkte hinzugefügt. Anschließend wird jede Transaktion der multidimensionalen Sequenzdatenbank in eine Transaktion einer traditionellen Sequenzdatenbank überführt, indem die Vereinigung der Ausprägungen der Dimensionen der multidimensionalen Datenbank für die Transaktion als erste Produktmenge in die Sequenz der Transaktion übernommen wird. Die Überführung kann in Tabelle 6.4 und Tabelle 6.5 eingesehen werden.

Multidimensionale Sequenzdatenbank				
SID	Alter	Familienstand	Berufstyp	Sequenz
1	21-30 Jahre	verheiratet	Angestellter	< (Kaffee, Wasser), Milch >
2	21-30 Jahre	ledig	Angestellter	< Kaffee, Milch >

Tabelle 6.4: Ein weiteres Beispiel einer multidimensionalen Sequenzdatenbank

Gewöhnliche Sequenzdatenbank	
SID	Sequenz
1	< (21-30 Jahre, verheiratet, Angestellter), (Kaffee, Wasser), Milch >
2	< (21-30 Jahre, ledig, Angestellter), Kaffee, Milch >

Tabelle 6.5: Die überführte Sequenzdatenbank

Nachdem die Datenbank nun der Struktur einer gewöhnlichen Sequenzdatenbank entspricht, kann prinzipiell ein beliebiger Algorithmus zur Suche nach häufigen Sequenzen in einer Sequenzdatenbank genutzt werden. Die Ergebnisse können anschließend wieder in multidimensionale Sequenzen zurücküberführt werden, indem das erste Sequenzglied mit den multidimensionalen Datenbankausprägungen (sofern vorhanden) abgeschnitten und ausgewertet wird. Es ist immer sichergestellt, dass für jede Dimension höchstens eine Ausprägung vorhanden ist, da die häufigen Sequenzen Teilsequenzen der Transaktionssequenzen sein müssen. Ist für eine Dimension keine Ausprägung vorhanden, so wird stattdessen das Metasymbol “*” genutzt, welches folgerichtig eine beliebige Ausprägung erlaubt.

Obwohl ein beliebiger Algorithmus zur Suche in einer Sequenzdatenbank genutzt werden kann, sind einige Algorithmen aufgrund der besonderen Transformationsstruktur der überführten Sequenzdatenbank besser geeignet als andere und ermöglichen ein günstigeres Laufzeitverhalten. Im UniSeq-Algorithmus wird hier auf den klassischen PrefixSpan-Algorithmus zurückgegriffen, der in Abschnitt 5.4.3 behandelt wurde. Dieser Algorithmus stellt dank seiner Suchstrategie durch projizierte Teilbäume sicher, dass die Erweiterung der Produktmenge durch die Ausprägungen der multidimensionalen Datenbank möglichst früh nach Behandlung der ersten Sequenzproduktmenge abgeschnitten werden kann.

6.2.4 Modellierung als sequenzielle Phasendatenbank

Der UniSeq-Algorithmus transformiert das multidimensionale Problem in ein gewöhnliches Sequenzproblem. In gleicher Veröffentlichung schlagen die Autoren vor stattdessen zwei unabhängige und unterschiedliche Algorithmen zur Lösung der Teilprobleme zu nutzen und die Teillösungen anschließend über projizierte Datenbanken wieder zu verbinden. Dieses Vorgehen entspricht im

Grundkern dem Vorgehen in einer sequenziellen Phasendatenbank mit einer zusätzlichen Phase, jedoch ohne einen gemeinsamen Algorithmus zu nutzen oder eine gemeinsame Sequenz zu bilden. Es werden auch keine weiteren Verallgemeinerungen auf einen generellen Ansatz angesprochen.

Die Modellierung über eine sequenzielle Phasendatenbank nutzt beide Vorteile. Zum einen wird der multidimensionale Anteil der Sequenz wie beim UniSeq-Algorithmus als erstes Glied einer verketteten Sequenz verstanden. Dieses neue Glied definiert eine neue Phase, die in allen Sequenzen an erster Stelle steht und deren neue Produkte in der Grundmenge dieser neuen Phase zugeordnet werden. Alle Voraussetzungen für eine Phasendatenbank sind damit eingehalten. Zum anderen kann nun durch einen gemeinsamen Algorithmus wie den PhaseTreeScan-Algorithmus das Problem gelöst werden, der weitaus effizienter ist als ein allgemeiner PrefixSpan-Algorithmus.

6.3 Mehrstufige Sequenzsuche

Die mehrstufige Sequenzsuche befasst sich mit der Suche nach häufigen Sequenzen in Sequenzen höherer Dimension. Im Unterschied zur mehrdimensionalen Sequenzsuche werden hierbei Sequenzen nicht um eine mehrdimensionale Datenbank von Attributen erweitert, sondern die Sequenzen selbst sind komplexerer Natur und bilden Sequenzen von Sequenzen beliebiger Ordnung. Hintergrund kann beispielsweise die Betrachtung mehrerer paralleler oder verschachtelter Strukturen wie die Auswertung von Netzwerkzugriffen nach Domains, IPs und schließlich Webseiten sein [CY02] [YC05]. Mehrstufige Sequenzsuchen können zusätzlich um mehrdimensionale Datenbanken erweitert werden [HZBR11].

Grundsätzlich kann die Mehrstufigkeit auf zwei verschiedenen Wegen formuliert werden. In der parallelen Mehrstufigkeit werden verschachtelte Sequenzen gleichberechtigt und gleichzeitig betrachtet. Dies bedeutet, dass zu einem festen Zeitpunkt eine beliebige Anzahl von parallelen Sequenzen existieren kann, die wiederum beliebig lang sein können. In der seriellen Mehrstufigkeit werden zwar auch mehrere Sequenzen von beliebiger Länge betrachtet, diese werden aber hintereinandergeschaltet. Beide Modellierungen bieten Vor- und Nachteile, auf die in den folgenden Abschnitten eingegangen wird.

6.3.1 Parallele Mehrstufigkeit

Definitionen und Problemstellung

Grundsätzlich bleiben die Definitionen von Produktmengen und 1-stufige Sequenzen bestehen. Ergänzt werden sie um eine neue rekursive Definition über n-stufige Sequenzen.

Definition 6.8. *Eine **n-stufige Sequenz** mit $n > 1$ ist eine endliche, geordnete Folge von*

(n-1)-stufigen Sequenzen. Sei S eine n-stufige Sequenz. Dann wird $S =< s_1, \ldots, s_m >_n$ *geschrieben und* $s_1, \ldots, s_m$ *bezeichnen (n-1)-stufige Sequenzen.*

Beispiel 6.9. *Gegeben sei ein Kunde, der mehrere Girokonten bei einer Bank parallel betreibt. Jedes Jahr werden für jedes Konto die Geldbewegungen in Sequenzen erfasst. Die Zeitsequenz ist eine 3-stufige Sequenz von 2-stufigen Sequenzen von Konten mit 1-stufigen Sequenzen von Geldbewegungen. Folgende 3-stufige Sequenz sei eingetreten:*

$$S =<<< 100, -50 >_1, < 50 >_1 >_2, << 100, -50 >_1, < 20 >_1, < 10, 10 >_1 >_2 >_3$$

Im ersten Jahr bedient der Kunde zwei Girokonten und eröffnet im zweiten Jahr ein drittes Girokonto, auf das er zweimal 10 Euro einzahlt.

Weiterhin wird die Definition einer Teilsequenz entsprechend um den n-stufigen Fall ergänzt.

Definition 6.10. *Gegeben seien zwei n-stufige Sequenzen* $S_a =< s_1^a, \ldots, s_l^a >_n$ *und* $S_b =< s_1^b, \ldots, s_m^b >_n$. S_a *ist eine* ***n-stufige Teilsequenz*** *von* S_b *wenn gilt: es existieren natürliche Zahlen* $i_1 < \ldots < i_l$ *mit "*s_j^a *ist (n-1)-stufige Teilsequenz von* $s_{i_j}^b$*" für* $j = 1, \ldots, l$. *Es wird dann* $S_a \subseteq S_b$ *geschrieben.*

Mit dieser Grundlage können absoluter und relativer Support über n-stufige Sequenzen gebildet werden und die Problemstellung kann folgerichtig verallgemeinert werden.

Problemstellung 6.11. *Gegeben sei eine Menge von n-stufigen Sequenzen* $D = \{S_1, \ldots, S_m\}$ *und ein Schwellwert* $0 \leq minsupp \leq 1$. *Finde alle möglichen n-stufigen Sequenzen*

$$T = \{T_i =< t_i^1, \ldots, t_i^m >_n : supp_D(T_i) \geq minsupp\}.$$

Der PrefixMDSpan-Algorithmus

Die Gültigkeit der Monotonieeigenschaft für häufige n-stufige Sequenzen ergibt sich trivial rekursiv, da bereits die Gültigkeit für 1-stufige Sequenzen gezeigt wurde. Auf dieser Basis kann ein Algorithmus zur Lösung der Problemstellung gebildet werden.

Wie der Name bereits suggeriert, besteht der PrefixMDSpan-Algorithmus im Kern aus einem angepassten PrefixSpan-Algorithmus. Der PrefixSpan-Algorithmus wird hierfür auf den n-stufigen Sequenzfall erweitert. Da der PrefixSpan-Algorithmus projizierte Datenbanken nutzt, wird an geeigneter Stelle eine rekursive Suchtabelle zur Erweiterung der aktuellen Präfix-Datenbank genutzt. Details können in [CY02] eingesehen werden.

Bewertung

Die Modellierung von parallelen mehrstufigen Sequenzen liefert eine hohe Flexibilität. Die Verschachtelung der Sequenzen kann nicht nur beliebig tief erfolgen,

es können auch eine beliebige Anzahl von Sequenzen beliebiger Länge zu jedem Zeitpunkt miteinander verknüpft werden. Als Ergebnisse der Problemstellung werden ebenso häufige Sequenzen ausgegeben, die eine sehr unterschiedlich strukturierte Gestalt annehmen können. Der Vorteil der Flexibilität wird durch einen starken Komplexitätsanstieg erkauft, denn im Worst-Case wächst die Anzahl der häufigen Sequenzen der Datenbank mit jeder weiteren Dimension und jeder weiteren parallelen Sequenz ungehindert exponentiell.

Bei der parallelen Mehrstufigkeit gehen insbesondere die Vorteile von Phasendatenbanken weitestgehend verloren, da die Partitionierung des Suchraums entweder über mehrere parallele Sequenzen aufgebrochen wird oder in tiefen Verschachtelungen verschwindet. Aus diesem Grund wird für das SMMP-Modell auf ein Modell der seriellen Mehrstufigkeit zurückgegriffen.

6.3.2 Serielle Mehrstufigkeit über Phasendatenbanken

Im Rahmen dieser Arbeit wird ein Modell für eine serielle Betrachtung mehrstufiger Sequenzen genutzt. Hierzu wird davon ausgegangen, dass innerhalb einer Stufendimension eine feste Reihenfolge der Sequenzen existiert. Diese Voraussetzung liefert die Möglichkeit, die Sequenzen mit neuer Codierung hintereinanderzuschalten und somit insbesondere die bestehenden Phasen nicht zu vermischen. Tatsächlich können die einzelnen Sequenzen damit selbst als Phasen begriffen werden. Die nachfolgenden Definitionen sind neuwertig und wurden in der Literatur auf diese Weise nach Wissen des Autors noch nicht betrachtet.

Definitionen und Problemstellung

Grundsätzlich bleiben die Definitionen von Produktmengen und 1-stufigen Sequenzen bestehen. Ergänzt werden sie um eine neue rekursive Definition über n-stufige Sequenzen.

Definition 6.12. *Eine **n-stufige Sequenz** S mit $n > 1$ ist eine Hintereinanderkettung einer endlichen Anzahl von (n-1)-stufigen Sequenzen S_i mit $i = 1, \ldots, k$. Man schreibt $S = S_1 \circ \ldots \circ S_k$.*

Hierzu wird jede $(n-1)$-stufige Sequenz als Phase der n-stufigen Sequenz begriffen. Insbesondere sind keine Überschneidungen der Grundmengen möglich. Werden k $(n-1)$-stufige Sequenzen mit den jeweiligen Grundmengen P_i mit $i = 1, \ldots, k$ als n-stufige Sequenz gebildet, so besitzt die neue n-stufige Sequenz die Grundmenge $P = P_1 \cup \ldots \cup P_k$. Gibt es Überschneidungen zwischen den einzelnen Grundmengen, so müssen sie durch neue Symbole vor der Vereinigung vermieden werden

Beispiel 6.13. *Gegeben seien die 1-stufigen Sequenzen $S_1 = ABCD$ und $S_2 = BACD$. Die 2-stufige Sequenz lautet $S = S_1 \circ S_2 = A^1B^1C^1D^1B^2A^2C^2D^2$.*

Besitzen die k $(n-1)$-stufigen Sequenzen jeweils l Phasen, so hat die verkettete n-stufige Sequenz insgesamt $k \cdot l$ Phasen.

Vorteile der Modellierung

Die Interpretation der Mehrstufigkeit als eine Verkettung zu einer langen Sequenz hat den theoretischen Vorteil, dass sämtliche Aussagen über Phasensequenzen und Phasendatenbanken bestehen bleiben. Insbesondere lassen sich die entstehenden Phasen sehr gut in unabhängige Teilphasen zerlegen, womit die Komplexität des Problems in vielen Anwendungsfällen verringert werden kann. Es stellt zusätzlich kein Problem dar, die Ergebnisse der Mustererkennung in der Verkettung wieder in Sequenzen niedrigerer Stufe aufzutrennen: hierzu müssen nur die Phasen an wohldefinierter Stelle wieder getrennt werden. Im letzten Kapitel wurde gezeigt, dass diese Trennung eindeutig ist.

Weiterhin besteht der praktische Vorteil, dass keine neuen Algorithmen zur Lösung des Problems geschaffen werden müssen. Da die Transformation bereits vor der Suche geschieht, können Algorithmen wie der in dieser Arbeit vorgestellte PhaseTreeScan-Algorithmus angewandt werden.

Nachteile der Modellierung

Die Modellierung besitzt einige Nachteile, deren Bedeutung von Anwendungsfall zu Anwendungsfall unterschiedlich gravierend sein können. Zunächst wird die Grundmenge der neuen Phasensequenz bei Dopplung von Elementen um neue Symbole erweitert. Im schlimmsten Fall besteht die neue Grundmenge aus einem Vielfachen der bisherigen Grundmenge. Dies hat nicht nur mögliche Auswirkungen auf die Performance der Suche, sondern transformiert auch die Problemstellung. Gemeinsam auftauchende Elemente in mehreren Phasenteilsequenzen können mitunter nicht mehr als solche erfasst werden und gehen bei der Bestimmung der Häufigkeit verloren.

Weiterhin spielt es nun eine Rolle, ob ein vorher gemeinsam genutztes Element nun in der zweiten Phasenteilsequenz oder in der dritten Phasenteilsequenz auftaucht. Diese Unterscheidung kann in manchen Anwendungen sogar gewünscht sein, ist jedoch im allgemeinen Fall nicht zu vermeiden. Dies ist ein charakteristischer Unterschied zur Modellierung über parallele Mehrstufigkeit, in der die Verdoppelung von Elementen der Grundmenge vermieden wird.

6.4 Das generalisierende SMMP-Modell

In den letzten Abschnitten konnte gezeigt werden, dass sich sowohl multidimensionale als auch mehrstufige Datenbanken über Phasendatenbanken abbilden lassen, wenn man mit einigen Einschränkungen einverstanden ist. An dieser Stelle soll das allgemeine Modell definiert werden.

Definition 6.14. *Eine **sequenzielle multidimensionale mehrstufige Phasendatenbank (SMMP)** besteht aus:*

1. *einer sequenziellen Phasendatenbank $S = \{S_1, \ldots, S_l\}$ mit Phasenmengen $P_1, \ldots, P_m$,*

2. *multidimensionalen Informationen* $A_i^S \subseteq A_1^S \times \ldots \times A_k^S$ *für jede Sequenz* $S_i \in S$,
3. *2-stufigen Sequenzen* $T = \{T_1, \ldots, T_n\}$ *mit*

$$T_i =< A_{a_1}^S > \circ S_{a_1} \circ \ldots \circ < A_{a_b}^S > \circ S_{a_b}$$

   mit $S_{a_1}, \ldots, S_{a_b} \in S$,
4. *multidimensionalen Informationen* $A_i^T \subseteq A_1^T \times \ldots \times A_j^T$ *für jede 2-stufige Sequenz* $T_i \in T$.

Zusammengefasst wird die SMMP in der Form $D = \{K_1, \ldots, K_n\}$ *mit* $K_i = (A_i^T, T_i)$ *für* $i = 1, \ldots, n$.

Die Flexibilität des SMMP-Modells lässt sich am besten an einem praktischen Beispiel erklären. Hierzu betrachten wir die Bausparverträge einer Bausparkasse. Die Bausparverträge einer Bausparkasse lassen sich als eine sequenzielle Phasendatenbank modellieren. Genaueres hierzu kann im Praxisteil dieser Arbeit gefunden werden. In diesem Beispiel existiert also eine Menge $S = \{S_1, \ldots, S_l\}$ an Bausparverträgen.

Jeder Bausparvertrag besitzt jedoch nicht nur sequenzielle Informationen wie Spargeldzugänge oder Tilgungsbeiträge sondern zusätzlich noch sequenzunabhängige Informationen wie beispielsweise die Höhe der Bausparsumme oder den zugehörigen Bauspartarif. Diese Informationen können in multidimensionalen Datenbanken in kategorischen Variablen gespeichert werden. Man erhält die Zusatzinformationen $A_i^S \subseteq A_1^S \times \ldots \times A_k^S$ für jeden Bausparvertrag S_i.

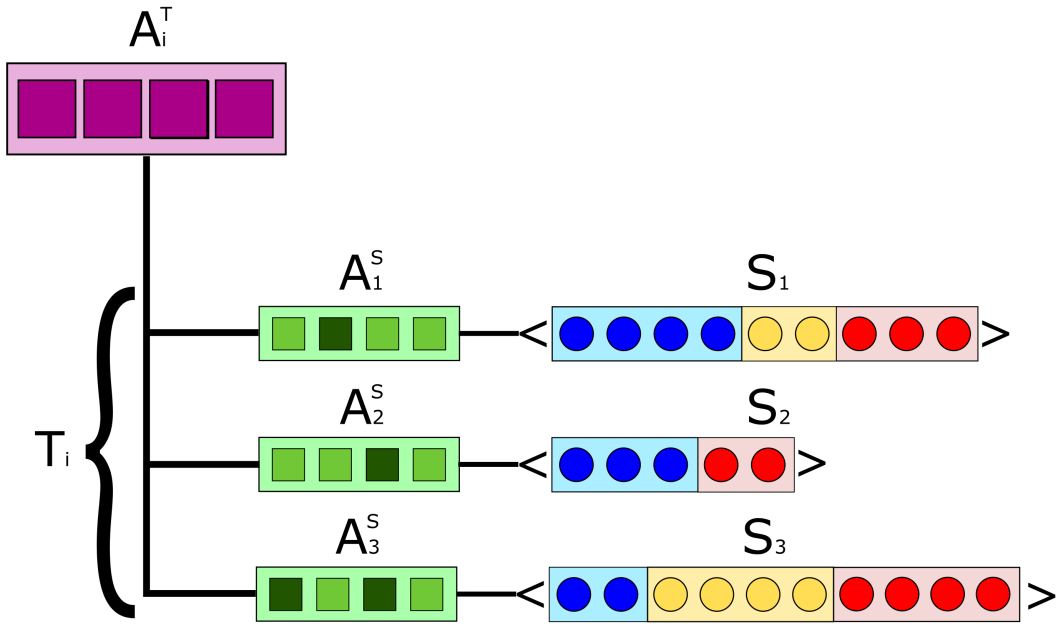
Bausparkassen schließen Verträge mit ihren Kunden und es ist nicht unüblich, dass ein Kunde hierzu mehrere Bausparverträge besitzt. So schließen viele Kunden nach dem Erhalt eines Darlehens einen neuen Bausparvertrag ab, den sie neu besparen. Die Menge der Bausparverträge a_1 bis a_b eines Kunden kann in einer 2-stufigen seriellen Sequenz

$$T_i =< A_{a_1}^S > \circ S_{a_1} \circ \ldots \circ < A_{a_b}^S > \circ S_{a_b}$$

erfasst werden.

Zusätzlich besitzt die Bausparkasse weitere Informationen über den Kunden, welche unabhängig von seinen Bausparverträgen sind. Hierbei kann es sich zum Beispiel um den Wohnort, die Berufsgruppe oder den Familienstand handeln. Diese Informationen werden wiederum in einer multidimensionalen Kundendatenbank $A_i^T \subseteq A_1^T \times \ldots \times A_j^T$ gespeichert.

Alle benötigten Daten zu einem Kunden i sind demnach in einem Tupel $K_i = (A_i^T, T_i)$ gespeichert. Die entstehende SSMP $D = \{K_1, \ldots, K_n\}$ kann daher als Grundlage von Mustererkennungsalgorithmen genutzt werden.



Visualisierung der Struktur eines Elements K_i der SMMP D

6.5 Suche nach häufigen Sequenzen im SMMP-Modell

Alle Informationen in einer SMMP lassen sich nach den Erkenntnissen aus diesem und dem letzten Kapitel in einer Menge von hintereinandergeketteten Sequenzen hinterlegen. Für einen Kunden K_i ergibt sich die Sequenz

$$K_i =< A_i^T > \circ < A_{a_1}^S > \circ S_{a_1} \circ \ldots \circ < A_{a_b}^S > \circ S_{a_b}.$$

Die Informationen aller Kunden können damit in einer sequenziellen Datenbank gespeichert werden. Da es sich aufgrund der Phasenstruktur um eine sequenzielle Phasendatenbank handelt, kann sie obgleich ihrer Länge effizient mit den entsprechenden Algorithmen aus Kapitel 5 nach häufigen Sequenzen durchsucht und analysiert werden.

Wenn alle häufigen Sequenzen gefunden wurden, können die Ergebnisse wieder auf ein strukturiertes Format zurückkonvertiert werden, ohne dass es hierbei zu Informationsverlust kommt. Die Rückführung von multidimensionalen und mehrstufig codierten sequenziellen Datenbanken wurde in den Abschnitten 6.2 und 6.3 behandelt.

6.6 Suche nach maximalen häufigen Sequenzen im SMMP-Modell

Grundsätzlich können alle entsprechenden Algorithmus zur Suche nach maximalen häufigen Sequenzen in sequenziellen Datenbanken zur Lösung genutzt werden. Ein Beispiel kann in Abschnitt 4.7.2 im VSMP-Algorithmus gefunden werden. Diese Algorithmen können jedoch nicht die zusätzlichen Informationen aus der Phasendatenbank nutzen, wodurch sie mitunter unnötige Berechnungsschritte durchführen müssen.

Der im Abschnitt 5.4.5 vorgestellte PhaseTreeScan-Algorithmus kann auf die Suche nach maximalen häufigen Sequenzen erweitert werden. Da er seine Ergebnisse bereits in einer Baumstruktur hinterlegt, kann diese Baumstruktur nach Ergebnisausgabe strukturiert auf maximale häufige Sequenzen untersucht werden. Hierzu werden Optimierungsansätze genutzt, die bereits im VSMP-Algorithmus verwendet werden.

1. Es wird ein Zwischenspeicher aller bisher gefundenen maximalen häufigen Sequenzen verwendet. Jedes Mal, wenn eine häufige Sequenz gefunden wird, wird sie mit dem Zwischenspeicher an Sequenzen verglichen. Existiert im Zwischenspeicher bereits eine Supersequenz, so kann die Sequenz nicht maximal sein und wird ignoriert. Existiert keine Supersequenz, wird die Sequenz aufgenommen und es werden alle Sequenzen im Zwischenspeicher entfernt, die Teilsequenz dieser Sequenz sind, da diese wiederum nicht maximal sein können.

2. Eine Sequenz muss nur dann mit dem Zwischenspeicher verglichen werden, wenn ihr natürliches Wachstum im Suchbaum keine häufige Supersequenz ergibt. Ansonsten ist bereits bekannt, dass es sich bei der Sequenz nicht um eine maximale Sequenz handeln kann. In der Praxis bedeutet dies, dass ausschließlich die Blätter des Baumes kontrolliert werden müssen, da Eltern von Baumknoten immer Teilsequenzen ihrer Kinder sind. Bereits bei Konstruktion des Baumes können deswegen in einer linearen Liste alle Blätter des Baumes notiert werden. Sobald ein Blattknoten einen Kindsknoten erhält, wird er wieder aus der Liste entfernt und das neue Kind hinzugefügt.

Der entstehende Algorithmus erhält den Namen **PhaseTreeScanMax** und gibt am Ende alle gesuchten häufigen maximalen Sequenzen aus.

Es muss erwähnt werden, dass der PhaseTreeScanMax-Algorithmus nicht schneller als der PhaseTreeScan-Algorithmus arbeitet, da das Ergebnis des PhaseTreeScan-Algorithmus zunächst abgewartet werden muss. Der PhaseTreeScan-Algorithmus setzt Sequenzen aus einzelnen Phasenteilsequenzen zusammen. Es kann jedoch bei der Zusammensetzung nie ausgeschlossen werden, dass eine Phasenteilsequenz später Teil einer maximal häufigen Sequenz wird. Insbesondere sind maximale häufige Sequenzen nicht zwingend in

ihren Phasenteilsequenzen maximal. Aus diesem Grund kann bei der iterativen Konstruktion des Baumes in Vorschritten kein Pfad vorzeitig abgeschnitten werden.

