

***Neues Konzept zur skalierbaren,  
explorativen Analyse großer Zeitreihen-  
daten mit Anwendung auf umfangreiche  
Stromnetz-Messdaten***

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

von der KIT-Fakultät für Informatik des  
Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

***M.Sc. Felix Bach***

Tag der mündlichen Prüfung: 25.07.2019

1. Referent: Prof. Dr. Veit Hagenmeyer  
2. Referent: Prof. Dr. Achim Streit

# Neues Konzept zur skalierbaren explorativen Analyse großer Zeitreihendaten mit Anwendung auf umfangreiche Stromnetz-Messdaten

von

Felix Bach



Dieses Werk ist lizenziert unter einer Creative Commons  
Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International Lizenz  
(CC BY-NC-ND 4.0 DE):

<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.de>

# Danksagung

*Die einzige Konstante im Universum ist die Veränderung.*  
Heraklit von Ephesus (etwa 540 - 480 v. Chr.)

An dieser Stelle möchte ich mich recht herzlich bei allen Personen bedanken, die zum Gelingen meiner Dissertation beigetragen haben. Die vorliegende Arbeit ist das Resultat meiner Tätigkeit als wissenschaftlicher Mitarbeiter in der Arbeitsgruppe für Simulation und Visualisierung am Institut für Automation und angewandte Informatik (IAI) des Karlsruher Instituts für Technologie (KIT). Allen Mitarbeitern, die mich in dieser Zeit begleitet haben, bin ich dankbar für die vielen wertvollen Diskussionen und die stets offene und ehrliche Kommunikation.

Mein besonderer Dank gebührt Prof. Veit Hagenmeyer und Prof. Georg Bretthauer, welche mir als Institutsleiter des IAI mit Rat und Tat zur Seite standen und für ausgezeichnete Arbeitsbedingungen sorgten sowie Prof. Achim Streit aus der Institutsleitung des Steinbuch Centre for Computing (SCC). Meinen Kollegen Heiko Maaß und Hueseyin Kemal Çakmak möchte ich für die freundlichen Gespräche danken, in denen nicht nur inhaltliche und methodische Fragen auf unkomplizierte und entgegenkommende Art geklärt werden konnten.

Die Anfertigung dieser Arbeit wurde jedoch überhaupt erst möglich durch die Unterstützung meiner Familie, der ich diese Arbeit widmen möchte. Daher danke ich von ganzem Herzen meiner lieben Frau Diana dafür, dass es sie gibt und ich mein Leben an ihrer Seite verbringen darf. Ich danke meiner Tochter Betti für ihr unbeschwertes Lachen, das sie trotz durchgestandener schmerzlicher Erfahrungen behalten hat und für ihre stets stürmischen Begrüßungen beim Nachhausekommen. Meinem Sohn Merlin danke ich für seine zauberhafte Art und seine bedingungslose Liebe.

Ein weiterer Dank geht an meine Schwester und meine Eltern, die mir als Kind einen unschätzbaren Reichtum an Erfahrungen ermöglicht haben, mir viele Türen geöffnet haben und mich stets in meiner Entwicklung und freien Entfaltung gefördert haben.

Für

Diana, Betti und Merlin



# Kurzfassung

Diese Arbeit beschäftigt sich mit der Entwicklung und Anwendung eines neuen Konzepts zur skalierbaren explorativen Analyse großer Zeitreihendaten. Hierzu werden zahlreiche datenintensive Methoden aus dem Bereich des Data-Mining und der Zeitreihenanalyse hinsichtlich ihrer Skalierbarkeit mit wachsendem Datenvolumen untersucht und neue Verfahren und Datenrepräsentationen vorgestellt, die eine Exploration sehr großer Zeitreihendaten erlauben, die mit herkömmlichen Methoden nicht effizient auswertbar sind und unter dem Begriff Big Data eingeordnet werden können.

Methoden zur Verwaltung und Visualisierung großer multivariater Zeitreihen werden mit Methoden zur Detektion seltener und häufiger Muster – sog. Discords und Motifs – kombiniert und zu einem leistungsfähigen Explorationssystem namens *ViAT* (von engl. *Visual Analysis of Time series*) zusammengefasst. Um auch Analysen von Zeitreihendaten durchführen zu können, deren Datenvolumen hunderte von Terabyte und mehr umfasst, wurde eine datenparallele verteilte Verarbeitung auf Basis von Apache Hadoop entwickelt. Sie erlaubt die Ableitung datenreduzierter Metadaten, welche statistische Eigenschaften und neuartige Strukturbeschreibungen der Zeitreihen enthalten. Auf dieser Basis sind neue inhaltsbasierte Anfragen und Auswertungen sowie Suchen nach bekannten und zuvor unbekanntem Mustern in den Daten möglich.

Das Design der entwickelten neuen Methoden und deren Integration zu einem Gesamtsystem namens *FraScaTi* (von engl. *Framework for Scalable management and analysis of Time series data*) wird vorgestellt. Das System wird evaluiert und im Anwendungsfeld der Stromnetzanalyse erprobt, welches von der Skalierbarkeit und den neuartigen Analysemöglichkeiten profitiert. Hierzu wird eine explorative Analyse hochfrequenter Stromnetz-Messdaten durchgeführt, deren Ergebnisse im Kontext des Anwendungsbereichs präsentiert und diskutiert werden.

# Abstract

The topic of this thesis is the development and application of a new concept for scalable, explorative analysis of big time series data. Several data-intensive methods from the fields of data mining and time series analysis are studied with focus on their scalability with data size. Subsequently, new techniques and data representations are presented that allow for exploration of large volumes of time-oriented data. These Big Data could previously not be analyzed efficiently due to their large volume and complexity.

Techniques for managing and visualizing big multivariate time series data are combined with methods for the detection of frequent and uncommon subseries – so-called discords and motifs to create a powerful system for explorative time series analysis called *ViAT (Visual Analysis of Time series)*. In order to allow analyses of even hundreds of terabytes of time series data and more, a distributed data-parallel processing pipeline is developed, which is based on the *Apache Hadoop* framework. It enables the extraction of data-reduced metadata that includes statistical properties of the time series and a new structure-based symbolic representation. Based on these, new evaluations and queries can be realized that enable searches for known and previously unknown patterns.

The design of the introduced methods and their integration to a software framework called *FraScaTi (Framework for Scalable management and analysis of Time series data)* is presented. This framework is evaluated and applied to the field of power grid analysis, which benefits from the scalability and new possibilities for analysis. An explorative analysis of high-frequency power measurement data is carried out and results are presented and discussed within the context of this field of application.

# Inhaltsverzeichnis

Kurzfassung .....	i
Abstract .....	ii
Inhaltsverzeichnis .....	iii
1 Einleitung.....	1
1.1 Bedeutung der Analyse großer Datenmengen.....	1
1.2 Stand der Entwicklung.....	9
1.3 Fragestellung.....	13
1.4 Ziele und Aufgaben .....	14
2 Neues Konzept zur skalierbaren, explorativen Analyse großer Zeitreihendaten.....	17
2.1 Komponenten und Datenflüsse.....	17
2.2 Datenparallele Auswertung und Merkmalsextraktion .....	20
2.3 Metadaten mit dimensionsreduzierten Zeitreihendaten für neuartige Anfragen .....	21
2.3.1 Zeitreihencodierung mit der Symbolischen Aggregat-Approximation (SAX).....	23
2.3.2 Ähnlichkeitsbewertung von Zeitreihen mit dem SAX-Distanzmaß MINDIST .....	25
2.3.3 Detektion seltener und häufiger Muster – Discords und Motifs .....	26
2.4 Client-Software mit Hadoop-Schnittstelle .....	28
2.5 Evaluation und Anwendung auf hochfrequente Stromnetz-Messdaten.....	29
3 Design neuer skalierbarer, explorativer Verfahren zur Analyse von Zeitreihen als verteiltes Softwaresystem .....	31
3.1 Voraussetzungen für verteilte Zeitreihenanalyse .....	31
3.1.1 Die Large Scale Data Facility (LSDF).....	32
3.1.2 Der Hadoop-Cluster der LSDF am KIT .....	33
3.1.3 Benötigte Funktionalität und Softwaremodule .....	34
3.2 Untersuchungen zu Performanz und Skalierung des Hadoop-Clusters der LSDF am KIT.....	35
3.3 Neuer Dateixplorer für Hadoop's Dateisystem .....	40
3.3.1 Schnittstelle zu HDFS und Benutzeroberfläche.....	41
3.3.2 Visualisierung großer komplexer Dateihierarchien.....	42

3.3.3	Navigation in langen Dateilisten.....	45
3.4	Neue datenparallele Erzeugung von Zeitreihen-Metadaten in Hadoop.....	46
3.4.1	Pig-Funktion für datenparallele Metadaten-Erzeugung.....	47
3.4.2	Ableitung und Aufbau der neuen Metadaten-Dateien .....	48
3.4.3	SAX-Codierung in Hadoop mit Lauflängencodierung.....	50
3.5	Einheitliche Schnittstelle für Zeitreihendaten.....	52
3.6	Neue interaktive Zeitreihenexploration und skalierbare visuelle Analyse.....	53
3.6.1	Visualisierungsmodul ViAT .....	54
3.6.2	Skalierbare interaktive Kurvendarstellung .....	56
3.6.3	Kooperation durch Benutzerkommentare .....	67
3.6.4	Tabellarische Farbdarstellung .....	71
3.6.5	Interaktiv parametrisierbare Box-Whisker-Plots zur Wertebereichs- und aggregierten Verlaufsdarstellung .....	72
3.6.6	Scatterplots zur interaktiven Korrelations- und Autokorrelationsanalyse .....	75
3.6.7	Zeitreihen-Histogramme mit Verlaufsdarstellung.....	77
3.7	Neue SAX-basierte Analysewerkzeuge für ViAT .....	79
3.7.1	Interaktiv parametrisierbare SAX-Codierung in ViAT .....	80
3.7.2	SAX-basiertes Clustering mit Dendrogramm-Darstellung.....	82
3.7.3	SAX-Strukturcharakteristik-Editor MetaSAX .....	87
3.7.4	SAX-Sequenzeditor zur interaktiven Suche unbekannter Muster .....	93
3.7.5	Modifizierte SAX-Sequenz-Histogramme.....	97
3.7.6	Detektion mit interaktiver Visualisierung von Discords und Motifs .....	98
3.7.7	Algorithmus zur Schnell-Detektion potentieller Discords und Motifs .....	104
3.8	Angepasste Volltextsuche für Metadaten .....	106
3.9	Zusammenfassung .....	107
4	Evaluation der Verfahren im Verbund .....	109
4.1	Zusammenspiel der Komponenten als skalierbares Gesamtsystem.....	109
4.2	Subsequenz-Mining mit SAX-Sequenzeditor .....	111
4.3	Detektion von Discords und Motifs .....	114
4.4	Strukturanalyse MetaSAX.....	125
4.5	Datenparallel auf dem Hadoop-Cluster erzeugte Zeitreihenmetadaten .....	127
4.6	Visualisierung.....	129
4.7	Zusammenfassung .....	135
5	Anwendung des neuen Analyseframeworks zur Untersuchung umfangreicher Stromnetz-Messdaten.....	137
5.1	EDR-Strommessung am KIT .....	137



5.1.1	Elektrischer Daten-Rekorder (EDR) und synchronisierte verteilte Strommessungen.....	137
5.1.2	EDR-Messdaten .....	139
5.1.3	EDR-Merkmalen.....	141
5.1.4	Archivierung und Ordnerhierarchie der Messdaten .....	144
5.2	Entwicklung eines Datenselektionswerkzeugs für HDFS-Daten .....	145
5.3	Analyse der Stromnetz-Messdaten des KIT .....	149
5.3.1	Definition von Anwendungsfällen .....	149
5.3.2	F1: Detektion von Transienten in EDR-Rohdaten.....	150
5.3.3	F2: Detektion kurzer Spannungseinbrüche in EDR-Merkmalen.....	157
5.3.4	Bewertung.....	169
6	Zusammenfassung.....	171
	Abbildungsverzeichnis .....	177
	Tabellenverzeichnis .....	183
	Code-Verzeichnis.....	184
	Abkürzungsverzeichnis.....	185
	Literaturverzeichnis.....	188
	Formelverzeichnis.....	210
	<i>- Anhang -</i>	
	Anhang A - Bewertung und Auswahl von Methoden zur skalierbaren Analyse umfangreicher Zeitreihendaten .....	
		A-1
A.1	Skalierbarkeitsbewertung bestehender Zeitreihenanalyseverfahren.....	A-1
A.1.1	Klassische Zeitreihenanalyse .....	A-2
A.1.2	Vorverarbeitung.....	A-4
A.1.3	Aktuelle Analyseaufgaben .....	A-7
A.1.4	Ähnlichkeitsbewertung von Zeitreihen .....	A-19
A.1.5	Dimensionsreduzierende Zeitreihenrepräsentationen.....	A-26
A.1.6	Wahl einer Zeitreihenrepräsentation für skalierbare Analysen .....	A-30
A.1.7	Abschnittsweise Aggregat-Approximation - PAA.....	A-31
A.1.8	Symbolische Aggregat-Approximation (SAX) .....	A-34
A.2	Explorative Analyse und interaktive Visualisierung.....	A-37
A.2.1	Skalierbarkeitsanforderungen.....	A-37
A.2.2	Kurvendarstellung .....	A-38
A.2.3	Detektion periodischen Verhaltens.....	A-39
A.2.4	Strukturanalyse mit VizTree .....	A-40