6.7 Suche nach geschlossenen häufigen Sequenzen im SMMP-Modell

Wie bereits im maximalen Fall können alle entsprechenden Algorithmus zur Suche nach geschlossenen häufigen Sequenzen in sequenziellen Datenbanken zur Lösung genutzt werden. Ein Beispiel kann in Abschnitt 4.7.1 im BIDE+-Algorithmus gefunden werden. Diese Algorithmen können jedoch nicht die zusätzlichen Informationen aus der Phasendatenbank nutzen, wodurch sie mitunter unnötige Berechnungsschritte durchführen müssen.

Der im Abschnitt 5.4.5 vorgestellte PhaseTreeScan-Algorithmus kann ebenso auf die Suche nach geschlossenen häufigen Sequenzen erweitert werden. Da es sich bei geschlossenen häufigen Sequenzen um maximale häufige Sequenzen bezüglich eines festen Support-Wertes handelt, kann ein ähnliches Verfahren wie zur Suche nach maximalen häufigen Sequenzen genutzt werden. Im Lösungsbaum des PhaseTreeScan-Algorithmus sind bereits alle notwendigen Informationen vorrätig, insbesondere ist für jede häufige Sequenz ihr Support bekannt.

1. Für jeden Supportwert wird ein Zwischenspeicher aller bisher gefundenen maximalen häufigen Sequenzen bezüglich dieses Supports verwendet. Jedes Mal, wenn eine häufige Sequenz gefunden wird, wird sie mit dem Zwischenspeicher an Sequenzen verglichen. Existiert im Zwischenspeicher bereits eine Supersequenz, so kann die Sequenz nicht maximal sein und wird ignoriert. Existiert keine Supersequenz, wird die Sequenz aufgenommen und es werden alle Sequenzen im Zwischenspeicher entfernt, die Teilsequenz dieser Sequenz sind, da diese wiederum nicht maximal sein können.

2. Eine Sequenz muss nur dann mit dem Zwischenspeicher verglichen werden, wenn ihr natürliches Wachstum im Suchbaum keine häufige Supersequenz mit gleichem Support ergibt. Da der Support einer Supersequenz nicht größer sein kann, fällt der Support mit Durchschreiten eines Baumpfades monoton ab. Sobald ein Knoten erreicht wird, dessen Support echt kleiner ist, kann der komplette Pfad für diesen Supportwert ignoriert werden.

Der entstehende Algorithmus erhält den Namen **PhaseTreeScanClosed** und gibt am Ende alle gesuchten häufigen geschlossenen Sequenzen aus.

6.8 Suche nach Generatoren im SMMP-Modell

Analog zur Suche nach geschlossenen Sequenzen im SMMP-Modell kann nach einem sehr ähnlichen Verfahren nach Generatoren gesucht werden. Dieses Verfahren wird bereits im VGEN-Algorithmus angewandt, der in Abschnitt 4.7.3 vorgestellt wurde. Hierzu wird erneut der PhaseTreeScan-Algorithmus um neue Funktionalitäten erweitert.

1. Für jeden Supportwert wird ein Zwischenspeicher aller bisher gefundenen Generatoren verwendet. Jedes Mal, wenn eine häufige Sequenz gefunden wird, wird sie mit dem Zwischenspeicher an Sequenzen verglichen. Existiert im Zwischenspeicher bereits eine Teilsequenz mit gleichem Support, so kann die Sequenz kein Generator sein und wird ignoriert. Existiert keine Teilsequenz mit gleichem Support, so wird die Sequenz aufgenommen und es werden alle Sequenzen im Zwischenspeicher entfernt, die Supersequenz dieser Sequenz sind und den gleichen Support besitzen, da diese wiederum nicht maximal sein können.

2. Eine Sequenz muss nur dann mit dem Zwischenspeicher verglichen werden, wenn ihr Vorgänger im Suchbaum nicht den gleichem Support hatte.

Der entstehende Algorithmus erhält den Namen **PhaseTreeScanGen** und gibt am Ende alle gesuchten häufigen Generatoren aus.

6.9 Suche nach Top-k-häufigen Sequenzen im SMMP-Modell

Wie bereits für die anderen Problemstellungen kann mit dem PhaseTreeScan-Algorithmus ebenfalls nach top-k-häufigen Sequenzen gesucht werden, wenn man den Algorithmus leicht erweitert. Die Erweiterungen orientieren sich hierbei am TKS-Algorithmus, der in Abschnitt 4.7.4 vorgestellt wurde.

1. In einem Zwischenspeicher werden die bisher gefundenen k Sequenzen mit dem höchsten Support gesammelt. Sobald ein neue Sequenz mit höherem Support gefunden wird, wird die Sequenz mit dem niedrigsten Support entfernt.

2. Es wird die Eigenschaft des Satzes 4.17 genutzt. Dies bedeutet, dass ein Pfad im Suchbaum nur dann weiterverfolgt werden muss, wenn der Support einer Sequenz größer ist als der Support der bisher gefundenen k Sequenzen mit höchstem Support. Ansonsten kann der gesamte folgende Teilbaum ignoriert werden.

3. Es wird ein Greedy-Verfahren angewandt, welches immer diejenigen Kinder weiter durchsucht, die den höchsten Support besitzen. Auf diese Weise erhofft man sich einen möglichst schnellen Wachstum des minimalen Supportwertes.

Der entstehende Algorithmus erhält den Namen **PhaseTreeScanTop** und gibt am Ende alle gesuchten k-häufigen Sequenzen aus.

6.10 Suche nach häufigen Assoziationsregeln im SMMP-Modell

In Abschnitt 3.3 wurde die Suche nach häufigen Assoziationsregeln für die Mustererkennung über Mengen definiert. Diese Definition lässt sich auf den (weit komplexeren) Fall des SMMP-Modells erweitern.

6.10.1 Definitionen und Problemstellung

Gegeben seien zwei verkettete Sequenzen K_1 und K_2 aus der SMMP D.

Definition 6.15. *Eine **Assoziationsregel** $A = (K_1 \Rightarrow K_2)$ über der SSMP D ist eine These*

$$(K_1 \subseteq T \Rightarrow K_1 \circ K_2 \subseteq T) \text{ für alle } T \in D.$$

Aufgrund der Definition und der Verkettungslogik der Sequenzen in einer SMMP können nur Assoziationsregeln gefunden werden, die sich auf der rechten Seite auf die gleiche oder spätere Phasen beziehen. Für das Thema dieser Arbeit ist dies jedoch ausreichend, da nur Assoziationsregeln betrachtet werden, die einen der sequenziellen Anteile der nichtverketteten SMMP erweitern.

Definition 6.16. *Der **absolute Support** $abssupp_D(A)$ einer Assoziationsregel $A = (K_1 \Rightarrow K_2)$ in der SSMP D ist die Mächtigkeit der Überdeckung von $K_1 \circ K2$. Der **relative Support** einer Assoziationsregel A in D ist*

$$supp_D(A) = \frac{abssupp_D(A)}{|D|} = \frac{abssupp_D(K_1 \circ K_2)}{|D|}.$$

Der Support einer Assoziationsregel gibt die Häufigkeit an, in der die Verkettung der beiden Sequenzen in der Datenbank vorkommt.

Definition 6.17. *Die **Konfidenz** $conf_D(A)$ einer Assoziationsregel $A = (K_1 \Rightarrow K_2)$ in D ist die relative Häufigkeit der Gültigkeit der These. Es gilt*

$$conf_D(A) = \frac{abssupp_D(K_1 \circ K_2)}{abssupp_D(K_1)}.$$

Die Konfidenz ist ein Wert zwischen 0 und 1 und berechnet die qualitative Wertigkeit der Assoziationsregel (bedingte Wahrscheinlichkeit). Eine Konfidenz von $0,5$ bedeutet, dass die Regel in der Hälfte der Transaktionen der gesamten Datenbank D gültig ist, in der die Sequenz K_1 enthalten ist. Es ist zu beachten, dass die Konfidenz nur definiert ist, wenn die Sequenz K_1 in mindestens einer Transaktion in D enthalten ist.

Mit Hilfe des Definitionsgerüsts kann nun die Problemstellung der Erkennung von häufigen Assoziationsregeln im SMMP-Modell aufgestellt werden.

Problemstellung 6.18. *Gegeben sei eine SSMP D, ein Schwellwert $0 \leq minsupp \leq 1$ und ein Schwellwert $0 \leq minconf \leq 1$. Finde alle Assoziationsregeln*

$$A_j = (K_1 \Rightarrow K_2) : (\exists\, T_i \in D : K_1 \subseteq T_i)$$

und

$$supp_D(A_j) \geq minsupp \text{ und } conf_D(A_j) \geq minconf.$$

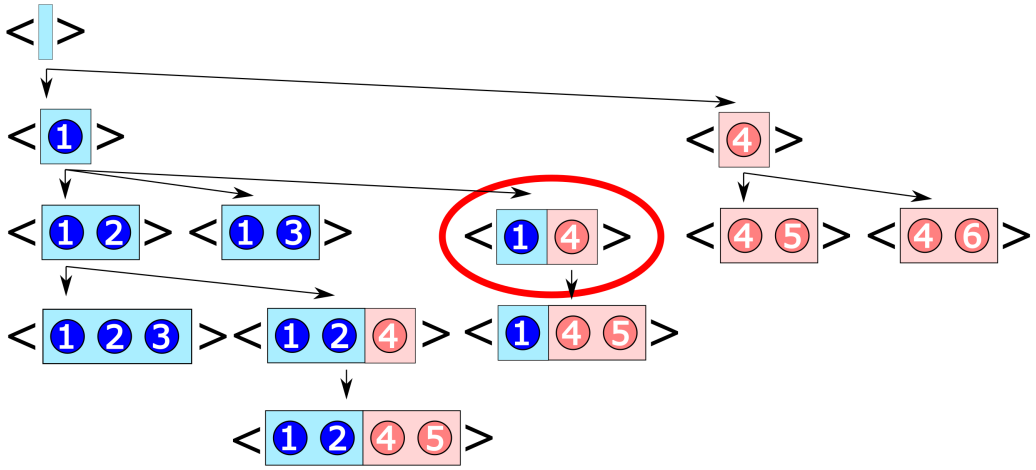
*A_j werden **häufige Assoziationsregeln** genannt.*

6.10.2 Verfahren zur Suche nach häufigen Assoziationsregeln

Assoziationsregeln im SMMP-Modell können gefunden werden, indem zunächst mit geeigneten Verfahren nach häufigen Sequenzen in D gesucht wird. Die häufigen Sequenzen können nun als die Verkettung der beiden Terme in der Assoziationsregel aufgefasst werden. Die Supportbedingung aus der Problemstellung ist damit bereits für alle Sequenzen der Ergebnismenge erfüllt.

Als nächster Schritt müssen aus den häufigen Sequenzen Assoziationsregeln formuliert werden. Hierzu wird jede Sequenz der Ergebnismenge in zwei Teilsequenzen $K = K_1 \circ K_2$ aufgespaltet. Die Assoziationsregel ergibt sich demnach als $A = (K_1 \Rightarrow K_2)$. Da die Assoziationsregel als Hintereinanderverkettung definiert ist, ergeben sich für jede Sequenz der Ergebnismenge $a + 1$ Möglichkeiten der Aufspaltung, wobei $a = |K|$, sofern leere Assoziationsglieder zugelassen werden.

Wurde ein Verfahren wie der PhaseTreeScan-Algorithmus genutzt, so liegen die Ergebnisse der Mustererkennung in einer Baumstruktur vor. Diese Struktur ist für die Erkennung von häufigen Assoziationsregeln sehr vorteilhaft, denn die Trennung in zwei Assoziationsglieder ist im Baum bereits auf natürliche Weise inkludiert. Betrachtet man einen beliebigen Knoten im Ergebnisbaum, so besteht K_1 aus der Sequenz, die sich als Pfad von der Wurzel bis zu diesem Knoten ergeben hat. Die Möglichkeiten für K_2 hingegen werden durch alle möglichen Pfade ab den Kindern bis zu einem der erreichbaren Blätter definiert.



$K_1 = <1\ 4>$ $K_2 = <5>$

Konstruktion von Assoziationsregeln $A = K_1 \circ K_2$ durch Baumtraversierung

Nach der Definition der Konfidenz

$$conf_D(A) = \frac{abssupp_D(K_1 \circ K_2)}{abssupp_D(K_1)}$$

sind alle benötigten Supportwerte im Baum vorhanden und müssen nur noch abgefragt werden. Da der Supportwert eines Knoten im Baum aufgrund der Monotonie-Eigenschaft für Sequenzen nie größer sein kann als der Supportwert seines Vorgängers, ergibt sich hieraus ein optimiertes Verfahren zur Suche nach allen häufigen Assoziationsregeln.

```
Data: Grundmenge P, Datenbank D, Häufigkeitsschranke minsupp,
      Konfidenzschranke minconf
Result: Menge der häufigen Assoziationsregeln A

Function Start(P, D, minsupp, minconf)
    Ergebnisbaum = PhaseTreeScan(P, D, minsupp);
    forall Knoten ∈ Ergebnisbaum do
        Temp = Knoten ;
        suppK2 = Knoten->Support ;
        repeat
            Temp = Knoten->Vorgänger ;
            suppK1 = Temp->Support ;
            conf = suppK2 / suppK1 ;
            if conf >= minconf then
                K1 = Temp->Sequenz ;
                K2 = Knoten->Sequenz ohne Temp->Sequenz ;
                A = A ∪ {K1 ⇒ K2} ;
            end
        until (Temp == Wurzel OR conf < minconf);
    end
    Return A ;
end
```

Algorithm 7: Bestimmung häufiger Assoziationsregeln durch den PhaseTreeScanAss-Algorithmus

6.11 Fazit

In diesem Kapitel wurde ein neues Datenmodell definiert, welches das in Kapitel 5 vorgestellte neue Modell um komplexere Datenstrukturen erweitert. Neben mehrdimensionalen sequenzunabhängigen Attributen wurde ebenso die Verknüpfung mehrstufiger Sequenzen behandelt. Es wurde gezeigt, dass alle Erweiterungen in einer sequenziellen Phasendatenbank abgebildet werden können. Hierdurch ist es möglich, effizientere Algorithmen zur Suche nach häufigen Mustern zu nutzen wie beispielsweise den neuen PhaseTreeScan-Algorithmus.

Zusätzlich wurden neue Erweiterungen zum PhaseTreeScan-Algorithmus entworfen, die es ermöglichen, mehrere abgewandelte Suchen wie die Suche nach maximalen oder geschlossenen Sequenzen durchzuführen.

Alle Ergebnisse dieses Kapitels können im folgenden Praxisteil der Arbeit genutzt werden, um neuwertige Erkenntnisse über das Data-Mining im Bausparwesen zu erlangen.

Teil II

Praxis

Kapitel 7

Allgemeine Beschreibung des Bausparens

Das Thema Bausparen erfreut sich in Deutschland konstanter Beliebtheit. Ende 2009 existierten nach Angaben der Deutsche Bank Research [AJ10] im Bestand deutscher Bausparkassen (öffentlich und privat) zirka 30 Millionen Bausparverträge. Hierunter fielen Guthaben in Höhe von rund 123 Milliarden Euro und Darlehen in Höhe von rund 113 Milliarden Euro. Diese Beträge waren auf insgesamt 24 Bausparkassen verteilt.

7.1 Die Bausparkasse

Nach §1 Absatz 1 des Bausparkassengesetzes gilt: "Bausparkassen sind Kreditinstitute, deren Geschäftsbetrieb darauf gerichtet ist, Einlagen von Bausparern (Bauspareinlagen) entgegenzunehmen und aus den angesammelten Beträgen den Bausparern für wohnungswirtschaftliche Maßnahmen Gelddarlehen (Bauspardarlehen) zu gewähren (Bauspargeschäft)". Ferner wird klargestellt: "Das Bauspargeschäft darf nur von Bausparkassen betrieben werden".

Bausparkassen schließen mit ihren Kunden Bausparverträge ab, in denen der Ablauf der Einzahlung von Bauspareinlagen und die Auszahlung und Tilgung von Bauspardarlehen geregelt wird. Im Allgemeinen werden Bausparmerkmale wie Guthabenzinsen, Darlehenszinsen, Regelzahlungsraten und fällige Gebühren in festen Bauspartarifen zusammengefasst, die von einer Kasse innerhalb eines Zeitintervalls auf dem Markt angeboten werden. Jeder Bausparvertrag ist zu einem festen Zeitpunkt genau einem Bauspartarif zugeordnet. Die Summe aller Bausparverträge einer Bausparkasse wird Bausparkollektiv genannt.

7.2 Der Bausparvertrag

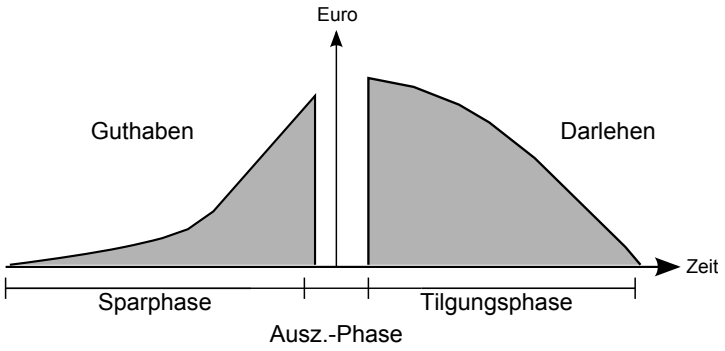
Nach §1 Absatz 2 des Bausparkassengesetzes ist ein Bausparvertrag definiert als ein Vertrag mit einer Bausparkasse, "durch den er (der Bausparer) nach

Leistung von Bauspareinlagen einen Rechtsanspruch auf Gewährung eines Bauspardarlehens erwirbt". Aus dieser Formulierung ist bereits ableitbar, dass ein Bausparvertrag während seiner Laufzeit mindestens zwei Phasen durchläuft: zunächst spart der Kunde ein Bausparguthaben an, um danach ein Bauspardarlehen zu tilgen.

In der Praxis können noch weitere Phasen im Leben eines Bausparvertrags definiert werden. Im Allgemeinen können folgende vier Phasen identifiziert werden:

1. Vertragsabschlussphase,
2. Sparphase,
3. Zuteilungs-/Auszahlungsphase,
4. Tilgungsphase.

Die Aufteilung des Vertragslebens in diese vier Phasen kann unter anderem in [Lau05], [Che05] und [Fak07] gefunden werden. Die Gewichtung der Lebensphasen eines Bausparvertrages ist je nach Fragestellung und Analyseziel unterschiedlich zu bewerten. So konzentrieren sich Kollektivprognosemodelle in der Regel auf die letzten drei Vertragsphasen, während die Anzahl und Verteilung von Vertragsabschlüssen von Fachexperten als Eingabegrößen manuell hinzugespielt werden. Jedoch ist im Rahmen von Kundenwertmodellen und Kundensegmentierungen die Betrachtung von Vertragsabschlüssen und potentiellen Folgeverträgen eines Kunden ein wichtiges Analysethema.



Die Lebensphasen eines Bausparvertrages

Weiterhin kann es sowohl in Simulationsmodellen als auch in Analysen hilfreich sein, die Sparphase in die Unterphasen der anfänglichen Sparphase und der Fortsetzungsphase aufzugliedern. Sobald der Bausparer genügend Sparleistungen erbracht hat, um seinen Rechtsanspruch auf ein Bauspardarlehen zu erlangen, kann er dieses Bauspardarlehen einfordern. Sollte er sich jedoch dagegen entscheiden, so kann er die Sparphase fortsetzen und zu einem späteren Zeitpunkt auf sein Darlehen zugreifen. Diese sogenannte Fortsetzung wird in der Regel als selbständiger Zustand betrachtet, da sie aus einer anderen Motivation heraus geschieht und aus diesem Grund zu anderen Verhaltensweisen seitens des Bausparers führt. Während der Bausparer in der anfänglichen Sparphase sparen

muss, um sein Darlehensziel zu erreichen, so muss er dies in der Fortsetzung nicht mehr. Tatsächlich wird sein Darlehen mit jeder weiteren Einzahlung um den Differenzbetrag gemindert, weswegen die meisten Bausparer ab Fortsetzungsbeginn die Sparzahlungen einstellen oder zumindest stark zurückfahren.

In den folgenden Abschnitten werden die einzelnen Phasen des Bausparvertrages näher beschrieben.

7.2.1 Vertragsabschlussphase

Möchte ein Kunde einen Bausparvertrag abschließen, so wird nach einer angemessenen Beratung der passende Bauspartarif aus dem bestehenden Angebot der Bausparkasse herausgesucht. Details zu dem Tarifangebot von Bausparkassen finden sich im nächsten Abschnitt.

Der Kunde kann hierbei entweder durch Selbstinitiative eine Bausparkasse aufsuchen oder wird durch gezielte Ansprache des Vertriebes für einen potentiellen Nachfolgevertrag aktiv beworben. Die Wahl von erfolgsversprechenden Kontaktaufnahmezeitpunkten ist ein wichtiger Untersuchungsgegenstand in der Aufstellung von Kundenwertmodellen und der Wahl von Kundensegmentierungen.

Festlegung der Bausparsumme

Im weiteren Verlauf werden ausschließlich Bauspartarife betrachtet, die eine feste Bausparsumme aufweisen. Gleichwohl gibt es in Theorie und Praxis Bauspartarife, die ohne die Festlegung einer Bausparsumme auskommen. Diese sind auf dem Markt aber Randerscheinungen.

Die Bausparsumme wird bei Vertragsabschluss festgelegt und entspricht der Summe aus dem angestrebten Guthaben und dem geplanten Darlehensbetrag in Euro. Beispiel: Der Kunde schließt einen Bausparvertrag über 20.000 Euro Bausparsumme ab und zahlt 8.000 Euro (inkl. Zinsen) ein. Nach positiver Bewertung und Zuteilung hat er einen Anspruch auf ein Darlehen in Höhe von 12.000 Euro. Vom Kunden wird an dieser Stelle eine gewisse Weitsicht verlangt. So sollte er bereits eine Vorstellung davon haben, welchen Betrag er nach Ende der Ansparzeit als Darlehen benötigt. Bauspartarife sehen aber in der Regel genügend Flexibilität vor, um die Bausparsumme in der Ansparphase zu erhöhen oder zu vermindern, sollten sich die Vorstellungen des Kunden ändern.

Vorfinanzierung

Nicht immer hat der Kunde die Möglichkeit seine Wohnfinanzierung langfristig abzuschätzen und rechtzeitig einen Bausparvertrag anzusparen. Im Falle des kurzfristigen Finanzierungsbedarfs kann ein Vorfinanzierungsmodell gewählt werden. Hierbei wird von der Kasse ein Vorfinanzierungskredit zu marktüblichen Konditionen gewährt und zeitgleich ein Bausparvertrag über gleiche

Höhe abgeschlossen. Der Kunde bedient hierbei die anfallenden Zinsen des Kredites und zahlt gleichzeitig einen Restbetrag auf das Bausparkonto ein. Sobald der Bausparvertrag zugeteilt wird, wird mit dem Auszahlungsbetrag der Vorfinanzierungskredit abgelöst. Der Kunde muss ab diesem Zeitpunkt nur noch den tariflichen Darlehenszins des Bausparvertrages leisten und kann die im Tarif vereinbarten Tilgungszahlungen bedienen.

Zwischenfinanzierung

Ähnliches Vorgehen ist bei einer Zwischenfinanzierung möglich. Besteht bereits ein Bausparvertrag, dessen Darlehen jedoch noch nicht abgerufen werden kann, so kann die Bausparkasse mit dem Kunden einen Zwischenfinanzierungskredit zu marktüblichen Konditionen abschließen. Der Kunde zahlt daraufhin die fälligen Kreditzinsen und bespart weiter seinen Bausparvertrag (sofern notwendig), bis es zur Guthabens- und Darlehensauszahlung des Bausparvertrages kommt. Der Kredit wird daraufhin abgelöst, während der Kunde nun seinen Bausparvertrag tilgen kann.

7.2.2 Sparphase

Sparverhalten

Nach Vertragsabschluss beginnt die Sparphase, in der der Bausparer seinen Bausparvertrag bespart. Obwohl viele Bauspartarife einen vorgeschlagenen Regelsparplan aufweisen, ist der Kunde in der Höhe und Zeitdauer seiner Besparung relativ frei. So kann er die erforderlichen Einlagen sofort nach Vertragsabschluss überweisen oder regelmäßige monatliche Zahlungen leisten. In finanziell schwierigen Zeiten kann er seinen Bausparvertrag ruhen lassen. Einige Tarife besitzen jährliche Sparobergrenzen, die nur in Absprache mit der Kasse überschritten werden dürfen. Vor- und Zwischenfinanzierungsmodelle bilden hier Sonderfälle, da der zugehörige Kredit eine vereinbarte Besparung des Bausparvertrages voraussetzt.

Bonusregelungen

Zusätzlich zu der klassischen Verzinsung des Guthabens kann die Bausparkasse in Renditetarifen die Sparleistung eines Kunden durch Gewährung von Bonuszinsen belohnen, die bei Guthabenauszahlung an den Kunden ausbezahlt werden.
Die Bonusmodularitäten sind auf dem Markt breit gefächert. Neben dem klassischen festen Bonuszins existieren zahlreiche Bedingungen an das Verhalten des Kunden, die zu Änderungen in der Laufzeit und der Höhe des Bonus führen können. Der Bonuszins ist häufig im Gegensatz zum Guthabenzins nicht über den gesamten Vertragsablauf fixiert und kann sich jährlich an aktuellen Marktzinsen ausrichten.

Kündigung

Der Kunde hat während der Sparphase jederzeit die Möglichkeit seinen Vertrag innerhalb der Kündigungsfrist zu kündigen. Je nach Kasse und Tarif kann die Kündigung Auswirkungen auf die Höhe von Bonusauszahlungen haben.