A.2.5	Zeitreihenpiktogramme zur Ähnlichkeitsvisualisierung .....	A-42
A.3	Data-Intensive Computing und Architekturwahl .....	A-45
A.3.1	„Big Data“ .....	A-46
A.3.2	Datenparallele verteilte Verarbeitung und Datenlokalität .....	A-51
A.3.3	Hadoop .....	A-54
A.4	Zusammenfassung .....	A-62
Anhang B - Stromnetzanalyse .....		B-1
B.1	Motivation der Stromnetzanalyse .....	B-1
B.1.2	Zunehmende Komplexität der Stromnetze .....	B-1
B.1.3	Das deutsche Stromnetz .....	B-2
B.2	Netzqualität .....	B-6
B.2.1	Stromausfälle und Versorgungsengpässe .....	B-7
B.2.2	Kurzfristige Spannungseinbrüche und Überspannungen .....	B-10
B.2.3	Netzurückwirkungen und Oberschwingungen .....	B-11





# 1 Einleitung

In diesem Kapitel wird dem Leser eine Einführung in die Thematik der vorliegenden Arbeit geboten. Dabei wird die Bedeutung für die einzelnen Teilbereiche und deren Zusammenhänge herausgestellt, die Arbeit wird eingeordnet und die durchgeführten Arbeiten werden motiviert. Die Darstellung des heutigen Stands der Entwicklung im Anwendungsfeld der großskaligen Analyse von Stromnetz-Messdaten führt zu Fragestellungen, die für eine Weiterentwicklung geklärt werden müssen. Im Anschluss wird die Gliederung des Textes in Kapitel beschrieben.

## 1.1 Bedeutung der Analyse großer Datenmengen

**Wissensextraktion aus Daten:** Die Gewinnung für den Menschen nützlichen Wissens aus großen Datenmengen (engl. *Knowledge Discovery From Databases*, kurz KDD) gewinnt seit einigen Jahren zunehmend an Bedeutung. Schon seit es Bibliotheken und kuratierte Sammlungen von Medien<sup>1</sup> überhaupt gibt, also bereits vor Erfindung des Buchdrucks<sup>2</sup>, werden Daten aufgezeichnet und analytisch untersucht<sup>3</sup>. Denn aus gesammelten Daten kann wertvolles, zuvor unentdecktes Wissen extrahiert werden, auch solches, an das bei der Datensammlung noch gar nicht gedacht wurde. Im Vorwort zu *The Fourth Paradigm: Data-Intensive Scientific Discovery* [109] beschreibt Gordon Bell von Microsoft Research die Theoriebildung auf Basis von Datensammlungen mit folgendem historischen Beispiel:

*“It was Tycho Brahe’s assistant Johannes Kepler who took Brahe’s catalog of systematic astronomical observations and discovered the laws of planetary motion. This established the division between the mining and analysis of captured and carefully archived experimental data and the creation of theories.”*

---

<sup>1</sup> Die frühesten überlieferten Schriftstücke in Form gravierter Tontafeln stammen aus Ägypten und Mesopotamien. Die ältesten Funde werden den Sumerern zugeschrieben und auf ca. 3000 v. Chr. datiert. Sie enthalten hauptsächlich Inventarlisten und wurden in Archiven, die meist einige tausend Tafeln umfassten, zur späteren Referenz aufgehoben [52].

<sup>2</sup> Der Druck mit beweglichen Schriftzeichen wurde bereits im 11. Jahrhundert in China praktiziert, rund vier Jahrhunderte vor Erfindung der modernen Druckerpresse im 15. Jahrhundert von Johannes Gutenberg [39].

<sup>3</sup> Ein frühes Beispiel gezielter Aufzeichnung von Beobachtungen und ihrer analytischen Auswertung stellt die Vorhersage astronomischer Ereignisse oder des Wetters zum Zwecke der besseren Planungssicherheit dar. Die babylonische Astronomie entwickelte sich beispielsweise bereits um 450 – 350 v. Chr. [226].

Brahe's Katalog war für Johannes Kepler noch in Papierform zugänglich. Während des Lesens machte sich Kepler Notizen über Muster und Regelmäßigkeiten, die ihm auffielen – stets mit Verweisen auf die Fundstelle in den Originaldaten, um Nachvollziehbarkeit für spätere Prüfungen zu gewährleisten. Später halfen Computer dabei, solche Auswertungen zu automatisieren, was wiederum Rückwirkungen auf die Art der Speicherung und Verwaltung von Daten hatte: Eine Strukturierung gespeicherter Daten sollte spätere Abfragen und statistische Analysen vereinfachen und standardisieren. Die Idee des Schemas bzw. Datenformats leistete hierzu einen wichtigen Beitrag und wurde z. B. in Form relationaler Datenbanken umgesetzt, welche durch die Ermöglichung komplexer Anfragen den Zugriff auch auf Teilaspekte von Datensätzen oder die Herausstellung von Gemeinsamkeiten und Differenzen mehrerer Datensätze erlauben.

Die noch vor 20 Jahren undenkbare Fülle und Diversität der Daten, die heute in zahlreichen Formen und über verschiedene Medien zur Verfügung stehen, enthält einen verborgenen Schatz an Wissen, der in der Disziplin des *Data-Mining* gefunden und zu Tage gefördert werden soll, indem neue Regeln und Muster abgeleitet werden. Jedoch stellt die stetig zuwachsende Daten- und Informationsflut das Themengebiet heute vor völlig neue Herausforderungen.