Änderungen der Bausparsumme

Der Kunde kann (in Absprache mit der Kasse) die Bausparsumme nachträglich erhöhen oder ermäßigen und somit Einfluss auf den Zeitpunkt der Zuteilung nehmen. Dies kann sinnvoll sein, wenn kurzfristiger Finanzierungsbedarf besteht und kein Zwischenfinanzierungskredit aufgenommen werden möchte. Bei einigen Sondermodellen (Abspaltungsmodelle) wird die Bausparsumme automatisch zu einem bestimmten Zeitpunkt vermindert, sodass das bisher eingezahlte Guthaben die Zuteilungsbedingungen erfüllt.

Tarifwechsel

Obwohl jeder Bausparvertrag an einem konkreten Tarif gebunden ist, ist es möglich, den Tarif zu wechseln. Manche Tariffamilien sind darauf ausgelegt, dass Bausparer sich während der Laufzeit noch für ein spezielleres Modell entscheiden können. Weiterhin besteht die Möglichkeit beim Start einer neuen Tarifgeneration alte Verträge an die neuen Tarife anzupassen, um so beispielsweise von einem besseren Zinsumfeld zu profitieren.

7.2.3 Fortsetzungsphase

Bewertungsstichtage

In regelmäßigen Abständen betreibt die Bausparkasse Bewertungsstichtage. Am Bewertungsstichtag werden alle Bausparverträge darauf überprüft, ob sie die Voraussetzung für eine Zuteilung erfüllen. Die Zuteilungsvoraussetzung der öffentlichen Bausparkassen beinhalten mindestens:

- den Mindestanspargrad: Jeder Bauspartarif definiert einen Mindestanspargrad, der normalerweise zwischen 40% und 50% der Bausparsumme liegt. Es werden nur Bausparverträge zugeteilt, deren Guthaben (inkl. Zinsen) diesen Mindestanspargrad überschreiten.
- die Mindestbewertungszahl: Jeder Bauspartarif definiert eine Mindestbewertungszahl, die der Vertrag erfüllen muss, bevor er zugeteilt werden kann. Die Bewertungszahl berechnet sich aus

$$BWZ = \frac{Guthaben + (Zinsen \cdot Zinsfaktor)}{Divisor \cdot Bausparsumme}.$$

  Zinsfaktor und Divisor werden für jeden Tarif im Vorfeld festgelegt, um eine Vergleichbarkeit der unterschiedlichen Tarife zu gewährleisten. Zusätzlich definiert die Bausparkasse je nach vorhandener Liquidität eine Ziel-

bewertungszahl, die für alle Verträge des Kollektivs überschritten werden muss.

- die Mindestwartezeit: Jeder Bauspartarif definiert eine Mindestwartezeit, die ein Vertrag vor Zuteilung einhalten muss.

In der privaten Bausparbranche sind die Zuteilungsmodalitäten wesentlich vielfältiger. Einige Bausparkassen führen sehr spezielle Zuteilungsmodelle, die ohne technische Hilfsmittel selbst für einen Einzelvertrag kaum zu überschauen sind.

Hat der Bausparer alle kassen- und tarifspezifischen Anforderungen an seine Sparleistung erfüllt, so kann der Bausparvertrag nach positiver Bewertung zugeteilt werden, wenn der Bausparer dies wünscht. Nach erfolgreicher positiver Bewertung vergehen noch einmal mehrere Monate, bevor es zur Zuteilung kommt.

Ist der Bausparer trotz positiver Bewertung nicht an einer Zuteilung interessiert, so kann er die Sparphase seines Vertrages fortsetzen. Hiermit kann er weiteres Guthaben auf seinem Vertrag ansammeln und somit das benötigte Darlehen verringern. Im Allgemeinen verliert der Bausparer während der Fortsetzung die positive Bewertung nicht, sodass er jederzeit im Rahmen der Bewertungsstichtage zugeteilt werden kann. Ausnahmen können bestehen, wenn es zu einer Erhöhung der Bausparsumme oder einem Tarifwechsel kommt. Durch eine Fortsetzung ist die Ansparung eines Guthabens von 100% der Bausparsumme oder mehr möglich. Die Bausparkasse hat mitunter die Möglichkeit, Kunden mit einem Anspargrad von über 100% zu kündigen.

7.2.4 Zuteilungs-/Auszahlungsphase

Guthabensauszahlung

Nach erfolgreicher Zuteilung wird dem Kunden das gesamte Guthaben inklusive aller Zinsen und Boni ausgezahlt. Die Auszahlung findet gewöhnlich zeitlich nahe zum Zuteilungstermin statt.

Darlehensauszahlung

Der Kunde kann flexibel entscheiden, ob und wann er von seinem erworbenen Rechtsanspruch auf Gewährung eines Darlehens Gebrauch macht und sich das Darlehen auszahlen lassen. Im Falle einer Darlehensauszahlung kontrolliert die Kasse, ob alle Anforderungen an ein Darlehen erfüllt sind. Das Zeitintervall zwischen positiver Bewertung und der letztlichen Darlehensauszahlung kann mehrere Monate bis Jahre betragen und ist sowohl vom Kunden wie auch der bearbeitenden Bausparkasse abhängig.

Darlehensverzicht

Sollte der Kunde auf sein Darlehen verzichten, so wird sein Vertrag nach Guthabenauszahlung geschlossen. Ein Darlehensverzicht kann eintreten, wenn der Kunde in der Zwischenzeit kein Darlehen mehr benötigt oder die Kreditzinsen auf dem Markt so attraktiv geworden sind, dass der Darlehenszins des Vertrages nicht mehr attraktiv erscheint.

7.2.5 Tilgungsphase

Nach Darlehensauszahlung beginnt der Kunde mit der Rückzahlung des Darlehens. Die Tilgungsoptionen sind ähnlich flexibel wie die Sparoptionen. Zwar ist der Bausparer verpflichtet, monatlich den tariflichen Mindesttilgungsbeitrag zu leisten, jedoch kann er auch höhere Tilgungsraten (Sonderzahlungen) bedienen, um Darlehenszinsen zu sparen und das Darlehen früher abzulösen.

Vollablösungen von Darlehen

Ein Sonderfall liegt vor, wenn sich der Kunde entscheidet, das gesamte Restdarlehen mit einer Zahlung zurückzuzahlen. Bausparkassen müssen diese Vollablösungen im Auge behalten, da hierdurch ein Risiko von plötzlich wegfallenden Zinseinnahmen für die Kasse entstehen kann.

7.2.6 Zusammenfassung

Interpretiert man einen Bausparvertrag als zeitabhängige Folge von Optionsereignissen seitens des Kunden und der Bausparkasse, so bleibt festzustellen, dass sowohl die Laufzeit wie auch die getätigten Optionen von Vertrag zu Vertrag stark abweichen können. So ist es selbst innerhalb eines identischen Bauspartarifs nicht unüblich, dass die Vertragslaufzeiten der Verträge zwischen 2 und über 30 Jahren variieren. In dieser Zeitspanne werden feste Zinsvereinbarungen für Guthaben und Darlehen geleistet, die bereits zu Vertragsabschluss festgeschrieben wurden. Aus diesem Grund ist es für jede Bausparkasse aus betriebswirtschaftlichen Gründen wichtig, möglichst präzise Informationen über die Struktur und prognostizierte Abwicklung des Bausparkollektivs zu gewinnen. Unverzichtbares Hilfsmittel sind hierbei statistische Analysen und Data-Mining-Verfahren, um Muster und Wechselwirkungen im Kollektiv aufzuspüren. Erkenntnisgewinne können daraufhin in Kollektivsimulationen umgesetzt und angewandt werden.

7.3 Bauspartarif

Ein Bauspartarif fasst alle rechtlichen und monetären Wesensmerkmale des Bausparprozesses zusammen und gibt Auskunft über mögliche Optionen oder Sonderkonditionen, die ihn von anderen Bauspartarifen unterscheiden. Im Allgemeinen bietet eine Bausparkasse mehrere aktuelle Bauspartarife einer Generation an,

die teilweise hierarchisch strukturiert die Bedürfnisse der Kunden abdecken sollen. Die Aufteilung und Benennung der Tarife kann je nach Bausparkasse sehr unterschiedlich sein. Grob unterteilt lassen sich jedoch vier unterschiedliche Tarifgruppen beschreiben.

7.3.1 Wohnungsbaufinanzierung

Das klassische Bauspargeschäft der Bausparkassen besteht aus Tarifen, die darauf ausgelegt sind, größere Summen zum Bau oder Kauf eines Hauses oder einer Eigentumswohnung zu finanzieren. Unter der Berücksichtigung des hohen Finanzierungsbedarfs und der individuellen Finanzierungsmöglichkeiten der Kunden werden häufig mehrere Varianten angeboten, die sich in den regelmäßigen Spar- und Tilgungsraten unterscheiden und einen an die Laufzeit angepassten Darlehenszins aufweisen.

7.3.2 Wohnmodernisierung

Modernisierungstarife sind darauf ausgelegt, dass kleinere Finanzierungssummen aufgenommen werden, die schneller angespart und zügig zurückgezahlt werden. Gelegentlich wird hierbei auch ein höherer Eigenanteil des Kunden eingefordert.

7.3.3 Vorsorge und Rendite

Bei Renditetarifen steht der Vorsorge- und Sparwunsch im Vordergrund. Hier finden sich Kunden, die in erster Linie ihre Sparbeiträge, staatliche Leistungen oder vermögenswirksame Leistungen vom Arbeitgeber sichern möchten, aber sich dennoch die Möglichkeit einer späteren Finanzierung offenhalten möchten. Da die Guthabenzinsen in den Tarifen entsprechend höher sind, werden später ebenso höhere Darlehenszinsen anfallen.

7.3.4 Eigenheimrente

Seit 2008 bieten Bausparkassen sogenannte Eigenheim- oder Riestertarife an, die staatlich gefördert werden. Aufgrund der hohen staatlichen Anforderungen an Bausparkasse und Bausparer nehmen diese Tarife eine Sonderrolle ein.

Gehört der Bausparer zu einer förderungsberechtigten Gruppe und zahlt einen gewissen monatlichen Mindestsparbeitrag, der sich an den monatlichen Einkünften des Sparers orientiert, so erhält er einen staatlichen Zuschuss, der von der Ehesituation und der Anzahl der Kinder des Bausparers abhängt. Die Bausparkasse hingegen verpflichtet sich zu Garantien an den Bausparer. So kann sie Riesterverträge nicht ohne Weiteres kündigen und muss mit Erreichen der Zuteilung im Rahmen einer Beitragsgarantie sicherstellen, dass der Bausparer mindestens die von ihm eingezahlten Sparbeiträge zurückerhält. Entscheidet sich der Bausparer gegen ein Darlehen, so muss die Kasse ihm die Möglichkeit

einer lebenslangen Riesterrente gewähren, die jedoch bei geringen Einzahlungen in Form einer Einmalauszahlung geleistet werden kann.

7.4 Die Kollektivsicht

Die Gesamtheit aller Bausparverträge einer Bausparkasse wird Bausparkollektiv genannt. Je nach Betrachtungsebene kann es sinnvoll sein, das Kollektiv als eine Menge von Bausparverträgen oder eine Menge von Kunden zu interpretieren.

7.4.1 Vertragsebene

Die Kollektivsicht auf Vertragsebene ermöglicht die effiziente Erfassung von Aggregationen, statistischen Auswertungen und die Definition und Ermittlung von Kollektivkennzahlen, die eine Übersicht über die Struktur und die betriebswirtschaftlichen Risiken des Kollektivs geben. Unter diese Kennzahlen fallen monetäre Größen wie

- Neugeschäftsvolumen,
- Bauspareinlagen,
- Zuteilungen,
- Bauspardarlehen,
- Tilgungsbeiträge.

Weiterhin werden abgeleitete Größen wie

- Anlegungsgrad,
- Trägheitsreserve,
- Sparintensitäten,
- Kündigungsquoten

ermittelbar. Besonders interessant werden Analysen, wenn die Kenngrößen über mehrere Kalenderjahre oder Kalenderquartale im Zeitverlauf betrachtet werden können. Die Vorhersage dieser Zeitverläufe ist ein wichtiges Ziel von Kollektivsimulationen.

7.4.2 Kundenebene

Die Kollektivsicht auf Kundenebene stellt den Kunden in den Mittelpunkt der Analysen. Der Kunde ist hierbei eine juristische oder natürliche Person, die Ausprägungen zu Attributen wie

- Alter,

- Berufsgruppe,
- Familienstand,
- Wohnort

aufweisen kann. Mitunter steht er durch einen Eheverbund oder eine Personengemeinschaft innerhalb des Kollektivs noch in Kontakt zu weiteren Personen. Jeder Kunde kann mehrere parallel oder seriell laufende Bausparverträge besitzen. Die Anzahl der Verträge und der Verlauf der bisherigen Verträge können Rückschlüsse auf das zukünftige Abschluss- und Ansparverhalten zulassen. Diese Informationen können genutzt werden, um die Qualität von Kollektivsimulationen zu verbessern. Zusätzlich geben sie die Möglichkeit, Kundenwertmodelle und damit Kundensegmentierungsmodelle aufzustellen.

7.4.3 Gegensteuerungsmaßnahmen der Bausparkassen

Die Bausparkasse muss wie jedes Unternehmen die betriebswirtschaftliche Lauffähigkeit und die notwendige Liquidität für ihre Kollektivabläufe sicherstellen. Sie steht deswegen unter ständiger Aufsicht der Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin), die beispielsweise neue Bauspartarife vor Markteinführung in einem Genehmigungsprozess kontrolliert.
Tritt ein Ungleichgewicht im Kollektiv oder im Marktzinsumfeld ein, welches eine Gefahr für die Funktionalität des Bausparmodells darstellt, so stehen der Kasse eine Vielzahl von Steuerungsmöglichkeiten zur Verfügung.

Wahl der Zielbewertungszahl

Reichen die Einlagen der Bausparkasse nicht aus, um die möglichen Ansprüche an Darlehensauszahlungen zu bedienen, so kann die Kasse die Zielbewertungszahl erhöhen und somit direkt den Zuteilungsprozess beeinflussen. Für den Kunden wirkt sich dies in einer Erhöhung der Wartezeit bis zur Zuteilung aus. Tatsächlich sind sämtliche berechneten Zuteilungstermine an den Kunden nur Prognosen. Garantierte Zuteilungsaussagen dürfen nicht getroffen werden. Der Kunde kann die Wartezeit jedoch mit einer Zwischenfinanzierung überbrücken.

Einführung neuer Tarife

Die Bausparkasse kann jederzeit bestehende Tarife vom Markt nehmen, wenn sie der Meinung ist, dass sie sich betriebswirtschaftlich nicht mehr lohnen. Bestehende Verträge müssen allerdings weiter in diesem Tarif geführt werden, es sei denn, die Kasse hat in die ABB geschrieben, dass in diesem Fall ein Tarifwechsel in einen marktoffenen Tarif notwendig wird.

Da Bauspartarife in einem Genehmigungsprozess von der BaFin zugelassen werden müssen und sich dieser Prozess über mehrere Monate hinziehen kann, muss die Planung von neuen Tarifen frühzeitig in Angriff genommen werden. Oftmals hat sich die Situation auf dem Markt und im Kollektiv bis zur

Einführung der neuen Tarife schon wieder verändert.
Aus diesem Grund ist es wesentlich und von der BaFin als zwingend vorausgesetzt, regelmäßige Kollektivsimulationen durchzuführen und hierbei auch den Einfluss von neuen Tarifen mit abzubilden.

7.5 Motivation der weiteren Kapitel

Unter Berücksichtigung diverser marktregulativer Vorschriften wie Basel II und III sowie MaRisk kommt der Auswertung eines Bausparkollektivs besondere Bedeutung zu. Bausparkassen nutzen vielseitige Analyseverfahren, um die aktuelle und zukünftige Zusammensetzung der Bausparkollektive zu bewerten. Die vorliegende Arbeit befasst sich mit drei wichtigen Analysegebieten.

7.5.1 Data-Mining

Unter Data-Mining wird die Analyse von Daten innerhalb des Bausparkollektivs beschrieben, die das Ziel verfolgen, qualitativ neuwertige Informationen über die Zusammensetzung und die Interaktionen innerhalb des Datensatzes zu generieren. Im Gegensatz zu einer gewöhnlichen Auswertung, in der monetäre Kennziffern ermittelt werden, besteht das Hauptaugenmerk in der Schaffung von strukturellen Erkenntnissen wie: 'Kunden mit einem Renditevertrag schließen im Alter zwischen 20 und 30 Jahren mit hoher Wahrscheinlichkeit einen Finanzierungsvertrag ab'.

7.5.2 Kollektivsimulationen

In Kollektivsimulationen wird anhand einer Vergangenheitsauswertung der aktuelle Bestand eines Bausparkollektivs in die Zukunft fortgeschrieben. Je nach genutztem Modell und Techniken können hierbei qualitativ messbare Aussagen über kurzfristige oder langfristige Änderungen in der Kollektivzusammensetzung getätigt werden. Unter Berücksichtigung von Expertenzugaben wie Neugeschäftsverteilungen und Marktzinsszenarien lassen sich hiermit zukünftig mögliche Liquiditätsengpässe frühzeitig erkennen und verhindern.

7.5.3 Individualprognosen

Individualprognosen stellen die korrekte Vorhersage des Verhaltens individueller Kunden oder Verträge in den Vordergrund. Individualprognosen können für Scoring-Ansätze oder Kundenwertmodelle genutzt werden und ermöglichen die gezielte Ansprache und Betreuung von Kundengruppen, für die beispielsweise eine hohe Wahrscheinlichkeit auf den Abschluss eines Folgevertrags oder einen Kundenverlust ermittelt wurde.

Kapitel 8

Häufige Mustererkennung in Bausparkollektiven

8.1 Data-Mining und der KDD-Prozess

Die elektronische Datenverarbeitung ist in modernen Unternehmen ein unverzichtbares Hilfsmittel geworden. Bausparkassen bilden hier keine Ausnahme. Im Jahr 2018 können alle öffentlichen deutschen Bausparkassen auf eine standardisierte, digitale Historie von mindestens zehn Jahren zurückgreifen. Neben den alltäglichen betriebswirtschaftlichen Erfordernissen ermöglicht die Digitalisierung, neue wissenschaftliche und praxisrelevante Erkenntnisse aus der Vielzahl der Daten zu gewinnen. Diese Aufgabe zur Schaffung qualitativ neuwertiger Erkenntnisse aus großen Datensätzen wird allgemein *Data-Mining* genannt. Der Gesamtprozess von der Bereitstellung der Daten bis zur Verifizierung der Data-Mining Ergebnisse wird unter dem Begriff *Knowledge Discovery in Databases* (KDD) zusammengefasst [Bra07].

Die einzelnen Phasen des KDD-Prozesses sind mittlerweile größtenteils standardisiert und unterscheiden sich je nach Literaturquelle hauptsächlich im Detailgrad oder der Reihenfolge der Zwischenschritte. Allen Prozessformulierungen ist gemein, dass der KDD Prozess in drei grobe Phasen unterteilt wird. Diese Phasen sind

1. Datenaufbereitung,
2. Data-Mining,
3. Verifizierung/Evaluierung.

Der erste Schritt der Datenaufbereitung ist nicht eindeutig spezifiziert. In [Bra07] wird die Datenaufbereitung in die Teilschritte Integration, Selektion und Preprocessing zerlegt, während in [FPSS96] die Aufteilung in Selektion, Preprocessing und Transformation untergliedert wird.

Dieses Kapitel beschreibt im Folgenden die Phase der Datenaufbereitung.

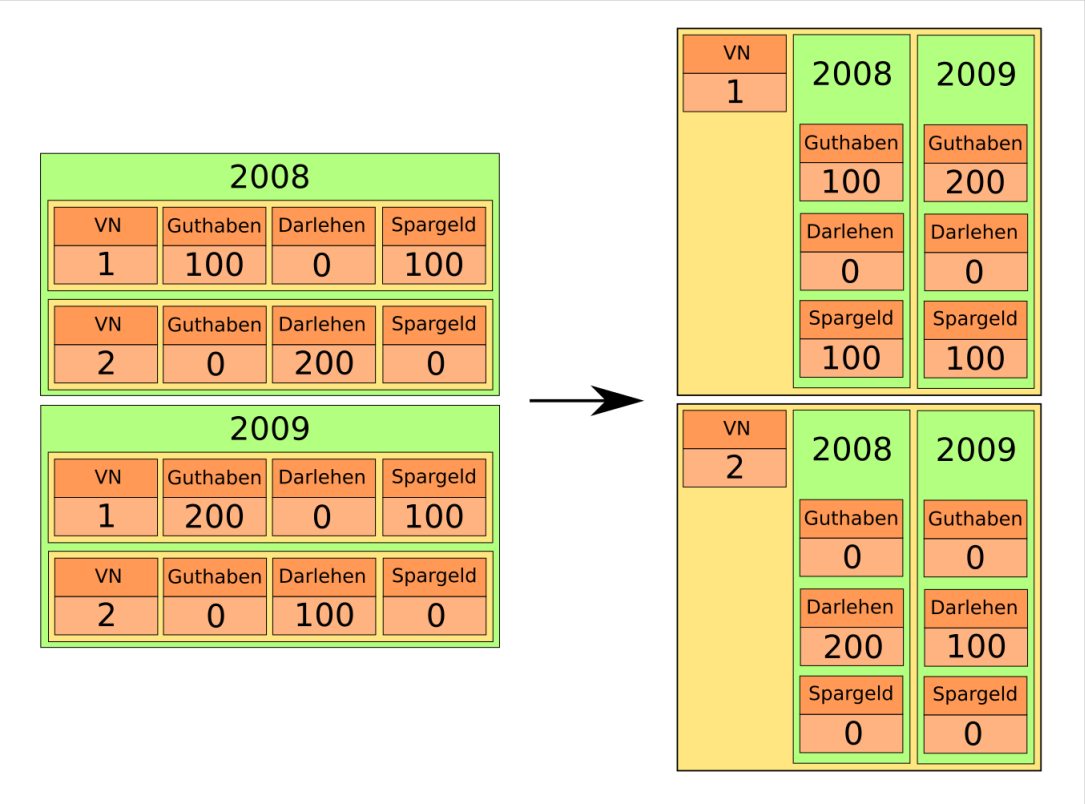
Die Phasen Data-Mining und Verifizierung werden in den weiteren Kapiteln behandelt.

8.2 Datenaufbereitung der Grunddaten

Die in dieser Arbeit genutzten Daten wurden von den öffentlichen deutschen Bausparkassen der Universität zu Köln zur Verfügung gestellt und stammen aus mehreren unabhängigen Datenquellen. Die Universität zu Köln und die öffentlichen deutschen Bausparkassen pflegten eine langjährige Forschungskooperation, die über mehr als 25 Jahre Grundlage zahlreicher Veröffentlichungen war ([AB88] [Kel92] [BBDW94] [Mer95] [Van96] [Bac97] [Che05] [Fak07] [Stu13]) und zur Schaffung von Programmen zur Kollektivsteuerung maßgeblich beigetragen hat. Diese Zusammenarbeit lief Ende 2017 aus. Die intensive Forschungskooperation ermöglichte, dass die Datenqualität der Eingangsdaten auf einem sehr hohen Niveau angelangt war.

Jede Bausparkasse lieferte jährlich oder quartalsweise den aktuellen Stand ihres Kollektivs in Form von anonymisierten Vertragsdaten an die Universität zu Köln. Die Daten wurden hierbei den jeweiligen Data-Warehouses der Kassen entnommen, die ihrerseits bereits einer Qualitätssicherung unterliegen. Da es sich nur um einen Teil der im Data-Warehouse vorhandenen Daten handelt, die für Kollektivsimulationen erforderlich sind, fand bereits eine Vorselektion der Daten statt.

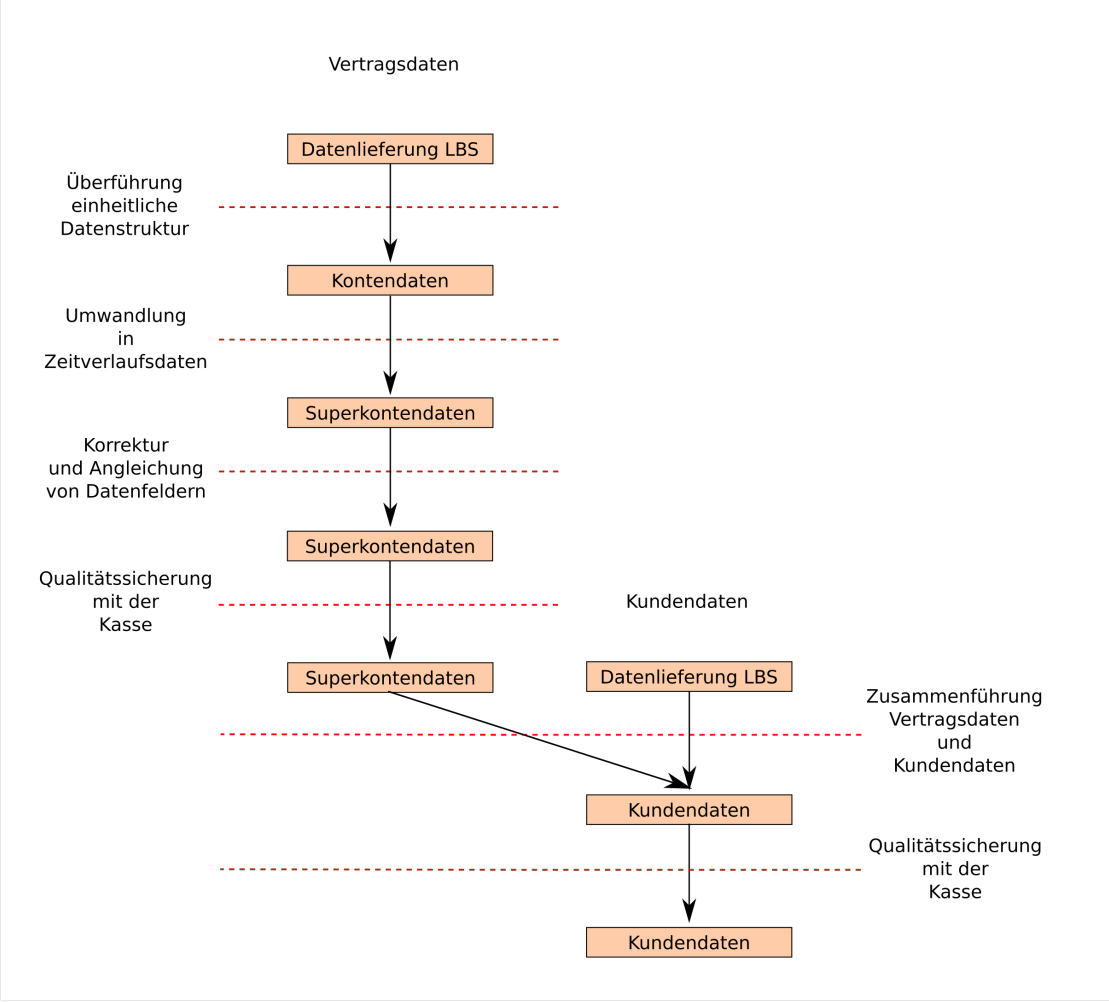
Die Daten der einzelnen Jahre bzw. Quartale wurden an der Universität zu Köln eingelesen und in Zeitverlaufsdaten gespeichert. Hierbei werden Jahresdaten auf Vertragsebene zu Vertragsdaten auf Jahresebene transformiert. Auf diese Weise ist es möglich, einzelne Bausparverträge als sequenzielle Objekte zu betrachten.