In beinahe allen Bereichen werden heute Daten erhoben und digital gespeichert, um spätere Auswertungen zu erlauben, sei es zur Gewinnmaximierung aus Nutzerprofilen und Unternehmensdaten oder zur Analyse experimentell gewonnener oder durch Simulationen erzeugter wissenschaftlicher Daten. Mussten früher noch gezielt Daten für geplante Untersuchungen erhoben werden, so werden heute unglaubliche Datenmengen in unserem Alltag unbemerkt erfasst und gespeichert, beispielsweise bei Suchen und Einkäufen im Internet, aber auch beim Betrieb industrieller und wissenschaftlicher Anlagen. Laut Aussagen der International Data Corporation [87] soll die Menge digitaler Daten zwischen 2005 und 2020 auf das dreihundertfache anwachsen, von 130 Exabyte auf 40 000 Exabyte (mehr als 5 200 GB pro Person). Bis 2020 wird eine Verdopplung der Datenmenge alle zwei Jahre prognostiziert. Basierend auf all diesen Daten und ihrer kombinierten Analyse sind plötzlich Untersuchungen und Anwendungen möglich, die man sich noch vor wenigen Jahren überhaupt nicht vorstellen konnte.

Andererseits stellen derartige Datenmengen eine Herausforderung für die bisherigen Verfahren der Datenverarbeitung dar. Um Datensätze miteinander in Beziehung setzen zu können, müssen sie *gleichzeitig* für Analyseanwendungen bereitstehen. Die technischen Voraussetzungen hierfür

sind einerseits eine Infrastruktur, welche die Speicherung und Verarbeitung jener neuen Datenflut ressourcensparend<sup>4</sup> erlaubt und andererseits die auf der Infrastruktur implementierten Analyse-Verfahren und -Methoden. Wie sich herausstellt, wurden viele Algorithmen aus der Anfangszeit des Data-Mining unreflektiert für die Implementierung auf der jeweils zur Verfügung stehenden Rechnerinfrastruktur entworfen und optimiert, die nur beschränkte Datenmengen verarbeiten konnten. Sie eignen sich daher eher für klassische serielle Ablaufstrukturen als für eine datenintensive Parallelverarbeitung und weisen außerdem oft exponentielle Komplexitäten<sup>5</sup> auf. Auch setzen viele Verfahren die Verfügbarkeit auszuwertender Daten im (beschränkten, relativ kleinen) Arbeitsspeicher voraus. Beides führt dazu, dass viele Verfahren schlecht mit den Daten skalieren und daher ungeeignet für große Datenmengen sind.

**Big Data und Data-Intensive Computing:** Mittlerweile hat die Größe auszuwertender Datensammlungen außerdem längst die Grenzen moderner Speichertechnologie und Datenbanksysteme überschritten, was oft eine Segmentierung großer Datensätze und eine verteilte Speicherung der Segmente auf Datenträgern mehrerer Rechner erforderlich macht. Soll ein solcher Datenbestand nun analysiert werden, stehen die relevanten Datenblöcke nicht länger auf lokalen, schnell lesbaren Medien zur Verfügung, sondern müssen über das Netzwerk geladen werden, was einen Flaschenhals<sup>6</sup> darstellt und zu inakzeptabel langen Laufzeiten solcher Analysen führen kann. Außerdem wird die maximale Größe der Daten, die gleichzeitig ausgewertet werden können, durch den lokalen Speicher und die Verarbeitungsdauer durch die Rechenleistung des Analyserechners beschränkt.

Da während der Erhebung bzw. Generierung der Daten meist noch unbekannt ist, wie sie später möglicherweise ausgewertet werden, weisen sie meist unterschiedliche, anwendungsabhängige

---

<sup>4</sup> Offensichtlich relevante Ressourcen sind die Anzahl durchzuführender Rechenschritte (Verarbeitungs- oder Zeitkomplexität) und der benötigte Speicherbedarf (Kapazitive Komplexität). Aber auch weniger offensichtliche Ressourcen können bei der Skalierung mit großen Eingangsdaten auf einmal bedeutsam werden: Etwa stellen Stromverbrauch und Kühlung, Betriebs- und Datensicherheit und für den Betrieb benötigtes Personal beschränkende Kriterien dar, welche die Umsetzbarkeit theoretisch denkbarer Analyseanwendungen einschränken.

<sup>5</sup> Algorithmen skalieren optimal mit den Daten, wenn ihre Laufzeit-Komplexität  $O(n)$  beträgt, also linear von der Datenmenge  $n$  abhängig ist. Viele Data-Mining Algorithmen weisen aber exponentielle Ordnungen von  $O(n^2)$  und höher auf.

<sup>6</sup> Der Begriff Flaschenhals stammt ursprünglich aus der Produktionsplanung und bezeichnet diejenige Teilfunktion innerhalb einer Verarbeitungskette, welche die höchste Auslastung aufweist und damit den maximal möglichen Durchsatz begrenzt [94]. Der Begriff wurde 1977 von John W. Backus aufgegriffen und auf die Informatik – insbesondere auf Rechnerarchitekturen – übertragen [19]. Dies führte zu dem heute weitverbreiteten Begriff des *Von-Neumann-Flaschenhalses*, benannt nach dem Mathematiker John von Neumann.

Strukturen und Datenformate auf, was eine Integration mit Daten aus anderen Anwendungsbereichen erschwert. Solche Datenbestände werden in aktueller Literatur als „*Big Data*“<sup>7</sup> bezeichnet (z. B. in [45], [340], [314], [284], [76], [84], [327], [65]). Dabei handelt es sich um große Sammlungen von Daten unterschiedlicher und teilweise unbekannter Herkunft und Struktur, welche aufgrund ihrer Gesamtgröße, Komplexität und Heterogenität nicht mit herkömmlichen Verfahren und Architekturen effizient verwaltet und ausgewertet werden können. Hierfür werden deshalb neue Methoden und Architekturen benötigt und unter der Gebietsbezeichnung *Data-Intensive Computing* (DIC) entwickelt. Unter DIC werden Ansätze zur *datenparallelen Verarbeitung* zusammengefasst, welche sich von *taskparalleler Verarbeitung* insofern unterscheiden, dass der hauptsächlichste Flaschenhals hier nicht aufseiten der Berechnungen liegt, sondern durch eine große Zahl an Lese- und Schreib-Operationen verursacht wird.

Eine Berücksichtigung der Datenlokalität ist hierfür außerordentlich wichtig. Vielversprechende neue Ansätze versuchen das Problem damit zu lösen, dass keine Eingangsdaten mehr zur auswertenden Recheneinheit transportiert werden, sondern der Programmcode zu jedem Rechner, auf welchem ein relevanter Datenblock gespeichert ist, wo er dann auf den jeweiligen lokalen Zieldaten parallel ausgeführt werden kann. So müssen lediglich die (Teil-) Ergebnisse der Berechnungen zusammengeführt und übertragen werden. Beispielsweise stellt das von Google Inc. eingeführte Programmiermodell *MapReduce* [66], [67] eine bereits etablierte Möglichkeit zur Umsetzung des Ansatzes dar. Allerdings stellt die Implementierung gebräuchlicher Analyse-Algorithmen in diesem neuen Kontext für Softwareentwickler keine leichte Aufgabe dar, da hierzu ein detailliertes Verständnis sowohl der Arbeitsweise der neuen Infrastrukturen als auch der Analysealgorithmen – und nicht selten zusätzlich anwendungsspezifisches Fachwissen – erforderlich ist.

**Zeitreihenanalyse und Visualisierung:** In der vorliegenden Arbeit wird ein besonderes Augenmerk auf die Analyse großer Zeitreihendaten gelegt. Eine Zeitreihe kann als zeitlich geordnete Sequenz diskreter Datenpunkte betrachtet werden. Eine solche kann beispielsweise durch Abtastung aus einem zeitlich kontinuierlichen Signal gewonnen werden, welches die Entwicklung von System- bzw. Prozesseigenschaften widerspiegelt. Meistens – aber nicht zwingend – sind die Datenpunkte dabei zeitlich äquidistant, immer jedoch in Auftrittsreihenfolge geordnet und sind somit als Vektoren darstellbar. Häufig wird zusätzlich ein Zeitstempel (engl. *Timestamp*) für jeden Datenpunkt gespeichert, welcher den genauen Messzeitpunkt angibt. Zeitstempel ermöglichen beispielsweise die Erkennung von Schwankungen der Abtastrate (*Sampling-Frequenz*). Alternativ werden auch manchmal die Zeitintervalle zwischen benachbarten Messwerten angegeben. Bei

---

<sup>7</sup> Da der Begriff *Big Data* nicht klar definiert ist und außerdem den Größenaspekt auf Dateiebene überbetont, wird neuerdings auch manchmal von *Complex Data* gesprochen.



exakt äquidistanten Zeitreihen entfällt die Notwendigkeit der Zeitangabe für einzelne Datenpunkte.

Des Weiteren gibt es multivariate Zeitreihen, d. h. dass sich mehr als eine Merkmalsausprägung über die Zeit verändert, wodurch für jeden Zeitpunkt der Reihe statt eines skalaren Wertes ein Vektor vorliegt. Prinzipiell können auch beliebige andere Sequenzen als Zeitreihen interpretiert werden, die weder eine Funktion der Zeit darstellen noch einen ordinalen Wertebereich aufweisen müssen, wobei den Elementen dann eine künstliche Ordnung zugewiesen wird (z. B. eine lexikografische). Gängige Beispiele hierfür sind DNA-Sequenzen oder Texte. Auch zahlreiche andere Arten von Daten können in Zeitreihen konvertiert werden, indem eine Serialisierungstransformation durchgeführt wird.

Problematisch bei Analyse und Vergleich mehrerer Zeitreihendaten sind die unterschiedlichen Wertebereiche und die außerordentlich hohe Dimensionalität. Die Skalierbarkeit klassischer Analysemethoden wird den Anforderungen vieler neuer Anwendungsbereiche nicht gerecht (vergl. z. B. [86], [129]), weshalb neue geeignete Datenrepräsentationen und Methoden benötigt werden, um die neuen Aufgaben der Zeitreihenanalyse zu ermöglichen. Hierzu gehören z. B. die Mustererkennung (*Pattern Matching*)<sup>8</sup>, die Entdeckung neuer, unbekannter Muster (*Pattern Discovery*)<sup>9</sup> sowie die ähnlichkeitsbasierte Suche (*Query by Content*) [81] und darauf aufbauende Methoden. Weitere aktuelle Forschungsschwerpunkte stellen das Mining von multivariaten Zeitreihen, in Echtzeit aktualisierten („gestreamten“) Zeitreihen sowie Sicherheits- und Datenschutzaspekte dar [86].

Folgendes Beispiel aus einer Veröffentlichung zur DNA-Sequenzanalyse [220] verdeutlicht die Problematik und ist dabei auf alle datenintensiven Wissenschaften übertragbar:

*“Most of the current bioinformatics analysis tools, however, do not support parallelization. As a result, this exponential data growth has made most of the current bioinformatics analytic tools obsolete because they fail to scale with data, either by taking too much time or too much memory. For example, it would take ~80 CPU years to BLAST the 268 GB cow rumen metagenome data (Hess et. al., 2011) against NCBI non-redundant database. De novo assembly of the full dataset by a short read assembler, such as Velvet (Zerbino and Birney, 2008), would require computers with 41 TB RAM and take several weeks to complete. Re-engineering the current bioinformatics tools to fit into parallel programming models requires software engineers with expertise in high-performance computing, and parallel algorithms take significantly longer time to develop than serial algorithms ([220],*

---

<sup>8</sup> *Pattern Matching* bezeichnet die Ermittlung des Vorkommens eines gegebenen Musters in einer Datenmenge.

<sup>9</sup> Der Begriff *Pattern Discovery* bedeutet das automatische Auffinden neuer, bislang unbekannter Muster in einem Datenbestand.

*Fig. 1). In addition, at this scale of computing hardware failures become more frequent, and most bioinformatics software lack robustness so that once they fail they have to be restarted manually. All these challenges contribute to the bottleneck in large-scale sequencing analysis.”*

Bei der Entwicklung existierender Verfahren und Werkzeuge zur Zeitreihenanalyse spielten Skalierungseffekte eine untergeordnete Rolle. Zeitreihen mit einigen tausend Datenpunkten galten bereits als groß und die Auswertung von Zeitreihen mit Milliarden von Datenpunkten wurde aufgrund der hohen Dimensionalität überhaupt nicht in Betracht gezogen, da keine geeignete Infrastruktur für derartige Berechnungen bekannt war. Ein Großteil der Forschungsarbeit setzte sich außerdem ausschließlich mit der Analyse der hochdimensionalen Zeitreihen im Sinne ihrer Ursprungswerte auseinander. Das fundamentale Problem der Zeitreihenrepräsentation wurde hingegen erst im letzten Jahrzehnt in den Fokus gerückt, um die Dimensionalität zu reduzieren und effektivere, effizientere Verfahren zu ermöglichen, weshalb viele der existierenden Ansätze nicht direkt für neue Repräsentationen anwendbar sind (vergl. [86], [334] für eine allgemeine Bewertung, [212] im Bereich der Wissensextraktion und [48], [57], [176], [182], [319] aus Sicht der symbolischen Analyse).