Überführung der Kontendaten zu sogenannten Superkontendaten

Die entstandenen Zeitverlaufsdaten wurden nun noch einmal durch eine Qualitätssicherung auf bekannte Definitionsunterschiede innerhalb der Kassen (beispielsweise Auszahlungsbeträge ohne oder mit Boni) geprüft und gegebenenfalls korrigiert.

In dieser Arbeit werden die zu Simulationszwecken generierten Zeitverlaufsdaten mit weiterführenden Kundenmerkmalen zusammengeführt, die der Universität zu Köln von Probekassen zur Verfügung gestellt wurden. Die neuen Daten ermöglichen die Gruppierung von Verträgen zu Vertragsgruppen, die von identischen Kunden geführt werden. Hierdurch können erstmals Aussagen über das Verhalten von Bausparern gemacht werden, die über Vertragsgrenzen hinausgehen. Sämtliche genutzten Attribute wurden in einem Qualitätssicherungsschritt mit der jeweiligen Kasse auf Konsistenz überprüft.



Überführung der Superkontendaten zu Zeitverlaufsdaten auf Kundenebene

8.3 Modellierung eines Bausparkollektivs als sequenzielle Datenbank

Die Daten, die der Universität zu Köln zur Verfügung gestellt wurden, beinhalten Bestands- und Flussgrößen der wichtigsten Geldbewegungen innerhalb eines Vertrages. So werden beispielsweise für jedes Datenjahr der geleistete Spargeldeingang oder die geleisteten Darlehensauszahlungen erfasst. Diese Kenngrößen sind in den meisten Fällen Euro-Beträge und können damit als kontinuierliche Größen aufgefasst werden.

In der Praxis der Kollektivprognose hat sich herauskristallisiert, dass eine Vorhersage des Vertragsverlaufs im Kern auf eine Prognose von Ereignissen hinausläuft, die der Vertrag im Laufe des Vertragslebens durchläuft. So kann der Abfluss des gesamten Guthabens in der Sparphase darauf zurückgeführt werden, dass ein Kündigungsereignis stattgefunden hat. Dieses Ereignis wurde entweder vom Inhaber des Vertrages oder der jeweiligen Bausparkasse in Gang gesetzt. Die monetären Abflüsse in den Vertragsdaten sind nur Folgereaktionen

aus diesem abstrakten Ereignis.

Jeder Bausparvertrag kann folglich in einem ersten Data-Mining-Schritt in eine Folge von Ereignissen transformiert werden. Bei dieser Abstraktion und Transformation gehen in der Regel Informationen verloren, die in einem späteren Schritt der Prognose wieder hinzugefügt werden müssen. So werden die getätigten Spargeldeinzahlungen für die Ereigniserfassung nicht centgenau erfasst sondern in prototypischen Sparverläufen (Prototypen) wie Soforteinzahlung oder Regelbesparung einsortiert. Da bei der späteren Prognose jedoch der Guthabenstand des Vertrages zu Prognosebeginn bekannt ist, kann hier ohne Bestandsverlust weitergearbeitet werden.

8.3.1 Auflistung der genutzten Ereignisse

Unter Berücksichtigung der Definition einer sequenziellen Datenbank aus Kapitel 4 wird eine Produktmenge P definiert, die alle möglichen Ereignisse im Leben eines Bausparvertrages enthält. Jeder Bausparvertrag wird dementsprechend als Sequenz $S_i =< s_1, \ldots, s_l >$ von Ereignissen erfasst. Aus der langjährigen Erfahrung mit Simulationen in Bausparkollektiven ist bekannt, welche Ereignisse für die Modellierung des Verlaufs eines Bausparvertrages wesentlich sind.

Ereignisse im Leben eines Bausparvertrages		
Ereignis	Abkürzung	Beschreibung
Abschluss	ABS	Vertragsabschluss getätigt
Soforteinzahlung	SP1	Sparverhalten: Soforteinzahlung nach Abschluss
Starke Regelbesparung	SP2	Sparverhalten: überdurchschnitt. Regelbesparung
Regelbesparung	SP3	Sparverhalten: Regelbesparung
Geringbesparung	SP4	Sparverhalten: weniger als Regelbesparung
Erhöhung	ERH	Bausparsumme wird im Vertragsverlauf erhöht
Ermäßigung	ERM	Bausparsumme wird im Vertragsverlauf ermäßigt
Kündigung	KUE	Vertrag wird vor Zuteilung gekündigt
Tarifwechsel	TAW	Vertrag wechselt in einen anderen Tarif
positive Bewertung	PBW	Vertrag erreicht die Zuteilungsbedingungen
Fortsetzungsanfang	FOA	Vertrag nimmt Zuteilung nicht an
Fortsetzungsende	FOE	Vertrag nimmt Zuteilung nach Fortsetzung an
Zuteilung	ZUT	Vertrag wird zugeteilt
Guthabenauszahlung	GAU	Guthabenabfluss nach Zuteilung
Darlehensverzicht	DVE	Vertrag verzichtet auf Darlehensauszahlung
Darlehensauszahlung	DAU	Darlehen wird ausgezahlt
Regeltilgung	RTI	Darlehen wird nach Tarifvorgaben zurückgezahlt
Vollablösung	VTI	Restdarlehen wird mit einer Zahlung zurückgezahlt

Tabelle 8.1: Abgebildete Ereignisse zur Modellierung des Lebenslaufs eines Bausparvertrages

Die obige Tabelle gilt nur als generelle Leitlinie möglicher Ereignisse. Je nach konkreter Aufgabenstellung kann es sinnvoll sein, die Ereignisse um unnötige Einträge zu reduzieren oder um detaillierte Ereignisse zu ergänzen. Beispielsweise kann es für eine Fragestellung wesentlich sein, ob der Vertrag in den ersten drei Vertragsjahren gekündigt wurde oder nach mehr als sieben Jahren. In

diesem Fall kann das Kündigungsereignis auf mehrere Zeitintervalle ausgedehnt werden.

Als Beispiel sei an dieser Stelle folgender Vertrag als Sequenz gegeben:

$$S =< (ABS, SP3), (PBW, ZUT), (GAU, DAU), VTI > .$$

Der Vertrag spart nach Abschluss mit einer Regelbesparung und lässt sich nach Erreichen der Zuteilungsbedingungen im gleichen Jahr zuteilen. Nach der Auszahlung von Guthaben und Darlehen zahlt er das Darlehen sofort vollständig zurück.

Gelegentlich kann es sinnvoll sein, genauere Zeitinformationen über die Ereignisse zu erhalten. In diesem Fall können weitere Ereignisse für das betrachtete Vertragsjahr hinzugefügt werden. Im Folgenden wird mit Vn das n.te Vertragsjahr bezeichnet. Gleicher Vertrag entspräche daraufhin der Sequenz

$$S =< (V1, ABS, SP3), (V7, PBW, ZUT), (V8, GAU, DAU), (V9, VTI) > .$$

Das volle Potential entfaltet die Modellierung eines Bausparvertrages als Sequenz von Ereignissen erst, sobald weitere Informationen im SMMP-Modell zugespielt werden und Aussagen möglich werden, die mehrere Bausparverträge miteinander verknüpfen.

8.4 Modellierung eines Bausparkollektivs als sequenzielle Phasendatenbank

In Kapitel 7 wurde detailliert auf das Wesen eines Bausparvertrages eingegangen und es wurden die verschiedenen Lebensphasen eines Bausparvertrages vorgestellt. Jede Phase kann hierbei, wie in Abschnitt 8.3 erläutert, als eine Folge von Ereignissen aufgefasst werden. Kombiniert man beide Erkenntnisse, kann darauf aufbauend ein Modell einer sequenziellen Phasendatenbank konstruiert werden, wie sie ausführlich in Kapitel 5 behandelt wurde.

Zur Modellierung wird die Phasenabbildung $\phi : P \longrightarrow \{1, \ldots, 5\}$ definiert, die jedem Ereignis aus P genau eine der Phasen:

1. Vertragsabschlussphase,
2. Sparphase,
3. Fortsetzungsphase,
4. Auszahlungsphase,
5. Tilgungsphase

zuordnet. Die Einteilung der Ereignisse in Phasen kann in folgender Tabelle abgelesen werden.

Sortierung von Ereignissen zu Phasen			
Phase 2	Phase 3	Phase 4	Phase 5
Soforteinzahlung	positive Bewertung	Zuteilung	Regeltilgung
Starke Regelbesparung	Fortsetzungsanfang	Guthabenauszahlung	Vollablösung
Regelbesparung	Fortsetzungsende	Darlehensverzicht	
Geringbesparung		Darlehensauszahlung	
Erhöhung	Erhöhung 2		
Ermäßigung	Ermäßigung 2		
Kündigung	Kündigung 2		
Tarifwechsel	Tarifwechsel 2		

Tabelle 8.2: Übersicht der Phasen eines Bausparvertrages mitsamt ihrer Ereignisse

Zur Einteilung der Ereignisse in Phasen seien folgende Beobachtungen mitgegeben.

- Die Vertragsphase 1 (Vertragsabschlussphase) wird gesondert im nächsten Abschnitt modelliert. In ihr finden sich sequenzunabhängige Attribute einer mehrdimensionalen Datenbank, die beispielsweise die anfängliche Höhe der Bausparsumme oder das Abschlussjahr umfassen.
- Einige Ereignisse können in mehreren Phasen auftauchen. Hierunter fallen vor allem die Sparphase und die Fortsetzungsphase. Diese Ereignisse müssen für eine Phasenuntersuchung dupliziert und als unabhängige Ereignisse mitgeführt werden. Dieses Verfahren ist allerdings durchaus gewünscht, da hinter den Entscheidungen pro Phase unterschiedliche Motivationen stecken können. Ausführungen hierzu können in Abschnitt 7.2 gefunden werden.
- In der Praxis kann es in seltenen Fällen vorkommen, dass ein Vertrag von der Fortsetzungsphase für längere Zeit zurück in die Sparphase fällt. Dies geschieht im Allgemeinen, wenn der Vertrag durch einen Tarifwechsel oder eine Erhöhung der Bausparsumme die Bewertungskriterien des Tarifs nicht mehr erreicht. Die Modellierung einer Phasendatenbank lässt diesen Rückfall nicht zu, weswegen solche Verträge bei einer Analyse ausgeklammert werden müssen.

8.5 Modellierung eines Bausparkollektivs als SMMP-Modell

Das SMMP-Modell wurde ausführlich in Kapitel 6 vorgestellt. Es setzt auf einer sequenziellen Phasendatenbank auf und erweitert sie um mehrstufige und multidimensionale Daten.

Im Rahmen eines Bausparkollektivs kann durch das SMMP-Modell ein Kollektiv von Kunden gebildet werden, die jeweils eine Reihe von Bausparverträgen besitzen können. Die Bausparverträge sind wiederum als sequenzielle Phasendatenbank von Ereignissen abgebildet.

8.5.1 Vertragsebene

Im letzten Abschnitt wurde die Modellierung eines Bausparvertrages als sequenzielle Phasendatenbank von Ereignissen behandelt. Jedes Ereignis ist hierbei eine der Phasen 2 (Sparphase) bis 5 (Tilgungsphase) zugeordnet.

Die erste Phase (Vertragsabschlussphase) wird im SMMP-Modell als zeitunabhängige multidimensionale Datenbank modelliert. In dieser Datenbank befinden sich Charakteristika des Vertrages, welche vom Vertragsablauf in der Regel unabhängig sind. In der folgenden Tabelle sind mögliche Datenfelder aufgelistet. Die Anzahl und Auflösung der einzelnen Attribute ist von der konkreten Fragestellung abhängig.

Zeitunabhängige Attribute eines Bausparvertrages		
Attribut	Abkürzung	Beschreibung
BS-Höhe	BS	Höhe der Bausparsumme zu Vertragsbeginn
Tarifart	TAA	Tarifart des Bausparvertrages (z.B. Finanzierer)
Tarif	TAR	Tarif der Bausparkasse bei Vertragsabschluss
Vertragsart	ART	gewöhnlicher Vertrag, VK, ZK, TBV ...
Abschlussjahr	ABS	absolut oder relativ zu Erstvertrag oder Alter des Kunden
VL-Bezug	VL	Bezug von vermögenswirksamen Leistungen
Zulagen-Bezug	RZU	Bezug von staatlichen Riesterzulagen
..		

Tabelle 8.3: Auflistung zeitunabhängiger Attribute eines Bausparvertrags

Die zeitunabhängigen Informationen des Bausparvertrages S_i werden als erste Phase an den Beginn der entsprechenden Sequenz gehängt. Hierbei wird wie in Abschnitt 6.4 erläutert die Attributausprägung $A_i^S \subseteq A_1^S \times \ldots \times A_k^S$ als Produktmenge aufgefasst und die allgemeine Grundmenge an Produkten um alle möglichen Ausprägungen aller Attribute erweitert.

8.5.2 Kundenebene

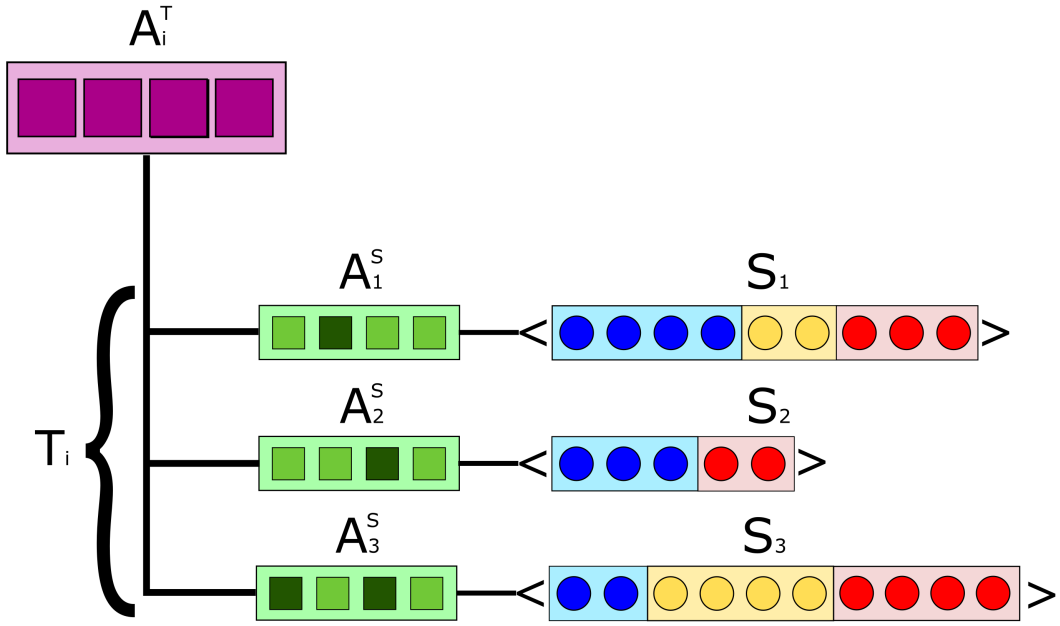
In der vorliegenden Arbeit wird im Rahmen der Studien an der Universität zu Köln erstmalig ein vollwertiger Kundenansatz verfolgt. Hierzu werden entgegen der bisherigen Praxis nicht mehr unabhängige Bausparverträge betrachtet. Stattdessen werden Bausparverträge mit gleichem Eigentümer als Gesamtheit aufgefasst und einem Kunden zugeordnet. Bei der Überführung von Vertragsebene auf Kundenebene gibt es einige Zusatzinformationen zu beachten.

Verknüpfung mehrerer Bausparverträge

Ein Kunde besitzt eine endliche Anzahl von Bausparverträgen, die er im Laufe seiner Beziehung mit der Bausparkasse abschließt und möglicherweise parallel fortführt. Insbesondere ist es üblich einen neuen Bausparvertrag abzuschließen, wenn der vorherige Bausparvertrag eine spätere Phase erreicht. Im letzten Abschnitt wurden bereits alle Bausparverträge als sequenzielle Phasendatenbank modelliert. Diese Verträge können nun wie in 6.4 besprochen als gemeinsame mehrstufige Sequenz abgebildet werden.

Betrachtung zusätzlicher Kundenattribute

Wie bereits im Falle von zeitunabhängigen Vertragsattributen können ebenso zeitunabhängige Kundenattribute definiert werden. Diese Kundenattribute ermöglichen eine genauere Selektion von Kundengruppen. So kann beispielsweise das Alter des Kunden bei Abschluss des ersten Vertrages Informationen über die möglichen Folgeverträge liefern, da Finanzierungswünsche in der Regel stark an die finanziellen Möglichkeiten des Kunden gebunden sind, die wiederum vom Alter geprägt sind.



Visualisierung der Struktur der Daten auf Kundenebene im SMMP-Modell

In der folgenden Tabelle sind mögliche Datenfelder aufgelistet. Die Anzahl und Auflösung der einzelnen Attribute ist von der konkreten Fragestellung abhängig.

Zeitunabhängige Attribute eines Kunden		
Attribut	Abkürzung	Beschreibung
Kundentyp	TYP	natürliche oder juristische Person
Alter	ALT	Alter bei Erstvertragsabschluss
Berufsgruppe	BER	Berufsgruppe des Kunden (z.B. Auszubildender)
Einkommen	EIN	durchschn. Einkommen des Kunden
Vertriebsart	VER	Betreuung des Kunden durch Außendienst/Drittleister
Familienstand	FAM	Familienstand des Kunden (ledig/verwitwet/...)
ABO-Bezieher	ABO	Erhält der Kunde die regelmäßige Zeitschrift der Kasse?
BS-Höhe	BSG	Gesamthöhe der Bausparsumme aller Verträge
...		

Tabelle 8.4: Auflistung zeitunabhängiger Attribute eines Kunden

8.6 Fazit

In diesem Kapitel wurde beschrieben, auf welchem Wege die zur Verfügung gestellten anonymisierten Daten der öffentlichen Bausparkassen in der Universität zu Köln weiterarbeitet wurden. Für diese Arbeit wurden erstmalig erweiterte Daten hinzugespielt, die eine Sicht auf Kundenebene ermöglichten. Alle Datenquellen wurden anschließend in ein sequenzielles Phasendatenbankmodell bzw-in ein SMMP-Modell überführt.

In den weiteren Kapiteln werden die konstruierten Modelle mit den entsprechenden Verfahren und Algorithmen aus den Kapiteln 5 und 6 bearbeitet und die Ergebnisse in Hinblick auf Verbesserungen in Kollektivsimulationen oder Individualprognosen vorgestellt.

Kapitel 9

Häufige Mustererkennung zur Verbesserung von Kollektivsimulationen

9.1 Simulation von Bausparkollektiven

Die Verwaltung und betriebswirtschaftliche Prognose der Aktivitäten von Millionen von Bausparverträgen stellt hohe Anforderungen an die EDV-Kapazitäten von Bausparkassen. Insbesondere die weitreichenden Optionsmöglichkeiten eines Bausparers, die nicht nur durch kollektivinterne Strukturen sondern auch durch marktbedingte exogene Faktoren beeinflusst werden, haben in den letzten Jahrzehnten zur Schaffung komplexer Kollektivsimulationsansätzen geführt.

9.1.1 Das NBI-Modell der öffentlichen Bausparkassen

An der Universität zu Köln wurde seit den 80er Jahren des letzten Jahrhunderts das Softwaresystem NBI [1] für Kollektivsimulationen entwickelt, welches aktuell in allen öffentlichen Bausparkassen eingesetzt wird (siehe [AB88], [Kel92], [BBDW94], [Mer95], [Van96], [Bac97], [Che05], [Fak07]). Neben jährlichen Neuauslieferungen mit aktuellen Bestandsdaten und der Implementierung neuer Funktionen wird das NBI in regelmäßigen Abständen von Wirtschaftsprüfern zertifiziert.

Das NBI basiert im Kern auf einem Prototypenansatz. Anstatt jeden einzelnen Bausparvertrag separat einzulesen und in die Zukunft fortzuschreiben, wird jeder Bausparvertrag einem oder mehreren sogenannter Prototypen zugeordnet. Diese Prototypen geben das Sparverhalten des Vertrags von Vertragseinlösung bis hin zur Zuteilung vor und werden in regelmäßigen Abständen durch Clusterverfahren bestimmt und durch Expertenwissen selektiert. Zusätzliche statistische Analysen ermitteln daraufhin für jeden Prototypen die aktuellen

[1] Die Abkürzung NBI steht für: Neuprogrammierung des Bauspartechnischen Instrumentariums

Häufigkeitsanteile für Sonderoptionen wie Kündigungen, Vertragsveränderungen, Darlehensverzichte oder Vollablösungen. Auf diesem Wege kann die Simulationskomplexität von mehreren Millionen Bausparverträgen auf die Prognose einiger hundert bis tausend Prototypen reduziert werden.

Da die Simulationsdauer und die Simulationsgenauigkeit entscheidend von der Anzahl und Art der ermittelten Prototypen abhängt, sind intensive Kollektivuntersuchungen unerlässlich, um die Qualität der Simulationsergebnisse sicherzustellen. Neben aufwändigen Backtest- und Konsistenzsimulationen wird hierbei insbesondere auf statistische Analysen und Data-Mining-Verfahren zurückgegriffen.

9.1.2 Motivation einer zusätzlichen Berücksichtigung von Kundendaten

Das NBI-Simulationsmodell der öffentlichen Bausparkassen basiert auf einer Auswertung und Prognose von Bausparverträgen. Diese Verträge werden losgelöst von Kunden betrachtet. Insbesondere gehen damit keine Informationen in das Modell ein, welche die individuelle Situation des Bausparers als Kunden oder die Wechselwirkung von mehreren Verträgen im Besitz der gleichen Person berücksichtigen.

In dieser Arbeit wurde untersucht, ob die zusätzlichen Informationen auf Kundenebene eine signifikante Verbesserung der Simulationsqualität ermöglichen und damit als Grundlage zur Verbesserung eines Simulationsmodell geeignet sind. Um diese Fragen zu beantworten, wurden zwei Problemkomplexe definiert.

1. Es wurde kontrolliert, ob die wichtigsten Prognoseziele wie die Vorhersage der Neugeschäftsverteilung, einer Kündigung oder einer Finanzierungsentscheidung von Kundenattributen stark beeinflusst werden. Diese Untersuchungen können in Abschnitt 9.2 gefunden werden.

2. Es wurde beleuchtet, auf welchen Wegen und mit welchem vertretbarem Aufwand die neuen Erkenntnisse in bestehende Simulationsmodelle auf Vertragsebene hinzugespielt werden können. Die Ergebnisse können in Abschnitt 9.3 eingesehen werden.

9.2 Ermittlung von Kundenattributen mit hohem Einfluss auf das Verhalten von Bausparern

In diesem Abschnitt werden Kundenattribute oder Vertragsereignisse gesucht, die einen besonders hohen Einfluss auf das Verhalten des Bausparers haben. Innerhalb eines Prognosemodells kann nur eine beschränkte Anzahl von Einflussfaktoren berücksichtigt werden, da mit jedem zusätzlich betrachtetem

Attribut in der statistischen Analyse der Ist-Daten die Anzahl der Fälle sinkt, die eine bestimmte Kombination von Ausprägungen aller Attribute annimmt. Sinkt die Anzahl der Fälle zu sehr oder sind sogar keinerlei Fälle vorhanden, ist auf dieser Auflösungsstufe keinerlei stabile Prognose des Verhaltens mehr möglich, da Ausreißer nicht mehr verlässlich abgefangen werden können. Einzelne Verträge könnten ansonsten in der Ist-Analyse Prognosewahrscheinlichkeiten stark verfälschen, die im Prognoseschritt (ein paar Simulationsjahre später) auf sehr viele Verträge angewendet werden.

Aus diesem Grund ist es für ein gutes Prognosemodell wichtig, die Attribute oder Ereignisse mit dem stärksten Einfluss auf ein Zielverhalten aus den Daten herauszufiltern und bei der Prognose zu berücksichtigen, während Attribute mit weniger Einfluss vernachlässigt werden, um die Stabilität der Prognose zu gewährleisten.

Aus der langjährigen Entwicklungserfahrung mit dem NBI ist bekannt, welche Attribute und Ereignisse eines betrachteten Vertrages für die Vorhersage von Sonderverhalten relevant sind. Im Folgenden konzentriert sich die Suche deswegen auf alle Attribute und Ereignisse, die über den aktuell betrachteten Vertrag hinaus gehen. So kann es für die Ermittlung einer Finanzierungswahrscheinlichkeit wesentlich sein, ob der Kunde bereits in der Vergangenheit ein Darlehen in einem Vorvertrag genommen hat.