Das langfristige Ziel der klassischen Zeitreihenbewertung ist es, Gesetzmäßigkeiten bzw. ein Modell über das Zustandekommen der Werte aufzustellen. Ein solches soll u. a. eine Analyse im Sinne der Erkennung bisheriger Entwicklungsstrukturen und kausaler Zusammenhänge ermöglichen, sowie eine Prognose – etwa für Wetterdaten – z. B. durch Regression, erlauben. Oft ist dabei das Ziel, kontrollierend, d.h. steuernd bzw. regelnd, in den zugrunde liegenden Prozess einzugreifen. Während die eleganten mathematisch orientierten Methoden sehr gute Ergebnisse für kleinere Zeitreihen liefern, versagen sie häufig aufgrund von Überbewertungen lokaler Unregelmäßigkeiten und extremer Ressourcenanforderungen für sehr große Zeitreihen, welche Ausreißer und Rauschen enthalten, und werden deshalb selten hierfür eingesetzt. Für letztere kann ein Modell daher in der Praxis meist nur mit hinreichendem Vorwissen über den Prozess bzw. das untersuchte System aufgestellt werden, welches jedoch oft nicht a priori bekannt ist.

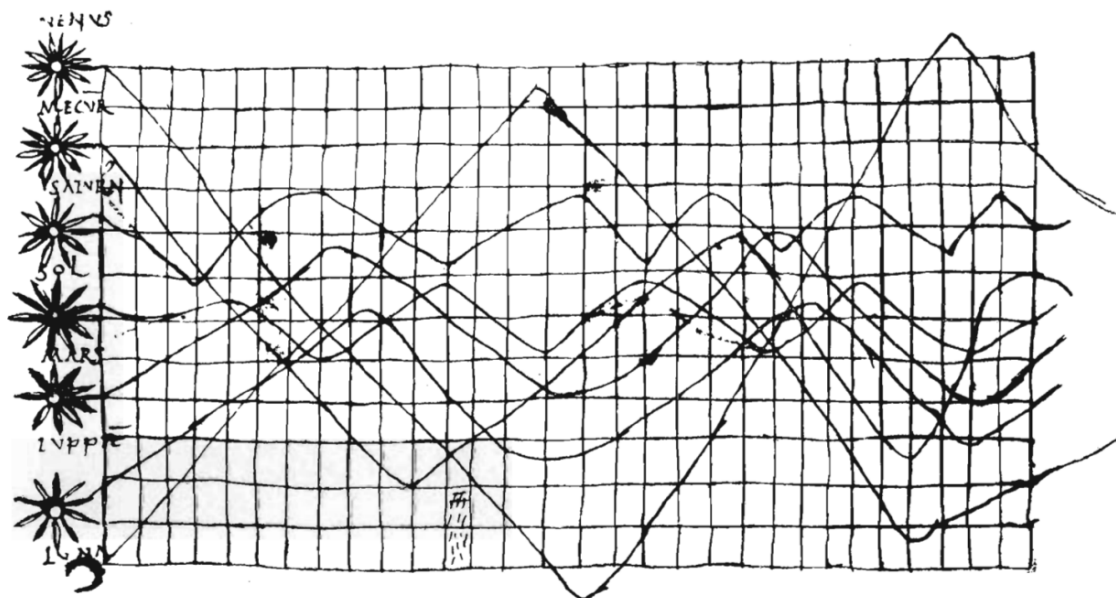
Die sog. *Explorative Datenanalyse* (EDA) setzt sich mit schwierig zu bearbeitenden großen Daten auseinander und stellt Techniken für erste Einschätzungen mit Überblickscharakter und für eine Hypothesenbildung bereit. Den ersten Schritt in der Erschließung und Charakterisierung großer Zeitreihendaten stellt nahezu immer die Visualisierung dar. Ein Kurven-Plot beispielsweise erlaubt bereits erste Aussagen über den zeitlichen Verlauf. Etwa sind Trends und Saisonalitäten sowie, bei multivariaten Zeitreihen, Abhängigkeiten zwischen einzelnen Kurven für den menschlichen Betrachter sofort erkennbar. Kurven-Plots ermöglichen außerdem die intuitive Erkennung von Ausreißern oder ungewöhnlichen lokalen Teilsequenzen (sog. *Discords*).

Kurven-Plots sind deshalb so erfolgreich, weit verbreitet und selbst aus alltäglichen Bereichen nicht wegzudenken, weil sie eine optimale Basis für die noch immer unübertroffene visuelle Mustererkennung des Menschen bereitstellen. Es existieren zahlreiche weitere graphische Darstellungen für Zeitreihendaten, wie z. B. Histogramme für Werteverteilungen, Streudiagramme (eng. *Scatterplots*), Box-Whisker Plots etc., wobei jede Darstellungsform ihre spezifischen Stärken und Schwächen in Bezug auf das Sichtbarmachen einzelner Merkmale aufweist.

In seinem Buch „*Visual Display of Quantitative Information*“ von 2001 [296] beschreibt Edward R. Tufte die Vorzüge der Kurvendarstellung folgendermaßen:

*„The time-series plot is the most frequently used form of graphic design. With one dimension marching along to the regular rhythm of seconds, minutes, hours, weeks, months, years, centuries, or millennia, the natural ordering of the time scale gives this design a strength and efficiency of interpretation found in no other graphic arrangement.“*

Abbildung 1.1 zeigt die nach Edward R. Tufte erste bekannte Darstellung eines zeitlichen Verlaufs als Kurven-Plot. Das Bild stammt etwa aus dem 10. Jahrhundert n. Chr. und stellt die zeitliche Entwicklung der orbitalen Inklinationen von sechs Planeten unseres Sonnensystems dar. Prinzipiell hat sich die Darstellung seither nicht wesentlich verändert – einem modernen Betrachter erschließt sich die Bedeutung spontan.



**Abbildung 1.1:** Frühe Kurvendarstellung der Inklinationen planetarer Orbits aus dem zehnten Jahrhundert n. Chr., laut E. Tufte das älteste bekannte Beispiel eines Versuches, Werte graphisch darzustellen. (Quelle: [296], Original: H. Gray Funkhouser, „A Note on a Tenth Century Graph“, *Osiris* (1936), S. 260-262).

Ein neues Skalierungsproblem, das bei der Kurvendarstellung großer Zeitreihen entsteht, ist dass die Dichte der Daten die Auflösungskapazität der Darstellung überschreiten kann. Sollen beispielsweise 100 000 Datenpunkte auf einem Display mit 1024 Pixeln Breite dargestellt werden, so geht unweigerlich Information bei der Darstellung verloren, da nicht alle Werte gleichzeitig angezeigt werden können. Dasselbe gilt für eine große Anzahl von Zeitreihen. Weitere Probleme beim Visualisieren großer Zeitreihendaten entstehen durch lange Übertragungszeiten, hohe Ressourcenanforderungen und die oft komplexe Bedienbarkeit und schwierige Interpretierbarkeit durch den Benutzer.

**Energie-Messdaten:** Ein relativ neues Anwendungsfeld, welches der beschriebenen Skalierungsproblematik gegenübersteht, ist die Analyse großer Energie-Messreihen, die durch hochfrequente und verteilte synchronisierte Strom- und Spannungsmessungen im Stromnetz erzeugt werden. Derartige Messdaten werden auch am *Karlsruher Institut für Technologie* (KIT) aufgezeichnet und untersucht. Das KIT nimmt seit Jahrzehnten eine wichtige Rolle in der Erforschung ziviler Energieerzeugung ein. Im Zuge der sog. *Energiewende*, also des Übergangs der vornehmlich durch Kern- und Kohlekraftwerke geprägten Energieerzeugung hin zu einer größeren Beteiligung sog. *Erneuerbarer Energien* wie Wind- und Solarenergie, wurde das *KIT Zentrum Energie* [160] ins Leben gerufen. Es gehört mit aktuell ca. 1250 Mitarbeitern zu den größten interdisziplinären Forschungseinrichtungen zum Thema Energie in Europa.

In diesem Rahmen wird am *Institut für Angewandte Informatik* (IAI) innerhalb des institutsübergreifenden *Simlab Energy* die komplexe Dynamik in Stromnetzen erforscht. Hierzu werden Strom und Spannung an mehreren Stellen innerhalb des eigenständigen Stromnetzes des KIT Campus Nord mit eigens entwickelten Messgeräten namens *Energy Data Recorders* (EDRs) [192], [193] synchron und in Echtzeit aufgezeichnet. Die Daten datieren zurück bis Februar 2012, umfassen also zum Zeitpunkt der Verfassung der vorliegenden Arbeit Messdaten über eine Zeitspanne von über drei Jahren. Die Aufzeichnungen fanden jeweils an mehreren Positionen gleichzeitig im Stromnetz des KIT Campus Nord statt. Der Datenbestand vergrößert sich kontinuierlich: Jedes EDR-Gerät zeichnet bei einer Abtastrate von 12,8 kHz<sup>10</sup> und einer Auflösung von 16 Bit täglich

---

<sup>10</sup> Die EDRs erlauben aktuell Sampling-Frequenzen von bis zu 25 kHz, die in der vorliegenden Arbeit u. a. betrachteten Messdaten wurden jedoch hauptsächlich mit 12,8 kHz aufgezeichnet.

rund 8,63 GiB<sup>11</sup> auf und es werden nach und nach zusätzliche Messgeräte installiert. Aktuell beträgt die Größe der Rohdaten über 40 TiB. Zusätzlich kommen vorberechnete Feature-Daten hinzu, welche z. B. durch Fourier-Analyse der Rohdaten generiert werden.

Bisher wurden am IAI erste Korrelations- und Differenz-Analysen kleinerer Teilmengen, z. B. Messungen einzelner Tage, mit gängigen Analyse-Tools (z. B. *Matlab*) durchgeführt, wobei selbst für die verhältnismäßig kleinen Datensätze relativ lange Laufzeiten benötigt wurden. Die einzelnen hochfrequenten Messdaten müssen verwaltet, vorverarbeitet, verglichen und analysiert werden. Die Analyseergebnisse können dann z. B. zur Parametrisierung von Simulationen einfließen. Das langfristige Ziel der Arbeitsgruppe ist eine Modellierung des Stromnetzes auf hohem Detailgrad und unter Berücksichtigung transienter Effekte.

Hierfür werden besondere Verfahren benötigt, die sowohl Auswertungen von kleineren Messreihen (z. B. Messungen eines EDR über einige Minuten) als auch sehr großen Messreihen (z. B. ein Jahr an Messungen mehrerer EDRs) erlauben sollen, also flexibel skalierbar sein müssen. Die Anwendungen umfassen die Übertragung, Archivierung, Verwaltung, Verarbeitung und Analyse sowie die interaktive Visualisierung und Navigation in den EDR-Daten. Außerdem soll im Datenbestand nach bestimmten Kriterien gesucht werden können und es sollen strukturelle Merkmale, wie häufige Muster oder untypisches Verhalten, detektiert werden können, welche beispielsweise mit transienten Spannungseinbrüchen und Stromausfällen einhergehen. Auf diese Datenbasis wird das in der vorliegenden Arbeit entwickelte System angewandt und praktisch erprobt.

## 1.2 Stand der Entwicklung

Nachfolgend wird ein Überblick über den aktuellen Stand der Entwicklung im Bereich der Analyse großskalig verteilt erfasster Messdaten aus Stromnetzen gegeben. Für den detaillierten Entwicklungsstand der einzelnen Teilgebiete der Analyse und Visualisierung von Zeitreihen sowie des Data-Intensive Computing, welche hierzu relevant sind, sei auf Anhang A und B verwiesen. Für eine Beurteilung der Strom- und Spannungsqualität und für ein tiefgehendes Verständnis der komplexen Dynamik in Stromnetzen und der Ursachen von Netzstörungen müssen verteilte hochfrequente Messungen im Stromnetz durchgeführt und ausgewertet werden. Des Weiteren

---

<sup>11</sup> GiB steht kurz für Gibibyte und unterscheidet sich vom Gigabyte, kurz GB, dadurch, dass es Datengrößen als Zweierpotenzen statt als Zehnerpotenzen beschreibt. 1996 wurde vonseiten der IEC der Vorschlag unterbreitet, neue Einheitenvorsätze einzuführen, die ausschließlich in binärer Bedeutung zum Einsatz kommen sollten (siehe ISO-Norm IEC 80000-13:2008 bzw. DIN EN 80000-13:2009-01). Zweierpotenzen sollen durch Binärpräfixe, Zehnerpotenzen hingegen durch SI-Präfixe in dezimaler Bedeutung bezeichnet werden, damit sich sowohl für Zweier- als auch für Zehnerpotenzen eindeutige Bezeichnungen ergeben.

müssen Simulationen auf hohem Detailniveau mit umfassenden, großflächigen Modellen des Stromnetzes durchgeführt werden, welche ebenfalls u. a. große Mengen an Zeitreihendaten produzieren, die nachträglich analysiert werden müssen. Skalierungseffekte spielen in beiden Fällen eine zentrale Rolle, da enorme Datenmengen anfallen können, die sich nur mit speziellen Methoden und unter Zuhilfenahme spezialisierter Rechnerarchitekturen analysieren lassen.