9.2.1 Suche nach häufigen Mustern im SMMP-Modell

Die Suche nach häufigen Mustern und Assoziationsregeln in der Datengrundlage ist ein ideales Verfahren, um Attribute mit einem besonders hohen Einfluss auf beliebige Sonderverhalten zu ermitteln. Hierbei können gleich mehrere Vorteile des Ansatzes genutzt werden.

1. Die Suche nach häufigen Mustern und Assoziationsregeln geschieht vollautomatisch und muss nur einmal durchgeführt werden, um beliebige Kombinationen aus Attributen und Sonderverhalten durchzutesten.

2. Aufgrund der Kombinationsvielfalt kann nicht nur ein hoher Einfluss einzelner Attribute ermittelt werden sondern ebenso ein hoher Einfluss einer Kombination aus Attributen.

3. Die beiden Steuerungsmöglichkeiten des Supports und der Konfidenz ermöglichen gleichermaßen die Vorgabe einer Mindeststabilität der Datengrundlage (Support) als auch die Ermittlung von starken Verhaltenseinflüssen (Konfidenz).

Erfassung von Wahrscheinlichkeiten

Um eine vollständige Mustersuche durchzuführen, ist es hilfreich, zusätzlich zu dem Vorkommen eines Sonderverhalten VER auch die theoretische Möglichkeit

eines Sonderverhaltens VER^T zu erfassen. Die theoretische Möglichkeit eines Sonderverhaltens markiert in einer Sequenz, dass der Kunde zumindest einmal vor der Wahl stand, das gesuchte Sonderverhalten zu nutzen, selbst wenn er es letztlich nicht gewählt hat. Soll beispielsweise die Kündigung des Vertrages im siebten Vertragsjahr für Kunden mit Familienstand "ledig" untersucht werden, dann sollte zusätzlich zu der Assoziationsregel

$$R =< FAM : ledig > \Rightarrow < (V7, KUE) >$$

die Assoziationsregel der theoretischen Möglichkeit

$$R^T =< FAM : ledig > \Rightarrow < (V7, KUE^T) >$$

erfasst werden. Dies liegt in dem Problem begründet, dass eine Abwesenheit eines Ereignisses in den Daten nicht zwingend bedeutet, dass das Gegenereignis eingetreten ist. Möglicherweise ist das Ereignis nur nicht innerhalb des Datenzeitraumes sichtbar, der Vertrag noch zu jung oder der Vertrag in einer Vertragsphase, in der das Ereignis zu diesem Zeitpunkt sachlogisch nicht möglich ist.

Da die theoretische Möglichkeit eines Sonderverhaltens immer dann existieren muss, wenn das Sonderverhalten auch tatsächlich gewählt wurde, gilt immer $supp_D(R^T) >= supp_D(R)$. Sind Assoziationsregeln R und R^T im Ergebnis enthalten, so kann die Wahrscheinlichkeit $p(R)$ für das Eintreten von R in der Datenbank D als

$$p_D(R) = \frac{conf_D(R)}{conf_D(R^T)}$$

berechnet werden.

Wird ein minimaler Support für die Suche nach Assoziationsregeln genutzt, so kann der Umstand eintreten, dass zwar R^T in den Ergebnissen enthalten ist, aber R keinen ausreichenden Support hat. Der Support von R kann in diesem Fall jedoch über einen linearen Scan der Datenbank ermittelt werden, indem die Assoziationsregel R in allen Datenpunkten gesucht wird, die auch R^T enthalten. Der umgekehrte Fall kann aufgrund der Support-Monotonie von R^T und R nicht eintreten.

9.2.2 Ermittlung von Kundenattributen mit hoher Varianz

Bevor Attribute mit einem "hohen Einfluss auf das Kundenverhalten" gefunden werden können, muss zunächst definiert werden, was darunter mathematisch verstanden wird. Intuitiv ist man bestrebt Attribute zu finden, deren Ausprägungen zu einer möglichst breiten Streuung der betrachteten Verhaltensanteile führen und damit eine möglichst starke Separation ermöglichen.

Definition 9.1. *Es existiere ein Verhalten VER und eine Assoziationsregel*

$$R =<> \Rightarrow < VER >,$$

deren Eintrittswahrscheinlichkeit über alle Datenpunkte einer Datenbank D bei $p_D(R) = r$ liege. Weiterhin sei ein Attribut A mit den Ausprägungen $A = \{A_1, \ldots, A_n\}$ gegeben. Für jede Ausprägung A_i sei mit a_i die Anzahl der Datenpunkte gegeben, die die Ausprägung A_i annehmen, und mit r_i die Eintrittswahrscheinlichkeit der Assoziationsregel

$$R_i =< A_i > \Rightarrow < VER > .$$

Dann ist die Wurzel der gewichteten Summe der quadratischen Abstände

$$d(A, VER) = \sqrt{\frac{1}{\sum_{i=1}^{n} a_i} \cdot \sum_{i=1}^{n} a_i \cdot (r - r_i)^2}$$

das Maß des Einflusses eines Attributs auf ein Ereignis.

Bei der Konstruktion von A sei sichergestellt, dass jeder Datenpunkt in D eine Ausprägung in A annimmt. Sollte dies nicht der Fall sein, so kann eine zusätzlich Ausprägung "unbekannt" zum Attribut hinzugefügt werden, auf die alle Datenpunkte ohne bisherige Erfassung abgebildet werden. Eine automatische Korrektur oder Reproduktion von Ausprägungen eines Attributs fand im Rahmen des Projekts nicht statt.

Die Wahl des Maßes wird ersichtlich, wenn man das Attribut A als eine Abbildung versteht, die jedem Datenpunkt der Datenbank D anhand seiner Ausprägung $A_i \in A$ die Eintrittswahrscheinlichkeit r_i zuordnet. In diesem Fall kann A als eine Wahrscheinlichkeitsverteilung angesehen werden und $d(A, VER)$ beschreibt gerade die Standardabweichung der Verteilung unter Berücksichtigung des Erwartungswertes r.

Hinweis: Die Definition des Einflussmaßes einer Kombination aus Attributen erfolgt analog dem Einzelfall, indem innerhalb der Summen jede mögliche Kombination von Ausprägungen der Attribute erfasst wird.

Interpretation der Suchergebnisse

Die Interpretation des Ergebnisses einer Suche nach häufigen Assoziationsregeln oder nach Attributen mit hohem Einfluss ist nicht selbstverständlich und setzt eine hohe Fachkompetenz und Erfahrung im Umgang mit Data-Mining-Verfahren voraus. Nicht nur ist bereits die möglicherweise hohe Anzahl an gefundenen Mustern eine Herausforderung sondern auch die Verbindung der relevanten Informationen zu aussagekräftigen Ergebnissen. In Kapitel 6 wurden mehrere Verfahren vorgestellt, mit denen sich die Anzahl der Lösungen durch Beschränkung der Ausgabeinformationen oder Vorselektion von Generatoren beherrschen lässt.

Im Rahmen der hier skizzierten Untersuchungen innerhalb des Kooperationsprojekts mit den Bausparkassen und der Universität zu Köln wurden

zahlreiche Fragestellungen behandelt, Analysen durchgeführt und Ergebnisse in einem interdisziplinären Team aus Fachexperten besprochen.

Da die volle Breite der Ergebnisse den Rahmen dieser Arbeit sprengen würde, wird an dieser Stelle ein exemplarischer Ausschnitt der Endergebnisse gegeben.

9.2.3 Attribute mit hohem Einfluss auf den Abschluss eines Folgevertrages

In Simulationen auf Vertragsbasis wird in der Regel das Neugeschäft (der erwartete Umfang des Abschlusses neuer Verträge) der Bausparkasse als Eingabeparameter in das Simulationsmodell gegeben. Eine Simulation auf Kundenbasis ermöglicht hingegen, den Abschluss eines Folgevertrages eines Kunden aufgrund seines bisherigen Verhaltens zu prognostizieren. Auf diese Weise kann die Schätzung der zukünftigen Neugeschäftsverteilung vom Prognosemodell unterstützt werden.

Betrachtung von Einzelattributen

Eine Berechnung des Maßes des Einflusses einzelner Attribute auf den Abschluss eines Folgevertrages ergab folgende führende Attribute.

Einflussreiche Attribute für den Abschluss eines Folgevertrages	
Attribut	Einflussmaß
Anspargrad des aktuellsten Vertrages	17,54
Tarifart des aktuellsten Vertrages	10,20
Berufsgruppe des Kunden	8,09
Bisheriges Sparverhalten des Kunden	7,03
Alter des Kunden	6,79
Bezieher einer Abo-Zeitschrift	5,12
Familienstand des Kunden	4,98
Bezieher staatlicher Förderung	4,29

Tabelle 9.1: Übersicht der einflussreichsten Attribute auf den Abschluss eines Folgevertrages

Betrachtung von Attributsverknüpfungen

Eine Berechnung des Maßes des Einflusses einer Kombination mehrerer Attribute kann in beliebiger Detailliertheit erfolgen. Aus Gründen der Übersicht sei an dieser Stelle nur eine Übersicht der Verbindung von vier Attributen gegeben.

Einflussreiche Attribute für den Abschluss eines Folgevertrages				
Attribut 1	Attribut 2	Attribut 3	Attribut 4	Einflussmaß
Anspargrad	Tarifart	Berufsgruppe	Sparverhalten	22,52
Anspargrad	Tarifart	Alter	Sparverhalten	22,15
Anspargrad	Tarifart	Berufsgruppe	Alter	22,10
Anspargrad	Tarifart	Berufsgruppe	Gesamt-BS	21,79
Anspargrad	Tarifart	Sparverhalten	Gesamt-BS	21,79
Anspargrad	Tarifart	Alter	Gesamt-BS	21,72
Anspargrad	Tarifart	Berufsgruppe	Förderung	21,72
Anspargrad	Tarifart	Abo	Sparverhalten	21,72
Anspargrad	Tarifart	Sparverhalten	Förderung	21,67
Anspargrad	Tarifart	Berufsgruppe	Abo	21,63

Tabelle 9.2: Übersicht der einflussreichsten Attribute auf den Abschluss eines Folgevertrages

Interpretation der Ergebnisse

Die Berechnung des Einflussmaßes einzelner Attribute sowie eine Kombination von Attributen hat herausgestellt, dass einige Attribute existieren, die einen signifikanten Einfluss auf die Streuung der Wahrscheinlichkeit des Abschlusses eines Folgevertrages haben. Hierunter zählen insbesondere der Anspargrad des aktuellsten Vertrages und die Tarifart des aktuellsten Vertrages. Erst mit etwas Abstand werden Kundenattribute wie die Berufsgruppe, das Alter oder der Familienstand des Kunden wichtig.

9.2.4 Attribute mit hohem Einfluss auf die Annahme eines Bauspardarlehens

Das Kerngeschäft von Bausparkassen liegt in der Vergabe von Bauspardarlehen. Neben den klassischen Finanzierungstarifen haben sich allerdings über die Zeit auch alternative Anlegeformen wie beispielsweise die Besparung eines Riester-Bausparvertrages mit anschließender Beantragung einer Geldrente gebildet. Das aktuelle Niedrigzinsumfeld stellt Bausparkassen zusätzlich vor eine große Herausforderung, denn zum Zeitpunkt einer möglichen Finanzierung wird der Kunde die Zinskonditionen seines Vertrages mit den Zinsen auf dem Markt vergleichen. In diesem Abschnitt wird die Frage beleuchtet, welche zusätzlichen Einflussfaktoren für die Entscheidung über eine mögliche Finanzierung wesentlich sind.

Betrachtung von Einzelattributen

Eine Berechnung des Maßes des Einflusses einzelner Attribute ergab folgende führende Attribute.

Einflussreiche Attribute für die Annahme eines Bauspardarlehens	
Attribut	Einflussmaß
Anspargrad des aktuellsten Vertrages	25,70
Bewertungszahl des aktuellsten Vertrages	20,00
Bisheriges Sparverhalten des Kunden	18,67
Zinsdifferenz des Tarifzins zum aktuellen Marktzins	17,66
Tarifart des aktuellsten Vertrages	12,74
Familienstand des Kunden	8,16
Alter des Kunden	7,13
Berufsgruppe des Kunden	6,12

Tabelle 9.3: Übersicht der einflussreichsten Attribute

Betrachtung von Attributsverknüpfungen

Einflussreiche Attribute für die Annahme eines Bauspardarlehens				
Attribut 1	Attribut 2	Attribut 3	Attribut 4	Einflussmaß
Anspargrad	Tarifart	BWZ	Zinsdifferenz	35,00
Anspargrad	BWZ	Zinsdifferenz	Sparverhalten	34,65
Anspargrad	BWZ	Zinsdifferenz	Gesamt-BS	34,37
Anspargrad	BWZ	Zinsdifferenz	Alter	34,37
Anspargrad	BWZ	Zinsdifferenz	voher. Finanzierung	34,36
Anspargrad	BWZ	Zinsdifferenz	Familienstand	34,31
Anspargrad	BWZ	Zinsdifferenz	Berufsgruppe	34,29
Tarifart	BWZ	Zinsdifferenz	Sparverhalten	33,88
Anspargrad	Tarifart	BWZ	Sparverhalten	33,30
Anspargrad	Tarifart	BWZ	Alter	33,05

Tabelle 9.4: Übersicht der einflussreichsten Attribute

Interpretation der Ergebnisse

Die Analysen zeigen, dass einige Attribute wie der Anspargrad des aktuellsten Vertrages oder die Bewertungszahl des aktuellsten Vertrages existieren, die einen besonders hohen Einfluss auf den Abschluss eines Darlehens haben. Dies ist nicht weiter verwunderlich, denn tatsächlich gilt laut Tarifbedingungen ein Mindestanspargrad und eine Mindestbewertungszahl als zwingende Voraussetzung für die Gewährung eines Darlehens. Anhand dieses Beispiels kann aber verifiziert werden, dass der gewählte Ansatz tatsächlich automatisiert wesentliche Attribute herausfiltert.

Interessanter ist hingegen, dass auch die Attribute Familienstand, Alter und Berufsgruppe des Kunden messbare Einflüsse auf die Gewährung eines

Bauspardarlehens haben, die bisher im Simulationsmodell NBI nicht genutzt werden.

9.2.5 Attribute mit hohem Einfluss auf die Möglichkeit eines Kundenverlustes

Bausparkassen sind daran interessiert, dass ihre Kunden zufrieden sind und diese ihre Bausparverträge über die geplante Zeit hinaus bedienen. Nicht nur fördert dies die Kundenbindung sondern sorgt auch für eine verlässliche Planbarkeit der Liquidität des Unternehmens. Insbesondere möchte ein Unternehmen einen Kundenverlust verhindern, der eintritt, wenn ein Kunde alle seine aktiven Verträge mit der Kasse aufkündigt.

Der Kundenverlust ist in dieser Form aktuell im Simulationsmodell NBI nicht abgebildet, da hierfür ein Kundenmodell notwendig ist, das mehrere Bausparverträge dem gleichen Kunden zuordnet.

Betrachtung von Einzelattributen

Einflussreiche Attribute für einen Kundenverlust	
Attribut	Einflussmaß
Tarifart des aktuellsten Vertrages	6,96
Berufsgruppe des Kunden	6,45
Bisheriges Sparverhalten des Kunden	6,36
Anzahl der bisherigen Verträge mit Kasse	6,08
Anspargrad des aktuellsten Vertrages	5,59
Alter des Kunden	4,79

Tabelle 9.5: Übersicht der einflussreichsten Attribute

Betrachtung von Attributsverknüpfungen

Einflussreiche Attribute für einen Kundenverlust				
Attribut 1	Attribut 2	Attribut 3	Attribut 4	Einflussmaß
Anspargrad	Anzahl Verträge	Sparverhalten	Berufsgruppe	17,49
Anspargrad	Anzahl Verträge	Sparverhalten	Tarifart	17,35
Anspargrad	Anzahl Verträge	Berufsgruppe	Tarifart	16,83
Anspargrad	Anzahl Verträge	Sparverhalten	Alter	16,73
Anspargrad	Anzahl Verträge	Tarifart	Alter	16,46
Anspargrad	Anzahl Verträge	Sparverhalten	Gesamt-BS	16,41
Anspargrad	Anzahl Verträge	Tarifart	Gesamt-BS	16,06
Anspargrad	Anzahl Verträge	Berufsgruppe	Alter	15,91
Anspargrad	Anzahl Verträge	Sparverhalten	Familienstand	15,74
Anspargrad	Anzahl Verträge	Berufsgruppe	Gesamt-BS	15,73

Tabelle 9.6: Übersicht der einflussreichsten Attribute

Interpretation der Ergebnisse

Es ist nicht überraschend, dass die Wahrscheinlichkeit eines Kundenverlustes im großen Maße von der Anzahl der bisherigen Verträge mit der Kasse abhängt. Interessanter ist hingegen schon, dass mit Anspargrad und Sparverhalten des aktuellsten Vertrages starke Faktoren eingehen, die bereits Warnsignale unter Berücksichtigung nur eines Vertrages geben können. Weiterhin spielen wie in den meisten Verhalten insbesondere das Alter und die Berufsgruppe des Kunden eine Rolle.

9.3 Integration von Kundenattributen in Vertragsprognosemodelle

9.3.1 Abstraktion der gewonnenen Analyseergebnisse

Im letzten Abschnitt wurden einige Analysen bezüglich des Einflusses von Kundenattributen auf verschiedene Vertrags- und Kundenverhalten vorgestellt. Hierbei haben sich folgende erwähnenswerte Beobachtungen herausgestellt:

- Kundenattribute sind für die zuverlässige Prognose von Kundenverhalten in Bausparkollektiven nicht von entscheidender Natur. Tatsächlich lassen sich mit Vertragsattributen bereits wichtige Informationen herausfiltern, die für eine akzeptable Vorhersage nützlich sind. Jedoch zeigen die Ergebnisse, dass sämtliche Verhaltensprognosen durch die Hinzunahme von Kundenattributen verbessert werden können und somit insbesondere die langfristige Prognose eines Bausparkollektivs positiv beeinflussen können.

- Die allgemeine Größe des Einflusses von Kundenattributen ist je nach betrachtetem Verhalten unterschiedlich stark ausgeprägt. Während einige Verhalten wie die Annahme eines Bauspardarlehens insbesondere von den tariflichen Regelungen und dem Vergleich mit Marktzinsen dominiert werden, sind andere Verhalten wie der Abschluss eines Folgevertrages stark von Faktoren geprägt, die über die Annahmen eines einzelnen Vertrages hinausgehen.

- Es hängt sehr vom betrachteten Ereignis ab, welche Kundenattribute einen wesentlichen Einfluss haben und welche nicht. Die Bedeutung der Kundenattribute schwankt je nach betrachteter Fragestellung, sodass keine festen wesentlichen Kundenattribute für alle Verhaltensweisen definiert werden sollten.

Zusammenfassend kann festgehalten werden, dass die Hinzunahme von Kundenattributen in ein Simulationsmodell hilfreich ist und die Simulationsqualität spürbar verbessern kann. Das Simulationsmodell sollte jedoch so flexibel erweitert werden, dass die Anzahl und Gewichtung der Kundenattribute je nach Kundenverhalten dynamisch angepasst werden.

9.3.2 Entwurf eines Stufenplans zur Erweiterung von Vertragssimulationsmodellen

In diesem Abschnitt soll skizziert werden, wie die gewonnenen Informationen über den Einfluss von Kundenattributen praktisch genutzt werden können, um bestehende Simulationsmodelle auf Basis von Vertragsdaten zu erweitern.

Simulationsmodelle zur Prognose von Bausparkollektiven sind komplexe Systeme, die bereits seit Jahrzehnten gepflegt werden und mitunter das Zusammenspiel von mehreren Dutzend Programmen und Prognoseschritten erfordern. Solche Systeme können nicht über Nacht durch neue Ansätze ersetzt werden. Aus diesem Grund schlägt der Autor dieser Arbeit eine stufenweise Integration von Verbesserungen in vorhandene Simulationsmodelle vor.

9.3.3 1. Stufe: Erfassung und Nutzung von wesentlichen Kundenattributen auf Vertragsebene

Neue Vertragsvariablen

In einem ersten Schritt können Kundenattribute zu ihren zugehörigen Verträgen gespielt und als Vertragsmerkmale aufgefasst werden. Auf diesem Wege spart man sich zunächst die Einführung einer neuen Datenstruktur und ersetzt dies durch die Hinzunahme neuer Datenfelder. Der Nachteil dieses Vorgehens besteht in seiner Künstlichkeit, Verträge nicht tatsächlich zu Kunden zusammenzufassen. Auf diesem Wege lassen sich nur eine Handvoll Attribute verwenden, die keine komplexeren Beziehungen über mehrere Verträge hinweg erlauben.

In diesem ersten Schritt sollte man sich auf wenige Kundenattribute beschränken, die nach Analyse einen starken Einfluss auf eine Vielzahl von Verhaltensweisen haben. Zu diesen Attributen zählen

1. das Alter des Kunden,
2. die Berufsgruppe des Kunden,
3. der Familienstand des Kunden,
4. die Anzahl der Vorverträge.

Erweiterte Berechnung der Anteile für Vertragsverhalten

Die neuen Kundenattribute auf Vertragsebene können dazu genutzt werden, um eine genauere Prognose von Vertragsverhalten zu ermöglichen. Wie genau die neuen Attribute in die Berechnung der Verhalten eingehen, hängt individuell vom verwendeten Modell ab.

Im NBI werden Verhaltensanteile über eine sogenannte Häufigkeitstabelle ermittelt. Die Häufigkeitstabelle ist ein zweidimensionales Datenfeld, in der jedem Vertrag anhand von Merkmalen wie dem genutzten Sparprototypen und dem Tarif genau eine Zeile der Tabelle zugeordnet werden kann. In dieser Zeile werden daraufhin die Wahrscheinlichkeiten zur Annahme aller möglichen Vertragsverhalten in Spalten gespeichert und innerhalb der Simulation angewendet.

Die neuen Kundenattribute können in die Häufigkeitstabelle als neue Dimensionen integriert werden. Hierzu können je nach gewünschter Vorgehensweise entweder neue Zeilen unter Berücksichtigung der Attribute gebildet werden oder neue Spalten für jedes Vertragsverhalten unter Berücksichtigung verschiedener Kombinationen an Kundenattributen.

Es muss beachtet werden, dass durch die Hinzunahme neuer Attribute sich die Anzahl der möglichen Ausprägungen für Vertragsverhalten vervielfältigt. Unter diesen Gesichtspunkten ist eine Speicherung der Verhalten in einer starren zweidimensionalen Tabelle nicht optimal. Eine Alternative über Entscheidungsbäume wird im nächsten Kapitel vorgestellt.

Umgang mit Neugeschäft

Während die Ausprägungen von Kundenattributen für bestehende Verträge aus den Daten übernommen werden können, existieren zunächst keine Informationen über die Zusammensetzung des Neugeschäftes, welches im Laufe der Simulation generiert wird.

Die Verteilung von Kundenattributen auf das Neugeschäft kann jedoch durch eine Ist-Analyse von passenden Neugeschäftsjahrgängen aus den Daten ermittelt werden und dementsprechend in die Simulation übergeben werden.

Abbildung des Kundenlebenszyklus

Kundenattribute sind in den meisten Fällen nicht starr und entwickeln sich mit dem Kunden weiter. Diese Weiterentwicklung muss in einem Prognosemodell, in welchem in der Regel Prognosen über einem Zeitraum von 20 Jahren geführt werden, zwingend berücksichtigt werden.

Im einfachsten Fall muss das Alter des Kunden mit jedem Simulationsjahr hochgezählt werden. Weitere Attribute wie die Berufsgruppe oder der Familienstand müssen ebenso regelmäßig aktualisiert werden. Wie bereits bei der Verteilung der Attribute im Neugeschäft kann ebenso für diesen Fall auf eine Analyse von Ist-Daten zurückgegriffen werden. Anhand der bisherigen Kundenhistorie werden für jedes Kundenattribut Ausprägungsänderungen unter Berücksichtigung des Kundenalters erfasst und in der Simulation fortgeschrieben.

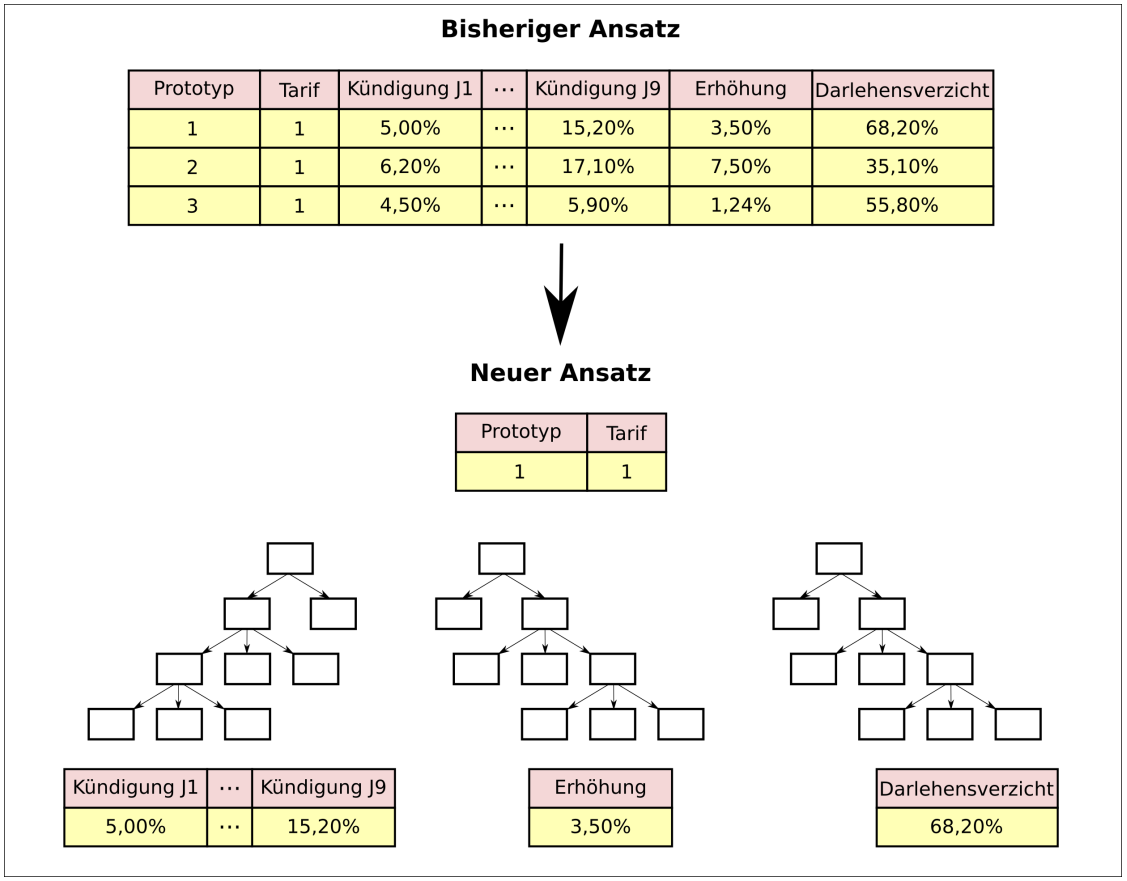
9.3.4 2. Stufe: Flexibilisierung der Verhaltensprognose auf Grundlage von Kundenattributen

Im ersten Schritt wurden feste Kundenattribute bestimmt, die für die Ermittlung aller Verhaltensweisen genutzt werden können. Dieses Vorgehen ist zwar technisch einfach zu implementieren, bildet aber nicht die Vielfältigkeit des Einflusses von Kundenattributen (und Vertragsattributen) auf die Bestimmung von Wahrscheinlichkeiten für Vertragsverhalten ab.

Die Untersuchungen dieses Kapitels haben gezeigt, dass für unterschiedliche Vertragsverhalten auch sehr unterschiedliche Kundenattribute in unterschiedlichen Einflussstärken berücksichtigt werden sollten. Aus diesem Grunde ist es anzuraten, die Ermittlung von Vertragsverhalten zu flexibilisieren und nicht für alle Vertragsverhalten die gleichen Attribute zu berücksichtigen. Eine detaillierte Beschreibung des empfohlenen Vorgehens zur Berücksichtigung eines einzelnen Vertragsverhaltens wird im nächsten Kapitel besprochen.

Im Falle des NBIs würde die Flexibilisierung darauf hinauslaufen, die Anteile für Sonderverhalten nicht in einer großen Häufigkeitstabelle zu sammeln sondern für jedes einzelne Sonderverhalten eine eigene “Häufigkeitstabelle” in Form eines Entscheidungsbaumes aufzuspannen, welcher zur Laufzeit ausgewertet wird. Auf diese Weise können effizient nur diejenigen Attribute genutzt werden, die für das entsprechende Sonderverhalten wesentlich sind.

Zur Konstruktion von entsprechenden Entscheidungsbäumen wird auf Abschnitt 10.3.2 verwiesen.



Transformation von einer Häufigkeitstabelle zu dynamischen Entscheidungsbäumen im NBI

9.3.5 3. Stufe: Übergang zu einem Prognosemodell auf Kundenebene

Ein Simulationsmodell, welches Kundenattribute zur Prognose nutzt, kann sein volles Potential erst entfalten, wenn es auch intern auf einer Kundenebene aufsetzt. Dies ermöglicht nicht nur die einfache Zuspielung neuer Erkenntnisse und Variablen sondern löst auch einige Probleme mit der Befüllung der Variablenfeldern oder der Nachbildung von komplexeren Kundenverhalten wie beispielsweise die Kündigung aller laufenden Verträge zum gleichen Zeitpunkt. Erst ein vollständiger Kundenansatz ermöglicht zum Beispiel innerhalb der Simulation den Abschluss und die Besparung eines Folgevertrages aufgrund der bisherigen Vertragsverhalten der Vorverträge.

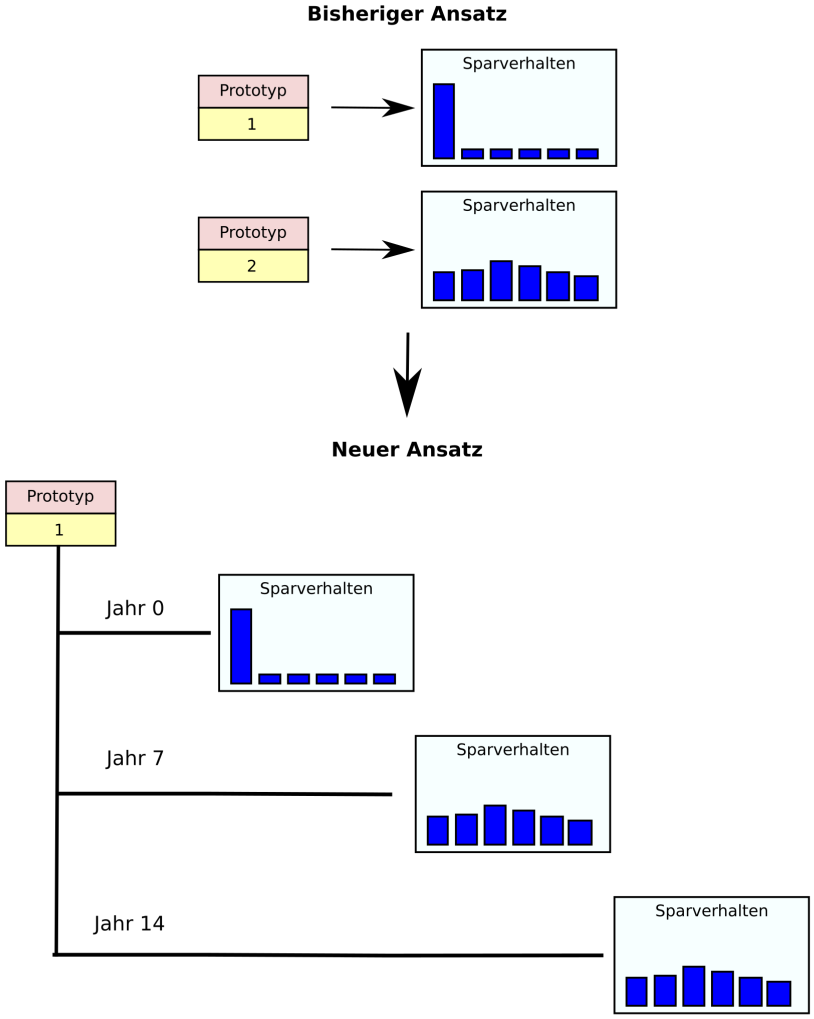
Die Ermittlung von Kundenprototypen

Das NBI setzt im Kern auf Sparerprototypen auf, die mustertypische Sparverhalten für einzelne Verträge vorgeben. Diese Sparerprototypen können weiterhin genutzt werden, um Kundenprototypen zu definieren.

Kundenprototypen bilden einen prototypischen Kundenlebenslauf ab und

können beispielsweise über das SMMP-Modell und eine Suche nach maximalen häufigen Mustern gefunden werden. Hierbei sollte man sich auf die wesentlichsten Faktoren wie die Anzahl der abgeschlossenen Verträge und ihre relativen Abschlussverschübe, die Tarifart des jeweiligen Vertrages (Rendite, Finanzierer, Altersvorsorge) und die erwarteten Sparprototypen beschränken. Alle weiteren Verhaltensweisen werden weiterhin über die Bestimmung von Häufigkeiten in Entscheidungsbäumen für Vertragsverhalten hinzugespielt.

Dieser Ansatz hat bereits wie die Definition von Sparerprototypen den Charme, dass die gewonnenen Prototypen intuitiv für Fachexperten im Bausparwesen ohne besondere Ausbildung in Data-Mining intuitiv verständlich sind und auch Wirtschaftsprüfern vermittelt werden können.



Transformation von Vertragsprototypen zu Kundenprototypen im NBI

9.4 Fazit

In diesem Kapitel wurde herausgearbeitet, dass die Hinzunahme von Kundenattributen in ein Simulationsmodell entscheidend dazu beitragen kann, die Prognosequalität des Simulationsmodells zu verbessern.

Neben der Definition der Problemstellung und der Berechnung von einflussreichen Kundenattributen wurde ebenso ein mehrstufiges Verfahren vorgestellt, um bestehende Kollektivprognosemodelle mit vertretbarem Aufwand auf Kundenmodell zu erweitern.

Im nächsten Kapitel wird detailliert darauf eingegangen, wie zusätzliche Kundenattribute genutzt werden können, um die Prognosen einzelner Vertragsverhalten möglichst effizient vorherzusagen.

Kapitel 10

Häufige Mustererkennung als Grundlage von Individualprognosen

10.1 Einleitung

Individualprognosen beschäftigen sich im Gegensatz zu Kollektivprognosen mit der Vorhersage des Einzelfalls. Zwar kann in der Theorie eine Kollektivprognose als Summe der Ergebnisse von Individualprognosen aufgefasst werden, jedoch unterscheiden sich die beiden Prognosearten in der Praxis häufig bereits in ihrem Grundkern. So nutzt beispielsweise das Kollektivprognosemodell der öffentlichen Bausparkassen keine Vorhersage individueller Verträge sondern summiert ähnliche Verträge zu Schichten, die daraufhin weitersimuliert werden. Auf diese Weise kann der Rechenaufwand der Prognose verringert werden. Rückschlüsse auf Einzelverträge sind nach Prognose allerdings nicht mehr möglich.

10.1.1 Motivation von Individualprognosen

Individualprognosen werden durchgeführt, wenn es notwendig ist, Aussagen über das zukünftige Verhalten von einzelnen Verträgen oder Kunden zu tätigen. Während das Ergebnis von Kollektivsimulationen daran gemessen wird, ob in der Summe die zu erwartenden Geldflüsse eintreten, gelten für Individualprognosen höhere Anforderungen. Individualprognosen müssen sicherstellen, dass jeder einzelne Vertrag möglichst treffend prognostiziert wird. Es ist nicht ausreichend, dass sich Fehler in den Prognosen einzelner Verträge gegenseitig ausgleichen.

Kundenwertmodelle

Individualprognosen können als Grundlage von Kundenwertmodellen genutzt werden. In Kundenwertmodellen wird versucht, die Wichtigkeit eines Kunden für die aktuellen Unternehmensziele in einer numerischen Form abzubilden. Neben einer Vergangenheitsbetrachtung spielt hierbei häufig auch die Prognose des Kundenverhaltens über die nächsten Jahre eine Rolle. Kundenwertmodelle ermöglichen es Unternehmen, ihre Kunden über eine Kundensegmentierung ge-

zielt anzusprechen und beispielsweise Kunden mit einem hohen Potentialwert intensiver zu fördern oder Kunden mit einem hohem Vergangenheitswert als treue Kunden einen besonderen Service anzubieten.

Kreditscoring-Verfahren

Kreditscoring-Verfahren werden genutzt, um die Kreditwürdigkeit einer natürlichen Person oder eines Unternehmens einzuschätzen. Anhand von Betrachtungen über die Vergangenheit, die Gegenwart und das zu erwartende zukünftige Verhalten des Kunden kann über die Vergabe oder die Höhe des Zinssatzes eines Kredites entschieden werden.

10.2 Erstellung einer Individualprognose über einen Standardprozess

Als ein Ergebnis dieser Arbeit wird ein Standardprozess zur Ermittlung einer Individualprognose für das Verhalten von Bausparkunden vorgestellt. Dieser Prozess ist in mehrere Stufen eingeteilt, die zu unterschiedlichen Zeitpunkten in unterschiedlichem Aufwand durchgeführt werden müssen. Die Stufen sind so gewählt, dass die größten Aufwände möglichst selten durchgeführt werden müssen und somit der Gesamtprozess betriebswirtschaftlich möglichst günstig strukturiert ist.

10.2.1 1. Stufe: Auswahl der Prognoseparameter

Die erste Stufe des Prozesses befasst sich mit der Auswahl der Prognoseparameter und muss zunächst nur ein einziges Mal durchgeführt werden. Es empfiehlt sich jedoch, den Einfluss der Prognoseparameter in regelmäßigen Abständen auf Änderungen zu überprüfen. Hierdurch kann ermittelt werden, ob sich möglicherweise neue Einflüsse in den Daten herauskristallisieren, die in den weiteren Prozess übernommen werden sollten.

Wahl des abzubildenden Kundenverhaltens

Zunächst muss vom Anwender entschieden werden, welches spezielle Kundenverhalten abzubilden ist. Alle weiteren Schritte des Standardprozesss beziehen sich auf die Ermittlung von Wahrscheinlichkeiten für dieses Kundenverhalten. Sollen mehrere unterschiedliche Kundenverhalten bestimmt werden, so muss die Prozesskette für jedes Verhalten separat durchlaufen werden.

Ermittlung von einflussreichen Attributen

In Abschnitt 9.2 wurde ein Verfahren vorgestellt, um Attribute mit einem hohen Einflussmaß auf ein Vertrags- oder Kundenverhalten zu ermitteln. Dieses Verfahren wird an dieser Stelle durchgeführt. Hierzu werden zunächst häufiger Muster

und häufige Assoziationsregeln in den Daten gesucht. Anhand der Ergebnisse können Attribute mit einem hohen Einflussmaß gefunden werden.

Auswahl und Sortierung der einflussreichen Attribute

Als Grundlage für die zweite Stufe müssen aus der Menge der gefundenen Attribute des vorherigen Schrittes wesentliche Kundenattribute ausgewählt und in eine absteigende Reihenfolge der Wichtigkeit sortiert werden. Diese Auswahl ist nicht trivial.

Wird als Maßstab ausschließlich das errechnete Einflussmaß betrachtet, so können die Attribute in absteigender Reihenfolge ihres Maßwertes geordnet werden. Es muss jedoch beachtet werden, dass eine gemeinsame Kombination von Attributen möglicherweise zu signifikant anderen Einflussmaßen führt als die Betrachtung der einzelnen beteiligten Attribute. Zur Ermittlung der Reihenfolge sollten also auch Kombinationen von Attributen untersucht werden.

In der Praxis gibt es häufig gute Gründe auf die Hinzunahme eines Attributs zu verzichten oder die Reihenfolge zu ändern, wenn sachlogische oder betriebswirtschaftliche Gründe dafürsprechen. So kann es sein, dass ein Attribut zwar rein statistisch einen hohen Einfluss zeigt, aber bekannt ist, dass die Datengrundlage des Attributes nicht für alle Datenpunkte korrekt erfasst wurde. Da man bei einer Individualprognose aber an einer möglichst guten Prognose für alle Datenpunkte interessiert ist, kann es sinnvoll sein, dieses Attribut nicht zu berücksichtigen.

Der wichtige Schritt der Auswahl und Sortierung der Attribute sollte aus diesen Gründen immer durch Absprache und Überprüfung eines Expertenteams mit fachlicher und technischer Expertise erfolgen. Innerhalb des Projektes wurde sich bewusst gegen eine vollautomatische Attributsauswahl über Gütemaße entschieden. Stattdessen wurden die Ergebnisse eines Backtests ausgewertet (siehe 4. Stufe) und unter Umständen Attribute ausgetauscht oder neue Attribute hinzugefügt.

10.2.2 2. Stufe: Konstruktion eines Entscheidungsbaums

Die zweite Stufe dient zur Füllung eines Entscheidungsbaumes, der für jede wesentliche Kombination von einflussreichen Attributen eine Vorhersage des Zielverhaltens ermöglicht. Diese Stufe muss bei jeder Erweiterung um neue Datenpunkte (in der Regel einmal jährlich) durchgeführt werden.

Ermittlung von häufigen Assoziationsregeln

Zunächst werden alle häufigen Assoziationsregeln gesucht, deren linke Seite aus einer beliebigen Kombination von Ausprägungen der ausgewählten Attribute und deren rechte Seite ausschließlich aus dem gesuchten Zielverhalten besteht. Hierbei wird ein Minimalsupport gewählt, der sicherstellt, dass die gefundenen

Assoziationsregeln eine ausreichend statistische Relevanz haben. In der praktischen Durchführung mit anschließendem Backtest hat sich ein Minimalsupport von 250 Kunden bewährt.

Man erhält eine Liste von Assoziationsregeln folgender Form:

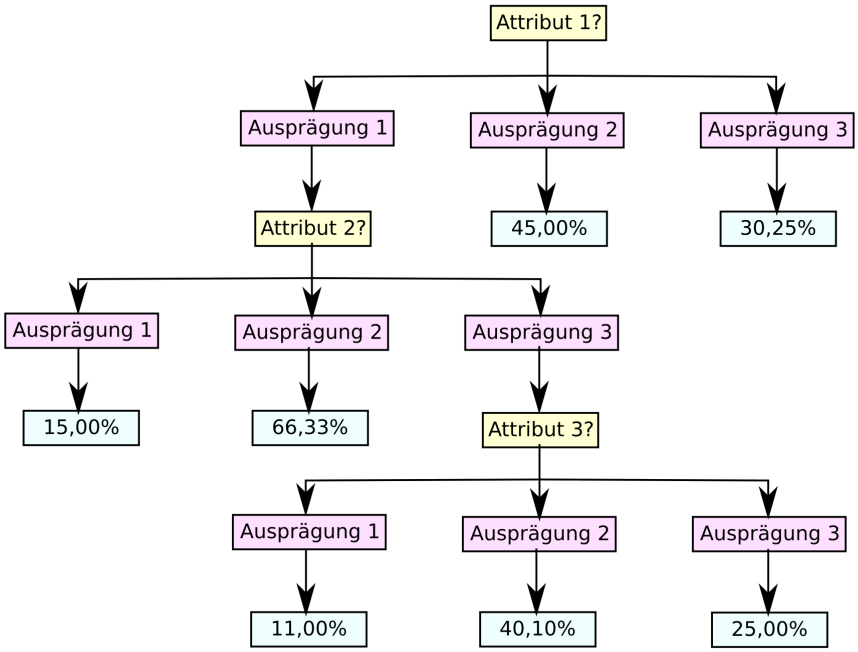
Auflistung von häufigen Assoziationsregeln		
Assoziationsregel	Support	Konfidenz
$< \{A_1 : a_1^1, A_2 : a_2^1\} > \Rightarrow < \{VER\} >$	0.1	0.15
$< \{A_1 : a_1^2\} > \Rightarrow < \{VER\} >$	0.2	0.6
$< \{A_2 : a_2^1, A_4 : a_4^1\} > \Rightarrow < \{VER\} >$	0.05	0.3
$< \{A_1 : a_1^3, A_2 : a_2^5, A_3 : a_3^2\} > \Rightarrow < \{VER\} >$	0.05	0.09
...	...	...
$< \{A_1 : a_1^1, A_2 : a_2^1\} > \Rightarrow < \{VER^T\} >$	0.2	0.3
$< \{A_1 : a_1^2\} > \Rightarrow < \{VER^T\} >$	0.3	0.9
$< \{A_2 : a_2^1, A_4 : a_4^1\} > \Rightarrow < \{VER^T\} >$	0.1	0.6
$< \{A_1 : a_1^3, A_2 : a_2^5, A_3 : a_3^2\} > \Rightarrow < \{VER^T\} >$	0.1	0.18

Tabelle 10.1: Beispielhafte Liste der herausgesuchten Assoziationsregeln

Erstellung eines Entscheidungsbaums

In diesem Schritt wird ein Entscheidungsbaum gebildet, der jedem Datenpunkt aus D oder einem beliebigen neuen Datenpunkt des SMMP-Modells eine Wahrscheinlichkeit zuordnet. Der Entscheidungsbaum durchsucht bei seinem Durchlauf die gefundenen häufigen Assoziationsregeln nach einer Regel, die möglichst viele der vorhandenen Ausprägungen der Attribute des Datenpunktes enthält. Die Attribute sind hierbei nach der Wichtigkeit geordnet, die in der ersten Stufe des Prozesses bestimmt wurde.

Der Entscheidungsbaum beginnt mit der Betrachtung aller möglichen Ausprägungen aller ausgewählten Attribute. Kann zu den jeweiligen Ausprägungen aller Attribute des Datenpunkts eine Assoziationsregel gefunden werden, so wird die Eintrittswahrscheinlichkeit dieser Regel berechnet und als Blatt des Baumes zur Ausgabe vermerkt. Ist dies nicht der Fall, wird das unwichtigste Attribut aus der Menge der betrachteten Attribute entfernt und erneut eine Suche nach einer häufigen Assoziationsregel gestartet. Dieses Verfahren wird so lange durchgeführt, bis nur noch das wichtigste Attribut verbleibt. Kann selbst für eine Ausprägung des wichtigsten Attributs keine Assoziationsregel mit ausreichendem Support gefunden werden, wird die durchschnittliche Wahrscheinlichkeit über die gesamte Datenbank berechnet und als Ausgabegröße vermerkt.



Beispiel eines konstruierten Entscheidungsbaums

Erweiterung um Zeitbeschränkungen

In der Praxis möchte man die Vorhersage einer Wahrscheinlichkeit in der Regel an strengere zeitliche Beschränkungen knüpfen. Typische Fragestellungen sind nicht, ob ein Kunde jemals seinen Vertrag kündigt sondern wie hoch die Wahrscheinlichkeit ist, dass er dies in den nächsten zwei Jahren tun wird. Um diese Fragestellungen zu klären, kann der Entscheidungsbaum um eine zeitliche Abhängigkeit ergänzt werden.

Hierzu wird für jedes Blatt des Entscheidungsbaumes statt einer einzelnen Wahrscheinlichkeit eine zeitabhängige Wahrscheinlichkeitsfunktion ausgegeben. Diese Zeitabhängigkeit ist grundsätzlich variabel definierbar. Jedoch ist man im Bausparen typischerweise an der vergangenen Zeit seit dem Vertragsabschluss des aktuellsten Bausparvertrages interessiert.

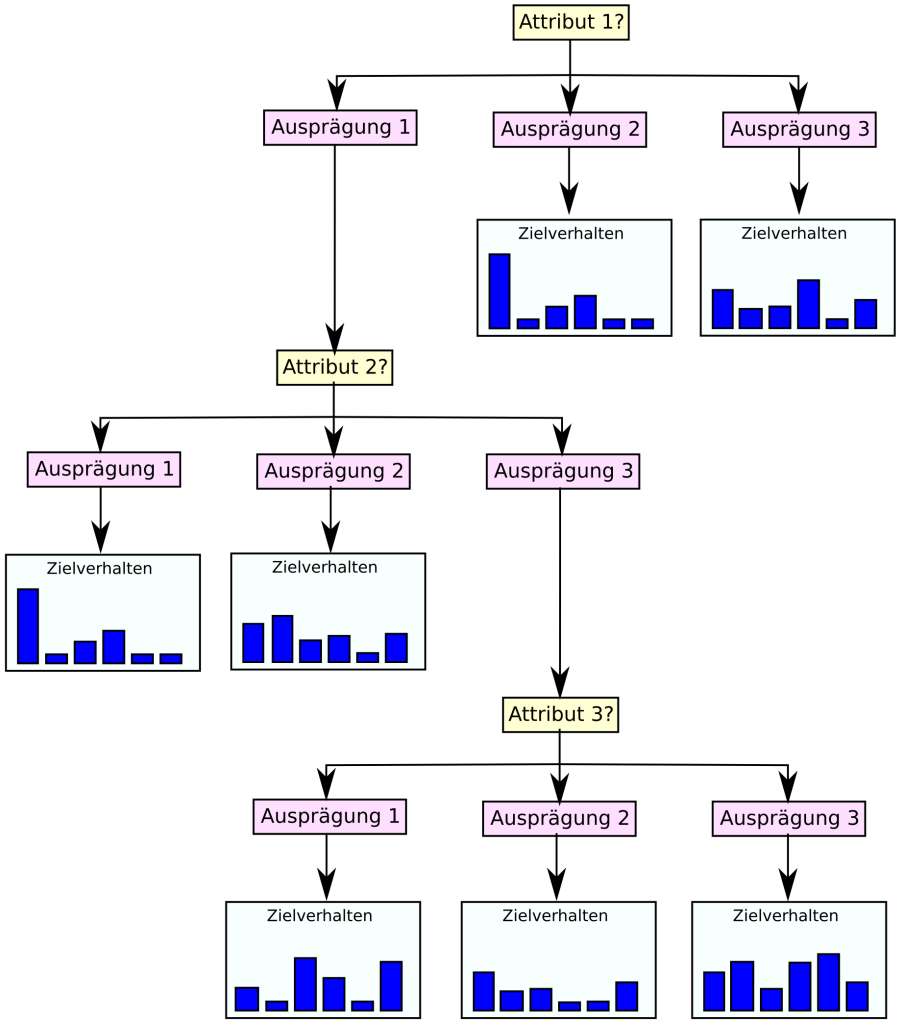
Diese Information kann nach Konstruktion aller Blätter durch einen linearen Scan der Datenbank gesammelt werden. Es wird für jeden Datenpunkt der entsprechenden Assoziationsregel vermerkt, zu welchem Zeitpunkt das gesuchte Kundenverhalten eintrat. Zur Konstruktion der Wahrscheinlichkeitsfunktion

$$f : \{0, \ldots, T\} \to [0, 1]$$

wird für jedes mögliche Jahr im Betrachtungszeitraum bis T Jahre ein Zähler und ein Nenner eingeführt. Für jeden Datenpunkt der Datenbank wird nun derjenige Zähler einen Punkt hochgezählt, der dem Eintrittszeitpunkts des

Kundenverhaltens entspricht. Weiterhin werden alle Nenner von 0 bis zum Eintrittszeitpunkts des Kundenverhaltens einen Punkt hochgesetzt. Sind alle Datenpunkte behandelt worden, so ergibt sich für die Wahrscheinlichkeitsfunktion die Wahrscheinlichkeit des Eintritts im Jahr i aus dem Quotienten des jeweiligen Zählers und Nenners.

In der nächsten Stufe des Prozesses wird erläutert, wie aus der Wahrscheinlichkeitsfunktion eine Wahrscheinlichkeit berechnet wird.



Beispiel eines konstruierten Entscheidungsbaums mit Zeitbeschränkungen

10.2.3 3. Stufe: Wahrscheinlichkeitsauswahl zur Laufzeit

Die dritte Stufe des Standardprozesses wird ausgeführt, sobald ein Datenpunkt der Datenbank bewertet werden soll. Es werden keinerlei Informationen zur Konstruktion des Modells mehr benötigt. In dieser Stufe kann jeder Kunde oder jeder Vertrag einzeln bewertet werden.

Generelles Vorgehen

Der Durchlauf durch den Entscheidungsbaum ist für einen Datenpunkt selbsterklärend. Für jeden Knoten des Baumes abseits der Blätter muss jeweils für ein Attribut in der Reihenfolge der Attributsauswahl der Nachfolgeknoten der jeweiligen Attributsausprägung durchlaufen werden. Ist der Datenpunkt an einem Blatt des Baumes angelangt, so übernimmt er als berechnete Wahrscheinlichkeit den Wert, der in diesem Blatt ausgegeben wird.

Berücksichtigung von Zeitbeschränkungen

Im Fall der zusätzlichen Berücksichtigung von Zeitinformationen erhält der Datenpunkt als Ausgabe eine zeitabhängige Wahrscheinlichkeitsfunktion

$$f : \{0, \ldots, T\} \to [0, 1].$$

Diese Wahrscheinlichkeitsfunktion muss anschließend in einen Wahrscheinlichkeitswert für den Datenpunkt umgerechnet werden.

Hierzu wird zunächst die aktuelle Position i des Datenpunktes innerhalb der Wahrscheinlichkeitsfunktion ermittelt. Beschreibt die Wahrscheinlichkeitsfunktion beispielsweise die vergangene Zeit seit Vertragsabschluss des aktuellsten Vertrages, so wird als Startwert die bisherige Laufzeit des aktuellsten Vertrages dieses Datenpunkts gewählt. Der betrachtete Zeithorizont ist eine festgelegte Vorgabe des Modells. Soll die Wahrscheinlichkeit des Eintritts des Kundenverhaltens im Verlauf der nächsten n Jahre bestimmt werden, so errechnet sich diese Wahrscheinlichkeit $p(f, i, n)$ iterativ als

$$\begin{aligned} p(f, n, 1) &= f(i), \\ p(f, n, k+1) &= p(f, n, k) + (1 - p(f, n, k)) \cdot f(i+k). \end{aligned}$$

Für den Fall, dass für einen Datenpunkt $i + n > T$ gilt, können die Ränder von f als letztbekannter Wert in die Zukunft fortgeschrieben werden.

10.2.4 4. Stufe: Modellvalidierung durch Backtest

In der letzten Stufe des Standardprozesses wird das konstruierte Modell einem Backtestvergleich unterzogen. In einem Backtest wird ein Prognosemodell evaluiert, indem das prognostizierte Ergebnis mit dem tatsächlich eingetretenen Ergebnis verglichen wird. Ein Backtest sollte zur Qualitätssicherung bei jeder Erweiterung der Datengrundlage um neue Zeitpunkte wiederholt werden.

Um einen Backtest durchzuführen, wird bei der Konstruktion des Modells nicht auf den gesamten historischen Datenzeitraum zurückgegriffen sondern nur bis zu einem Zeitpunkt geschaut, der einen Vergleich der Prognose mit einem Teil der historischen Daten ermöglicht. Steht beispielsweise ein Datenzeitraum

von 2000-2017 zur Verfügung und ist eine Prognose über 2 Jahre gewünscht, so kann der Datenzeitraum von 2000-2015 zur Konstruktion des Modells genutzt werden, um die Jahre 2016 und 2017 vorherzusagen.

Für Individualprognosen stellt sich beim Backtesting das Problem, dass das Modell für einen einzelnen Datenpunkt eine Wahrscheinlichkeit ausgibt, der Datenpunkt in der Realität aber nur zwischen den beiden Zuständen "Verhalten hat stattgefunden" und "Verhalten hat nicht stattgefunden" wählen kann. Um einen Vergleich zwischen Wahrscheinlichkeiten und Anteilen zu ermöglichen, müssen die Datenpunkte folglich in Teilmengen gruppiert werden, die eine relative Anteilsberechnung ermöglichen.

Im Rahmen des Projekts, auf dem diese Arbeit aufbaut, wurde als Lösungsweg eine Einteilung aller Datenpunkte in Gruppen mit ähnlichem Prognoseergebnis gewählt. So wurden die Kunden in Prognosekategorien von $0\% - 5\%, 5\% - 10\%, \ldots, 95\% - 100\%$ eingeteilt und innerhalb jeder Gruppe der durchschnittliche Prognosewert mit dem erreichten Realwert verglichen. Im Idealfall stimmen beide Werte überein. Die Wahl der Gruppeneinteilung hat den Vorteil, dass visuell schnell erfasst werden kann, ob das Modell Wahrscheinlichkeiten überschätzt oder unterschätzt.

Im nächsten Abschnitt werden zwei Prognosemodelle mit durchgeführtem Backtest aus der Praxis vorgestellt.

10.3 Prognose der Wahrscheinlichkeit eines Folgevertrags

Der im letzten Abschnitt vorgestellte Standardprozess zur Konstruktion einer Individualprognose wurde im Rahmen des durchgeführten Kooperationsprojekts auf eine Reihe von praktischen Fragestellungen angewandt. Zwei dieser Problemstellungen sollen beispielhaft die Funktionsfähigkeit und Prognosequalität des Ansatzes verdeutlichen.

10.3.1 1. Stufe: Auswahl der Prognoseparameter

Wahl des abzubildenden Kundenverhaltens

Es wird eine Individualprognose zur Bestimmung der Wahrscheinlichkeit des Abschlusses eines Folgevertrages innerhalb der nächsten 2 Jahren aufgestellt. Der Ausdruck "Folgevertrag" bezieht sich hierbei auf den Abschluss eines weiteren Bausparvertrages, solange mindestens ein Bausparvertrag aktuell aktiv ist. In der Prognose wird kein Unterschied vollzogen, ob es sich hierbei um den zweiten, dritten oder vierten Bausparvertrag des Kunden handelt.

Die Festlegung des Prognosezeitraumes auf die nächsten 2 Jahre erfordert

eine Erweiterung des Modells um Zeitbeschränkungen, wie sie im letzten Abschnitt vorgestellt wurde.

Ermittlung von einflussreichen Attributen

Die Problemstellung wurde bereits in Abschnitt 9.2.3 aufgegriffen und dort besprochen. Eine Auflistung von einflussreichen Attributen und deren Verknüpfung kann dort nachgelesen werden.

Auswahl und Sortierung der einflussreichen Attribute

Innerhalb eines fachlichen Expertenteams der Bauspargruppe wurden die Ergebnisse der Berechnung des Einflussmaßes analysiert und eine Auswahl und Sortierung an Attributen für die Aufstellung einer Prognose festgelegt.

1. Aktueller Anspargrad des letzten Vertrages - 11 Kategorien: Unbekannt, 0%-10%, 10%-20%, 20%-30%, 30%-40%, 40%-50%, 50%-60%, 60%-70%, 70%+, zugeteilt, gekündigt
2. Tarifart des letzten Vertrages - 3 Kategorien: Finanzierer, Unentschlossene, Renditetarife
3. Aktuelle Berufsgruppe des Kunden - 8 Kategorien: Angestellte, Arbeiter, Beamte, Juristische Personen, Rentner/Pensionäre, Selbständige/Freiberufler, Nichterwerbspersonen, Unbekannt
4. Aktuelles Sparverhalten des letzten Vertrages - 6 Kategorien: Soforteinzahler, Hohe Regelsparer, Regelsparer, Zuzahler, Geringsparer, Unbekannt
5. Alter des Kunden zum Zeitpunkt des Folgevertragsabschlusses - 6 Kategorien: 0-15, 16-21, 22-30, 31-45, 46-64, 65 und mehr
6. Bausparsumme des letzten Vertrages - 5 Kategorien: 0k-25k, 25k-50k, 50k-100k, 100k-150k, 150k und mehr
7. aktuelle Förderungszahlungen des Kunden - 2 Kategorien: VL, keine VL
8. Aktueller Bezug eines Haus-Abos des Kunden - 2 Kategorien: Ja, Nein
9. Aktueller Familienstand des Kunden - 2 Kategorien: ledig/unbekannt, verheiratet/geschieden/verwitwet
10. Aktueller Finanzierungsstatus des Kunden - 2 Kategorien: Bereits finanziert, noch nicht finanziert

Zur Ermittlung der Zeitbeschränkung der Prognose wurde die Laufzeit des aktuellsten Vertrages mit 20 Kategorien (pro Jahr) gewählt.

10.3.2 2. Stufe: Konstruktion eines Entscheidungsbaums

Ermittlung von häufigen Assoziationsregeln

Entsprechend der Auswahl und Kategorisierung der Attribute wurden häufige Assoziationsregeln erstellt. Folgende Tabelle gibt die häufigen Assoziationsregeln mit maximaler und minimaler Wahrscheinlichkeit wieder, die sich nur auf ein Attribut beziehen. Die Wahrscheinlichkeit bezieht sich in diesem Schritt noch auf einen unbeschränkten Prognosehorizont.

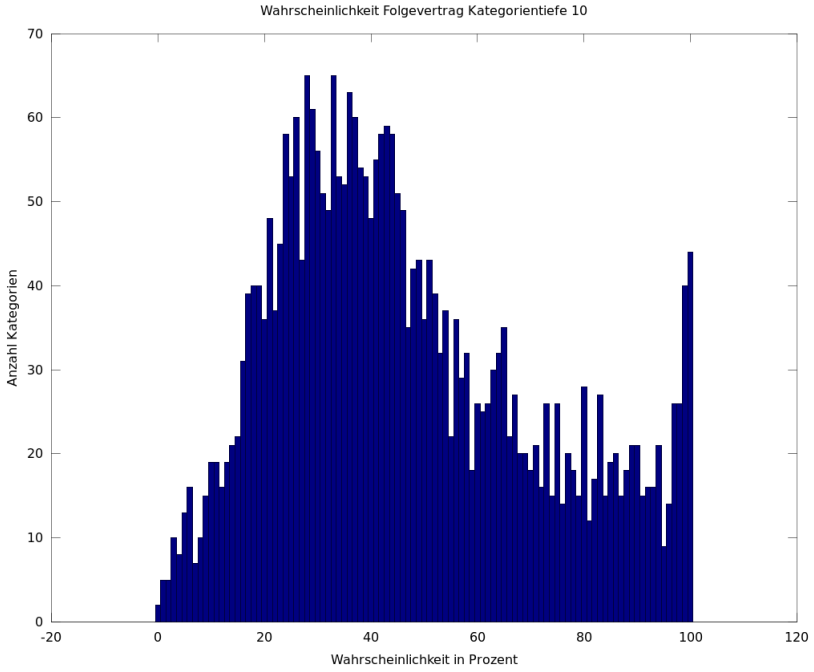
Auflistung von häufigen Assoziationsregeln	
Assoziationsregel	Eintrittswahrscheinlichkeit
$< \{Berufsgruppe : Juristische\ Person\} > \Rightarrow < \{FOLG\} >$	70,30%
$< \{Tarifart : Unentschlossene\} > \Rightarrow < \{FOLG\} >$	62,03%
$< \{Abo : Ja\} > \Rightarrow < \{FOLG\} >$	60,07%
$< \{Berufsgruppe : Beamte\} > \Rightarrow < \{FOLG\} >$	58,10%
$< \{Berufsgruppe : Rentner\} > \Rightarrow < \{FOLG\} >$	56,24%
...	...
$< \{Tarifart : Renditesparer\} > \Rightarrow < \{FOLG\} >$	34,17 %
$< \{Anspargrad : 10\% - 20\%\} > \Rightarrow < \{FOLG\} >$	34,01 %
$< \{Anspargrad : 0\% - 10\%\} > \Rightarrow < \{FOLG\} >$	30,28%
$< \{Berufsgruppe : Nichterwerb\} > \Rightarrow < \{FOLG\} >$	23,14%
$< \{Alter : 0 - 16\} > \Rightarrow < \{FOLG\} >$	16,42%

Tabelle 10.2: Liste von Assoziationsregeln mit auffälligen Wahrscheinlichkeiten für den Abschluss eines Folgevertrages

Insgesamt wurden 622.080 Assoziationsregeln gefunden, die die Kriterien für die Auswahl und Sortierung der Attribute erfüllen. Hiervon erfüllen 3.147 Assoziationsregeln den Minimalsupport von 250 Datenpunkten.

Erstellung eines Entscheidungsbaums

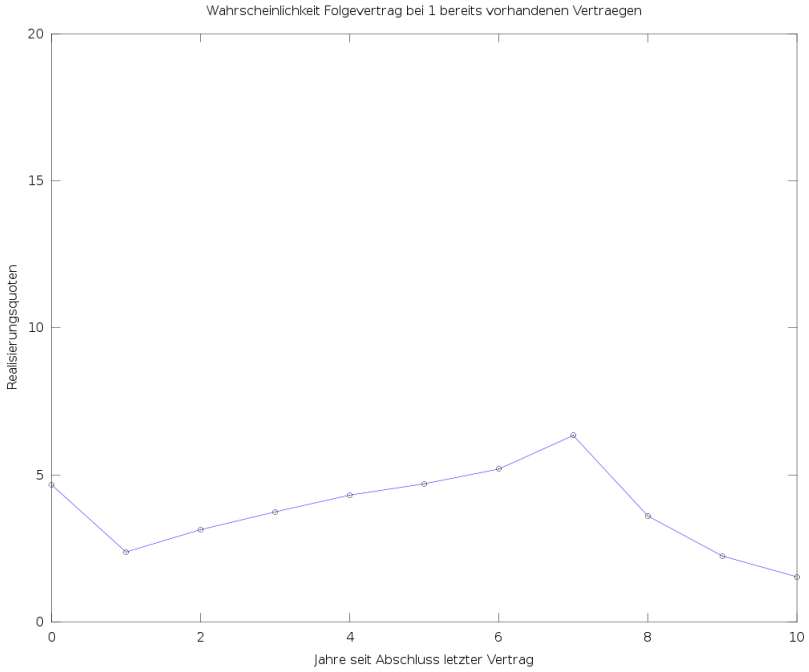
Der Entscheidungsbaum wurde wie im Standardprozess beschrieben erstellt. Folgende Grafik gibt einen Überblick über die Verteilung der 3.147 Blätter (Kategorien) des Entscheidungsbaumes. Angeführt wird die Anzahl der Blätter, die einen Wahrscheinlichkeitswert annehmen, der innerhalb eines 1%-Intervalls bezüglich des Wertes auf der horizontalen Achse liegt. Die Wahrscheinlichkeiten beziehen sich zu diesem Zeitpunkt noch auf einen unbeschränkten Prognosezeitraum.



Visualisierung der Verteilung der Wahrscheinlichkeiten für einen Folgevertrag im Entscheidungsbaum

Erweiterung um Zeitbeschränkungen

Nach Vorlage des Standardprozesses wurde zusätzlich eine Zeitanalyse für jedes Blatt des Entscheidungsbaumes durchgeführt. Folgende Grafik zeigt die Verteilung der Wahrscheinlichkeiten für einen Folgevertrag über alle Attribute, wenn bereits ein vorheriger Vertrag vorhanden ist.

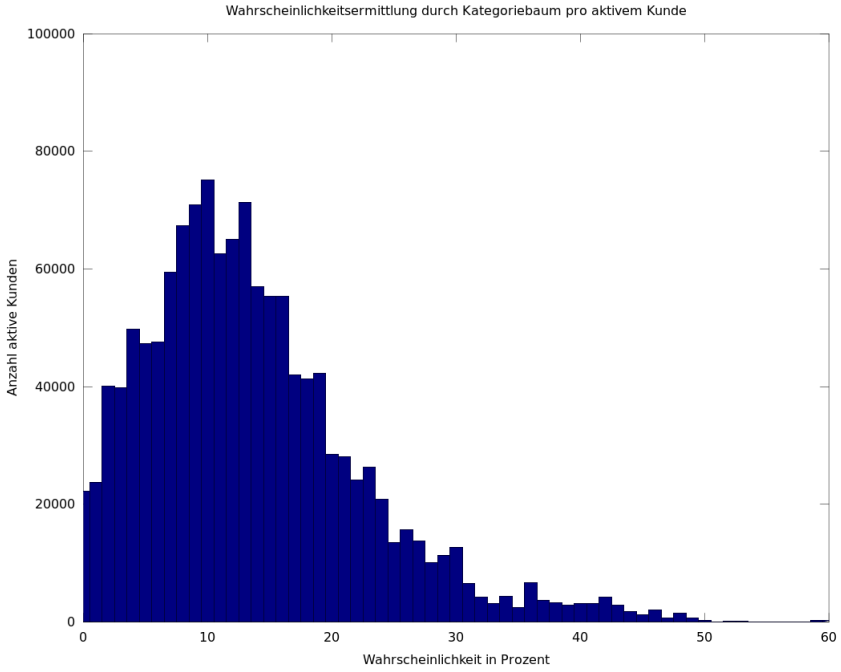


Verteilung von Wahrscheinlichkeiten auf die Laufzeit des aktuellsten Vertrages

Die Verteilung macht einen typischen Verlauf deutlich, der jedoch abhängig von der Wahl der Attribute und ihren Ausprägungen deutlich anders ausfallen kann.

10.3.3 3. Stufe: Wahrscheinlichkeitsauswahl zur Laufzeit

Ein Durchlauf aller Datenpunkte, die mindestens einen aktiven Vertrag besitzen und somit für den Abschluss eines Folgevertrages infrage kommen, ergab folgende Wahrscheinlichkeitsverteilung für einen Prognosezeitraum von 2 Jahren.

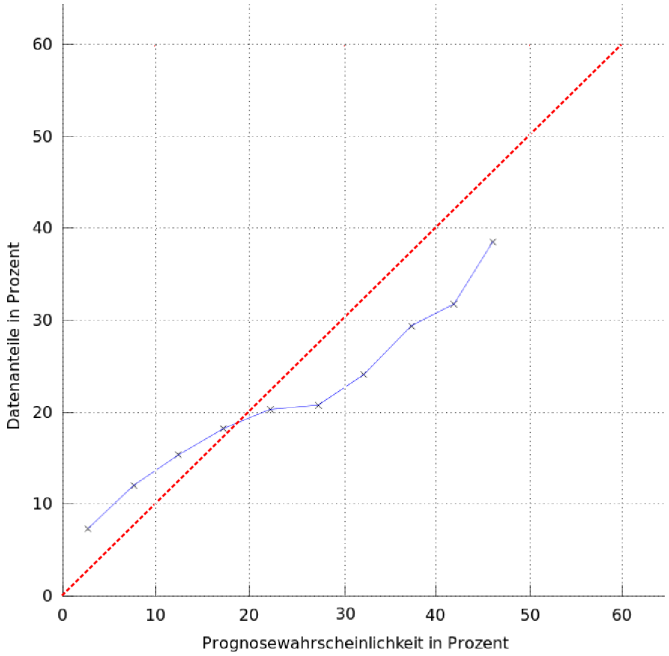


Ergebnis einer Prognose über zwei Jahre für alle aktiven Datenpunkte

Die durchschnittliche Wahrscheinlichkeit lag hierbei bei 13,52 %.

10.3.4 4. Stufe: Modellvalidierung durch Backtest

Wie im Standardprozess beschrieben, wurde ein Backtesting durchgeführt, in dem die erhaltene Prognose mit den Realisierungsquoten aus den historischen Daten verglichen wurde. Hierzu wurden die Datenpunkte in Wahrscheinlichkeitsgruppen innerhalb des gleichen Prognosewertintervalls von 5 Prozentpunkten eingeteilt und sowohl für die Prognose als auch die Realanteile der durchschnittliche Wert berechnet.



Ergebnis eines Backtests für alle aktiven Datenpunkte

In einer idealen Prognose müsste die erhaltene blaue Linie der roten Linie entsprechen. Tatsächlich führt das Prognosemodell allerdings dazu, dass niedrige Prognosen noch etwas unterschätzt werden, während hohe Prognose noch überschätzt werden. Allerdings muss berücksichtigt werden, dass sich in den oberen Randbereichen wenig Datenpunkte befinden (vergleiche letzte Grafik). Innerhalb des Kerngebiets der Wahrscheinlichkeitsverteilung kann die Prognose tatsächlich den Realverlauf gut wiedergeben.

Bei dem vorgestellten Modell handelt es sich um einen ersten Entwurf eines neuwertigen Prognosemodells auf Kundenebene. Es muss berücksichtigt werden, dass dieses Modell innerhalb weniger Wochen erarbeitet wurde, während typische Simulationsmodelle über Jahre und Jahrzehnte verfeinert werden. Wie jedes Simulationsmodell muss auch dieses Modell in regelmäßigen Abständen kontrolliert und iterativ verfeinert werden. Die bereits bei der ersten Anwendung akzeptablen Backtestergebnisse legen nahe, dass ein vollwertiger Kundenansatz wertvolle Zusatzinformationen für eine Prognose liefern kann und weiterverfolgt werden sollte.

10.4 Prognose der Wahrscheinlichkeit eines Kundenverlusts

10.4.1 1. Stufe: Auswahl der Prognoseparameter

Wahl des abzubildenden Kundenverhaltens

Es wird eine Individualprognose zur Bestimmung der Wahrscheinlichkeit für einen Kundenverlust innerhalb der nächsten 2 Jahren aufgestellt. Ein Kunden-

verlust tritt ein, wenn der aktuell aktive Kunde sämtliche noch bestehenden Verträge mit der Kasse beendet. Dies kann durch Kündigung, Darlehensverzicht oder Darlehensende geschehen und wird möglicherweise durch mehr als einen Vertrag zum gleichen Zeitpunkt verursacht.

Die Festlegung des Prognosezeitraumes auf die nächsten 2 Jahre erfordert eine Erweiterung des Modells um Zeitbeschränkungen, wie sie im letzten Abschnitt vorgestellt wurde.

Ermittlung von einflussreichen Attributen

Die Problemstellung wurde bereits in Abschnitt 9.2.5 aufgegriffen und dort besprochen. Eine Auflistung von einflussreichen Attributen und deren Verknüpfung kann dort nachgelesen werden.

Auswahl und Sortierung der einflussreichen Attribute

Innerhalb eines fachlichen Expertenteams der Bauspargruppe wurden die Ergebnisse der Berechnung des Einflussmaßes analysiert und eine Auswahl und Sortierung an Attributen für die Aufstellung einer Prognose festgelegt.

1. Aktueller Anspargrad des letzten Vertrages - 11 Kategorien: Unbekannt, 0%-10%, 10%-20%, 20%-30%, 30%-40%, 40%-50%, 50%-60%, 60%-70%, 70%+, zugeteilt, gekündigt
2. Anzahl der bisherigen Verträge des Kunden - 3 Kategorien: Erstvertrag, Zwei Verträge, Drei oder mehr Verträge
3. Aktuelles Sparverhalten des letzten Vertrages - 6 Kategorien: Soforteinzahler, Hohe Regelsparer, Regelsparer, Zuzahler, Geringsparer, Unbekannt
4. Aktuelle Berufsgruppe des Kunden - 8 Kategorien: Angestellte, Arbeiter, Beamte, Juristische Personen, Rentner/Pensionäre, Selbständige/Freiberufler, Nichterwerbspersonen, Unbekannt
5. Tarifart des letzten Vertrages - 3 Kategorien: Finanzierer, Unentschlossene, Renditetarife
6. Alter des Kunden zum Zeitpunkt des Folgevertragsabschlusses - 6 Kategorien: 0-15, 16-21, 22-30, 31-45, 46-64, 65 und mehr
7. Bausparsumme des letzten Vertrages - 5 Kategorien: 0k-25k, 25k-50k, 50k-100k, 100k-150k, 150k und mehr
8. Aktueller Familienstand des Kunden - 2 Kategorien: ledig/unbekannt, verheiratet/geschieden/verwitwet
9. Aktueller Finanzierungsstatus des Kunden - 2 Kategorien: Bereits finanziert, noch nicht finanziert

Zur Ermittlung der Zeitbeschränkung der Prognose wurde die Laufzeit des aktuellsten Vertrages mit 20 Kategorien (pro Jahr) gewählt.

10.4.2 2. Stufe: Konstruktion eines Entscheidungsbaums

Ermittlung von häufigen Assoziationsregeln

Entsprechend der Auswahl und Kategorisierung der Attribute wurden häufige Assoziationsregeln erstellt. Folgende Tabelle gibt die häufigen Assoziationsregeln mit maximaler und minimaler Wahrscheinlichkeit wieder, die sich nur auf ein Attribut beziehen. Die Wahrscheinlichkeit bezieht sich in diesem Schritt noch auf einen unbeschränkten Prognosehorizont.

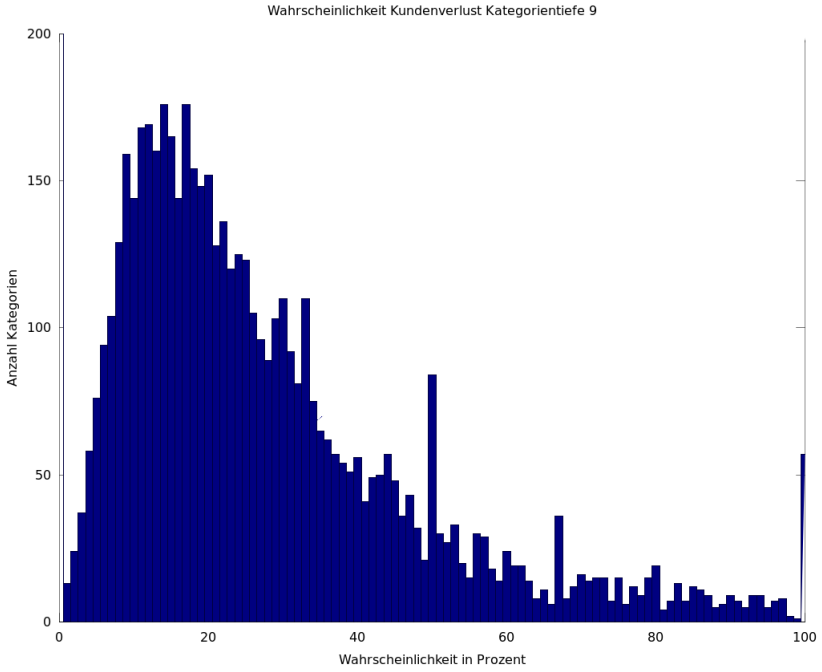
Auflistung von häufigen Assoziationsregeln	
Assoziationsregel	Eintrittswahrscheinlichkeit
$< \{Berufsgruppe : Juristische\ Person\} > \Rightarrow < \{KUVER\} >$	46,34%
$< \{Tarifart : Unentschlossene\} > \Rightarrow < \{KUVER\} >$	43,42%
$< \{Anspargrad : 0\% - 10\%\} > \Rightarrow < \{KUVER\} >$	36,00%
$< \{Anzahl\ Vertraege : Erstvertrag\} > \Rightarrow < \{KUVER\} >$	35,04%
$< \{Anzahl\ Vertraege : Zwei\ Vertraege\} > \Rightarrow < \{KUVER\} >$	34,62%
...	...
$< \{BS : 100k - 150k\} > \Rightarrow < \{KUVER\} >$	19,36 %
$< \{Finanzierung : Ja\} > \Rightarrow < \{KUVER\} >$	18,96 %
$< \{Alter : 16 - 21\} > \Rightarrow < \{KUVER\} >$	13,20%
$< \{Berufsgruppe : Nichterwerb\} > \Rightarrow < \{KUVER\} >$	9,25%
$< \{Alter : 0 - 16\} > \Rightarrow < \{KUVER\} >$	7,51%

Tabelle 10.3: Liste von Assoziationsregeln mit auffälligen Wahrscheinlichkeiten für einen Kundenverlust

Insgesamt wurden 466.560 Assoziationsregeln gefunden, die die Kriterien für die Auswahl und Sortierung der Attribute erfüllen. Hiervon erfüllen 3.431 Assoziationsregeln den Minimalsupport von 250 Datenpunkten.

Erstellung eines Entscheidungsbaums

Der Entscheidungsbaum wurde wie im Standardprozess beschrieben erstellt. Folgende Grafik gibt einen Überblick über die Verteilung der 3.431 Blätter (Kategorien) des Entscheidungsbaumes. Angeführt wird die Anzahl der Blätter, die einen Wahrscheinlichkeitswert annehmen, der innerhalb eines 1%-Intervalls bezüglich des Wertes auf der horizontalen Achse liegt. Die Wahrscheinlichkeiten beziehen sich zu diesem Zeitpunkt noch auf einen unbeschränkten Prognosezeitraum.



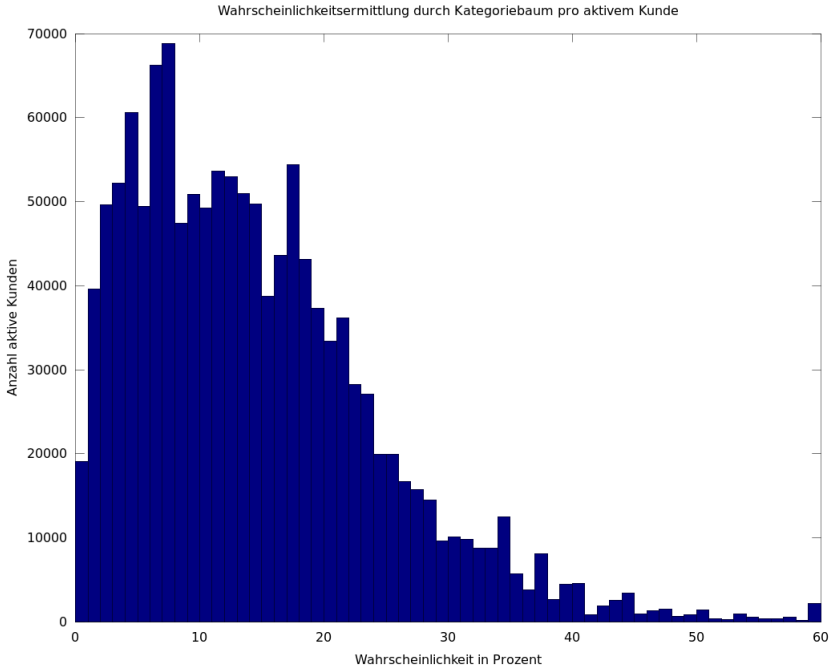
Visualisierung der Verteilung der Wahrscheinlichkeiten für einen Kundenverlust im Entscheidungsbaum

Erweiterung um Zeitbeschränkungen

Nach Vorlage des Standardprozesses wurde zusätzlich eine Zeitanalyse für jedes Blatt des Entscheidungsbaumes durchgeführt.

10.4.3 3. Stufe: Wahrscheinlichkeitsauswahl zur Laufzeit

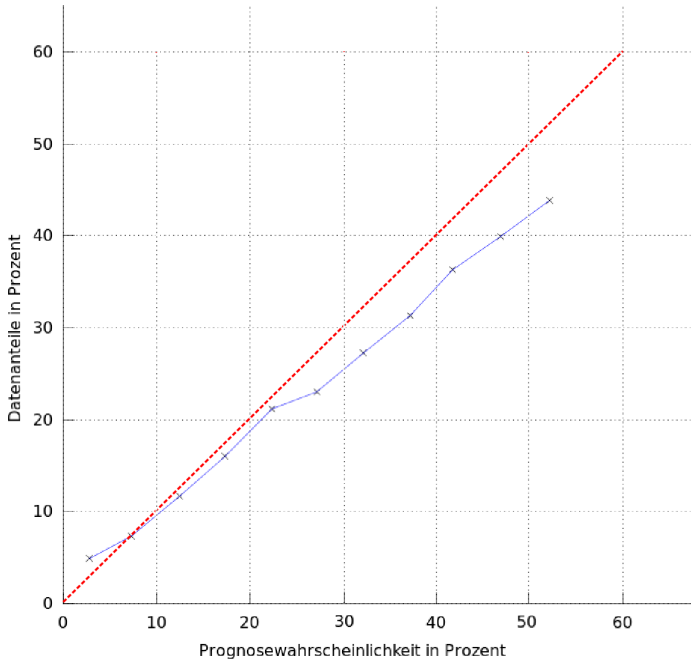
Ein Durchlauf aller Datenpunkte, die mindestens einen aktiven Vertrag besitzen und somit für einen Kundenverlust infrage kommen, ergab folgende Wahrscheinlichkeitsverteilung für einen Prognosezeitraum von 2 Jahren.



Ergebnis einer Prognose über zwei Jahre für alle aktiven Datenpunkte

Die durchschnittliche Wahrscheinlichkeit lag hierbei bei 14,51 %.

10.4.4 4. Stufe: Modellvalidierung durch Backtest



Ergebnis eines Backtests für alle aktiven Datenpunkte

Der Idealverlauf der roten Linie wird durch das Prognosemodell im Bereich bis rund 22% sehr gut abgebildet. Anhand der vorherigen Grafik ist erkennbar, dass in diesem Bereich die große Mehrheit der aktiven Datenpunkte liegt. Erst

in einem Bereich größer als 22% beginnt das Prognosemodell die tatsächlichen Realisierungsquoten eines Kundenverlustes zu überschätzen. Insbesondere die Werte über 40% Prognosewert werden allerdings nur von wenigen Datenpunkten angenommen.

Insgesamt ist damit bereits nach dem ersten Aufbau eines Simulationsmodells eine treffende Prognose des Eintritts eines Kundenverlustes ermöglicht worden. Dies ist umso erfreulicher, da bestehende Prognosemodelle auf Vertragsbasis wie das NBI diese Fragestellung in der Form gar nicht beantworten können.

10.5 Fazit

In diesem Kapitel wurden die Möglichkeiten der häufigen Mustererkennung im SMMP-Modell untersucht, um Individualprognosen in Bausparkollektiven zu ermöglichen. Es wurde ein Standardprozess zur semiautomatischen Gewinnung aller benötigten Attribute besprochen und anhand zweier realer Testszenarien aufgezeigt, dass eine zuverlässige Prognose bereits nach dem ersten Modellaufbau möglich ist.

Die erhaltenen Individualprognosemodelle können nicht nur losgelöst für beliebige dynamische Fragestellungen genutzt werden sondern bieten ebenso eine solide Grundlage für die Konstruktion eines Kollektivsimulationsmodells auf Kundenebene.

Kapitel 11

Zusammenfassung und Ausblick

Zusammenfassung

Die vorliegende Dissertation befasste sich ausgiebig mit der Suche von häufigen Mustern und Assoziationsregeln in sequenziellen Datensätzen. Hierzu wurden neue theoretische Modelle aufgestellt, Strukturanalysen durchgeführt und neuwertige Algorithmen zur Ausnutzung der besonderen Struktur von Phasendatenbanken entworfen. Diese Algorithmen konnten im direkten Vergleich die gängigen Standardalgorithmen bei vielen Parametern um eine Größenordnung in der Laufzeit schlagen.

Im praktischen Teil wurde ein neuwertiger Kundenansatz als Grundlage für Data-Mining-Verfahren in Bausparkollektiven erarbeitet, welcher auf Phasendatenbanken und der Suche nach häufigen Assoziationsregeln aufsetzt. Die gewonnenen Ergebnisse wurden genutzt, um sowohl für Kollektivsimulationen wie auch Individualprognosen neue Erkenntnisse zu generieren und Prognoseverbesserungen einzuführen. Backtestergebnisse zeigen, dass bereits die Einbeziehung weniger Kundendaten genügt, um verlässliche Vorhersagen zu ermöglichen.

Ausblick

Die Ergebnisse dieser Arbeit können auf vielfältige Weise genutzt und weiter ausgebaut werden:

- Bestehende Kollektivsimulationsmodelle können von der Hinzunahme neuer Kundenattribute profitieren und ihre Prognosequalität steigern.
- Der Standardprozess für Individualprognosen kann genutzt werden, um neue Fragestellungen zu behandeln, zu denen bisherige Prognoseverfahren auf Vertragsbasis keine Antworten liefern können.
- Die Ergebnisse der Individualprognosen können genutzt werden, um ein vollständiges Simulationsmodell auf Kundenbasis aufzubauen und einzusetzen.

- Erkenntnisse aus der Arbeit können zur Konstruktion eines Kundenwertmodells oder zur Unterstützung in Scoring-Verfahren angewandt werden.
- Ergebnisse der Arbeit können bei Fragestellungen der Kundensegmentierung und der genaueren Ansprache von Kundengruppen genutzt werden.

Der Autor dieser Arbeit ist der Überzeugung, dass die Konstruktion und Anwendung von Data-Mining-Verfahren auch in traditionellen Unternehmensumgebungen wie dem Bausparen genutzt werden kann, um neuwertige und letztlich umsatz- und gewinnsteigernde Erkenntnisse zu generieren. Dementsprechend sollten die Möglichkeiten des Data-Minings weiter erforscht und in der praktischen Arbeit vorangetrieben werden.

Literaturverzeichnis

[AAK17] Khubaib Amjad Alam, Rodina Binti Ahmad, and Kwangman Ko. Enabling far-edge analytics: Performance profiling of frequent pattern mining algorithms. *IEEE Access*, 5:8236–8249, 2017.

[AB88] Bernhard Korte und Rainer Schrader Achim Bachem. Mathematische modelle für bausparkollektive. *Bankpolitik, finanzielle Unternehmensführung und die Theorie der Finanzmärkte*, page 49, 1988.

[AFGY02] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, 2002.

[Agg15] Charu C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015.

[AJ10] Sophie Ahlswede and Tobias Just. Status quo bausparkassen. *Deutsche Bank Research*, 2010.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proceedings of the 1994 International Conference on Very Large Data Bases*, 1994.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. *Proc. 1995 Int. Conf. Data Eng.*, 1995.

[Bac97] Achim Bachem. *Analyse großer Datenmengen und Clusteralgorithmen im Bausparwesen*. Universität zu Köln, 1997.

[Bay98] Roberto Bayardo. Efficiently mining long patterns from databases. 27:85–93, 06 1998.

[BBDW94] A Bachem, L Bettmer, S Ditzen, and I Windmöller. *Simulationsmodelle für Bausparkollektive*. Univ., 1994.

[BCF+05] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu. Mafia: a maximal frequent itemset algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1490–1504, Nov 2005.

[BPT+00] Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational Logic?CL 2000*, pages 972–986. Springer, 2000.

[BR01] Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 267–273. ACM, 2001.

[Bra07] Max Bramer. *Principles of Data Mining*. Springer-Verlag London Limited 2007, 2007.

[CCA14] Jiawei Han (eds.) Charu C. Aggarwal. *Frequent Pattern Mining*. Springer International Publishing, 1 edition, 2014.

[CF04] Yin-Ling Cheung and Ada Wai-Chee Fu. Mining frequent itemsets without support threshold: with and without item constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1052–1069, 2004.

[Che05] Thomas Chevalier. *Ein Risikomodell für Bausparkollektive*. Hundt Druck GmbH, Köln, 2005.

[CXCS11] Xiaohong Cui, Jihai Xiao, Junjie Chen, and Lijun Sang. Improved algorithm for mining n-most interesting itemsets. In *Emerging Research in Artificial Intelligence and Computational Intelligence*, pages 183–189. Springer, 2011.

[CY02] Yen-Liang Chen and Chung-Ching Yu. Mining sequential patterns from multi-dimensional sequence data. 2002.

[Fak07] Petra Fakler. *Ein verbandstheoretisches Modell zur Prognose von Kreditausfallwahrscheinlichkeiten*. Hundt Druck GmbH, Köln, 2007.

[FH10] Hermann Freter and Nikolaus A.D. Hohl. *Kundensegmentierung im Kundenbeziehungsmanagement*, pages 177–199. Gabler, Wiesbaden, 2010.

[FKT00] Ada Wai-Chee FU, Renfrew Wang-Wai KWONG, and JIAN TANG. Mining n-most interesting itemsets. *Lecture notes in computer science*, pages 59–67, 2000.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[FVGG+13] Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Espérance Mwamikazi, and Rincy Thomas. Tks: Efficient mining of top-k sequential patterns. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications*, pages 109–120, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[FVGŠH14] Philippe Fournier-Viger, Antonio Gomariz, Michal Šebek, and Martin Hlosta. Vgen: Fast vertical mining of sequential generator patterns. In Ladjel Bellatreche and Mukesh K. Mohania, editors, *Data Warehousing and Knowledge Discovery*, pages 476–488, Cham, 2014. Springer International Publishing.

[FVLKK17] Philippe Fournier-Viger, Chun-Wei Lin, Uday Sai Kiran, and Yun Sing Koh. A survey of sequential pattern mining. 2017.

[FVWGT14] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S. Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In Marina Sokolova and Peter van Beek, editors, *Advances in Artificial Intelligence*, pages 83–94, Cham, 2014. Springer International Publishing.

[FVWT13] Philippe Fournier-Viger, Cheng-Wei Wu, and Vincent S. Tseng. Mining maximal sequential patterns without candidate maintenance. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications*, pages 169–180, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Gro] NIST Big Data Public Working Group. *NIST Big Data Interoperability Framework: Volume 1, Definitions*. NIST Special Publication 1500-1.

[GWHZ08] Chuancong Gao, Jianyong Wang, Yukai He, and Lizhu Zhou. Efficient mining of frequent sequence generators. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1051–1052, New York, NY, USA, 2008. ACM.

[HP00] Jiawei Han and Jian Pei. Mining frequent patterns by pattern-growth: methodology and implications. *ACM SIGKDD explorations newsletter*, 2(2):14–20, 2000.

[HPMA+00] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 355–359, New York, NY, USA, 2000. ACM.

[HPMA+01] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and MC Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, 2001.

[HWY13] Meng Han, Zhihai Wang, and Jidong Yuan. Closed sequential pattern mining in high dimensional sequences. 8, 06 2013.

[HZBR11] Guoyan Huang, Na Zuo, Lan Bai, and Jiadong Ren. Exploring multi-dimensional sequential patterns across multi-dimensional multi-sequence databases. 5:426–435, 12 2011.

[Kel92] Irene Kellershohn. *Mathematische Simulation von Bausparkollektiven mit Hilfe von empirischen Verteilungen in monothetischen hierarchischen Kollektivclusterungen*. Universität zu Köln, 1992.

[KG02] Marzena Kryszkiewicz and Marcin Gajek. Concise representation of frequent patterns based on generalized disjunction-free generators. In *Advances in Knowledge Discovery and Data Mining*, pages 159–171. Springer, 2002.

[Kle08] T. Kleiner. *Ansätze zur Kundensegmentierung und zu deren Implementierung im Finanzdienstleistungssektor: Eine empirische Analyse im Privatkundensegment von Banken*. Gabler Edition Wissenschaft : Schriften zum europäischen Management. Gabler Verlag, 2008.

[KN17] Sherly Kuriakose and Raju Nedunchezhian. Efficient adaptive frequent pattern mining techniques for market analysis in sequential and parallel systems. *Int. Arab J. Inf. Technol.*, 14(2):175–185, 2017.

[Kry01] Marzena Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 305–312. IEEE, 2001.

[Lau05] Hans Laux. *Die Bausparfinanzierung*. Verlag Recht und Wirtschaft GmbH, Frankfurt am Main, 2005.

[LHC08] Ming-Yen Lin, Sue-Chen Hsueh, and Chia-Wen Chang. Fast discovery of sequential patterns in large databases using effective time-indexing. *Information Sciences*, 178(22):4228 – 4245, 2008.

[LK98] Dao-I Lin and Zvi M. Kedem. Pincer-search: A new algorithm for discovering the maximum frequent set. In *Advances in Database Technology — EDBT'98*, pages 103–119, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[LL05] Ming-Yen Lin and Suh-Yin Lee. Efficient mining of sequential patterns with time constraints by delimited pattern growth. *Knowledge and Information Systems*, 7(4):499–514, May 2005.

[LLW08] Guimei Liu, Jinyan Li, and Limsoon Wong. A new concise representation of frequent itemsets using generators and a positive border. *Knowledge and Information Systems*, 17(1):35–56, 2008.

[MCP98] F. Masseglia, F. Cathala, and P. Poncelet. The psp approach for mining sequential patterns. In Jan M. Żytkow and Mohamed Quafafou, editors, *Principles of Data Mining and Knowledge Discovery*, pages 176–184, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[Mer95] Ralph Mermagen. *Evaluierung von Clusteralgorithmen für Bausparkollektive*. Diplomarbeit, Universität zu Köln, 1995.

[NAS+14] Shamila Nasreen, Muhammad Awais Azam, Khurram Shehzad, Usman Naeem, and Mustansar Ali Ghazanfar. Frequent pattern mining algorithms for finding associated frequent patterns for data streams: A survey. *Procedia Computer Science*, 37:109 – 116, 2014.

[NLWF05] Sze-Chung Ngan, Tsang Lam, Raymond Chi-Wing Wong, and Ada Wai-Chee Fu. Mining n-most interesting itemsets without support threshold by the cofi-tree. *International Journal of Business Intelligence and Data Mining*, 1(1):88–106, 2005.

[NWG] Arnab Roy Wo L. Chang Nancy W. Grady, Mark Underwood. Big data: Challenges, practices and technologies. *2014 IEEE International Conference on Big Data*.

[PCL+05] Marc Plantevit, Yeow Wei Choong, Anne Laurent, Dominique Laurent, and Maguelonne Teisseire. M2sp: Mining sequential patterns among several dimensions. In *Knowledge Discovery in Databases: PKDD 2005*, pages 205–216. Springer, 2005.

[PFV18] Vincent S. Tseng Philippe Fournier-Viger, Jerry Chun-Wei Lin. Spmf - an open-source data mining library. *Website http://www.philippe-fournier-viger.com/spmf/*, Version 2.32, März 2018.

[PHMA+04] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1424–1440, 2004.

[PHP+01] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88. ACM, 2001.

[PHW07] Jian Pei, Jiawei Han, and Wei Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2):133–160, Apr 2007.

[PLT06] Marc Plantevit, Anne Laurent, and Maguelonne Teisseire. Hype: mining hierarchical sequential patterns. In *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pages 19–26. ACM, 2006.

[RAS93] T. Imielinski R. Agrawal and A.N. Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 22(2), 1993.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. *Mining sequential patterns: Generalizations and performance improvements*. Springer, 1996.

[SA00] R. Srikant and R. Agrawal. Mining generalized association rules. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 29(2), 2000.

[SFMH11] Eliana Salvemini, Fabio Fumarola, Donato Malerba, and Jiawei Han. Fast sequence mining based on sparse id-lists. In Marzena Kryszkiewicz, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Raś, editors, *Foundations of Intelligent Systems*, pages 316–325, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[SK12] Abdus Salam and M Sikandar Hayat Khayal. Mining top- k frequent patterns without minimum support threshold. *Knowledge and information systems*, 30(1):57–86, 2012.

[Stu13] Tobias Stursberg. *Clusteranalyse vorklassifizierter Datensätze. Prototypengenerierung des Sparverhaltens von Bausparern*. Diplomarbeit. Universität zu Köln, 2013.

[TC10] Konstantinos Tsiptsis and Antonios Chorianopoulos. *Customer Segmentation*, pages 189–224. John Wiley and Sons, Ltd, 2010.

[TYH05] Petre Tzvetkov, Xifeng Yan, and Jiawei Han. Tsp: Mining top-k closed sequential patterns. *Knowledge and Information Systems*, 7(4):438–457, May 2005.

[Van96] Imke Miyata Vannahme. *Clusteralgorithmen zur mathematischen Simulation von Bausparkollektiven*. Hundt Druck GmbH, Köln, 1996.

[WB13] Jonathan Stuart Ward and Adam Barker. Undefined by data: A survey of big data definitions. *CoRR*, abs/1309.5821, 2013.

[WH04] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 79–, Washington, DC, USA, 2004. IEEE Computer Society.

[WHLT05] Jianyong Wang, J. Han, Y. Lu, and P. Tzvetkov. Tfp: an efficient algorithm for mining top-k frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):652–663, May 2005.

[WHP03] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 236–245, New York, NY, USA, 2003. ACM.

[Yan04] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 344–353. ACM, 2004.

[YC05] Chung-Ching Yu and Yen-Liang Chen. Mining sequential patterns from multidimensional sequence data. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 17(1), 2005.

[YHA] Xifeng Yan, Jiawei Han, and Ramin Afshar. *CloSpan: Mining: Closed Sequential Patterns in Large Datasets*, pages 166–177.

[YZZ+11] Shengwei Yi, Tianheng Zhao, Yuanyuan Zhang, Shilong Ma, and Zhanbin Che. An effective algorithm for mining sequential generators. *Procedia Engineering*, 15:3653 – 3657, 2011. CEIS 2011.

[Zak00a] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, May 2000.

[Zak00b] Mohammed J. Zaki. Sequence mining in categorical domains: Incorporating constraints. In *CIKM*, 2000.

[ZH02] Mohammed J. Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SDM*, 2002.

Kapitel 12

Vielen Dank!

An dieser Stelle möchte ich mich bei ein paar Menschen bedanken.

Ohne die Aufnahme an das Institut für Informatik durch Herrn Prof. Dr. Rainer Schrader und die Betreuung meiner Promotion, die Bereitstellung einer angenehmen Arbeitsumgebung und die notwendige Geduld, wäre diese Arbeit niemals entstanden. Ebenso danke ich Herrn Prof. Dr. Oliver Schaudt für die Zweitbetreuung und die vielen gemeinsamen Jahre in der Arbeitsgruppe.

Mein Dank geht auch an meine vielen Kolleginnen und Kollegen im Forschungsumfeld, mit denen ich in den letzten 10 Jahren zusammenarbeiten durfte. Zu ihnen zählen insbesondere Dr. Thomas Chevalier, Dr. Andrea Claßen, Dr. Fabian Senger, Matthias Weil, Dr. Jun-Gyu Kim, Dr. Roland Mainka, Dr. Dominique Ziegelmeyer, Dr. Vera Weil, Daniel Herrmann, Birgit Eppler, Annette Koenen und Martin Olschewski.

Weiterhin bedanken möchte ich mich bei meiner Mutter Sybille Schwalb für ihre Fürsorge zeit meines Lebens und bei meiner Mitarbeiterin Heike Flamm-Springstrow für die Ermutigungen, diese Arbeit nach langer Zeit abzuschließen.

Ich möchte auch anonymisiert meinen zahlreichen Gesprächspartnern, Kolleginnen und Kollegen, Kontakten und Zulieferern in den Landesbausparkassen danken, die erst den praktischen Teil meiner Arbeit ermöglichten.

Abschließend möchte ich noch allen Leserinnen und Lesern danken, die es bis zum Ende dieser Dissertation geschafft haben. Diese Arbeit hat mich die letzten 10 Jahre meines Lebens begleitet. Auch wenn sie nur einen Bruchteil der Themen umfasst, mit denen ich mich in dieser Zeit beschäftigt habe, so war es eine lehrreiche und spannende Zeit, die ich nicht missen möchte.

Kapitel 13

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie - abgesehen von unten angegebenen Teilpublikationen - noch nicht veröffentlicht worden ist, sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen der Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Herrn Prof. Dr. Rainer Schrader betreut worden.