Bei der Analyse von Stromnetz-Messdaten lassen sich, neben der allgemeinen Verwaltung und Organisation, die bei einer großen Zahl von Datenquellen und bei großen Daten ebenfalls nicht trivial ist, zwei Aufgaben unterscheiden, bei denen die Skalierungsaspekte unterschiedlich stark ins Gewicht fallen:

- Große Mengen archivierter Messdaten sollen explorativ analysiert und durchsucht werden, um Regelmäßigkeiten und Muster zu erkennen. Aufgetretene Ereignisse innerhalb des Netzes können so im Nachhinein durch Spezialisten auf abstraktem Niveau untersucht und bewertet und möglicherweise in Zukunft vermieden werden.
- Schnelle operationale Entscheidungen sollen aufgrund der kurzfristigen Erfassung des aktuellen Netzzustands, des Stromangebots sowie der Leistungsnachfrage auf Abnehmerseite getroffen werden. Beispielsweise soll die Netzinfrastruktur aufgrund kurzfristiger Messungen beurteilt werden, erzeugter Strom sofort dorthin gelenkt werden, wo er verbraucht werden kann oder Strompreise entsprechend Angebot und Nachfrage kontinuierlich reguliert werden. Der Fokus liegt hierbei auf schnellen Reaktionszeiten bei der Erkennung von Ereignissen.

Die erstgenannte Aufgabe der Analyse großer historischer Messdaten und die dabei entstehenden Skalierungsprobleme stellen ein aktives Forschungsfeld dar und werden hier genauer betrachtet. Die zweite Aufgabe hängt hingegen eng mit technischen Komponenten der Netzinfrastruktur zusammen und die durchzuführenden Analysen operieren auf verhältnismäßig kleinen Datenmengen, weshalb sie hier nicht thematisiert werden. Für nähere Informationen zu aktuellen Arbeiten in diesem Bereich sei z. B. auf die zahlreichen Veröffentlichungen von Math Bollen u. a. [17], [33], [34], [36], [40], [99], [172] verwiesen.

Die verteilte Akquisition, Archivierung und Analyse von Stromnetz-Messdaten wird bereits seit einigen Jahren im Rahmen verschiedener Projekte thematisiert, um langfristig Instabilitäten des Netzzustandes besser zu verstehen und ihnen operativ entgegenzuwirken. Messungen finden bislang spärlich und nahezu ausschließlich an kritischen Punkten in Übertragungsnetzen sowie in den Hoch- und Mittelspannungsnetzen der Verteilungsnetze statt. Tiefgehende Analysen langfristig archivierter, verteilter Messungen mit hochfrequenter Abtastrate in Niederspannungsnetzen, also den Versorgungs- bzw. Verteilungsnetzen, welche Stromendkunden anbinden, sind bislang in der Literatur nicht bekannt. In Niederspannungsnetzen wird die Strom- und Spannungsqualität bisher aufgrund relativ kurzfristiger Messungen durch Spezialhardware beurteilt. Dabei

wird zwar hochfrequent gemessen, jedoch werden die resultierenden Daten meist zur Auswertung gemäß der Norm *DIN EN 61000-4-30* bzw. *VDE 0847-4-30* aggregiert und nicht über längere Zeiträume gespeichert. Das resultiert zu einem Großteil aus der Tatsache, dass die ansonsten anfallenden hochfrequenten Zeitreihendaten aus vielen Datenquellen aufgrund ihres Volumens nicht mit den bisher eingesetzten Verfahren auswertbar waren.

In Hoch- und Mittelspannungsnetzen existieren Leitstellen, in welchen sog. SCADA-Systeme (von engl. *Supervisory Control and Data Acquisition*) in Kombination mit operativen Energiemanagement Systemen (EMS) eingesetzt werden. Die Systeme haben jedoch niedrige Abstraten und sind somit ungeeignet zur Erfassung der vollen Dynamik in den Verbundnetzen. Außerdem werden die Daten eher stichprobenhaft und nur selten in vollem Umfang ausgewertet und meistens gemittelt bzw. aggregiert [234], [235].

Die *Tennessee Valley Authority* (TVA) [300], eine Körperschaft der Regierung der Vereinigten Staaten von Amerika, beschäftigt sich beispielsweise im Auftrag der *North American Electric Reliability Corporation* (NERC) mit der Messung, Archivierung und Auswertung von PMU-Daten (PMU steht kurz für engl. *Phasor Measurement Unit*, deutsch: *Phasor-Messeinheit*). Das erklärte Ziel der TVA ist es, die Zuverlässigkeit des nordamerikanischen Stromnetzes zu überwachen und langfristig zu verbessern. Mit Hilfe von PMU-Geräten, die an Hochspannungsleitungen und Transformatorstationen angeschlossen sind, werden verhältnismäßig hochfrequente Messungen durchgeführt („einige tausend Mal pro Sekunde“ [290]). Das Gesamtsystem ist groß angelegt und umfasst momentan 19 Unternehmen, 10 verschiedene Hersteller von PMU-Geräten und eine Summe von 103 aktiv messenden PMUs. Die Messdaten werden anschließend aggregiert und archiviert. Sie enthalten GPS-basierte Zeitstempel, Amplitude und Winkel der Spannungen und Stromstärken aller drei Phasen als Positiv-, Negativ- und Null-Sequenzen sowie Änderungsgrad der Frequenz pro Zeiteinheit und zusätzliche Statusindikatoren [290]. Die aggregierten Daten werden über LAN-zu-LAN VPN-Tunnel<sup>12</sup> zu sog. *Super Phasor Concentrators* (SPDC) übertragen und von dort aus an ein zentrales Archivsystem. Ein Verbund aus Servern der TVA leitet die Daten mittels Übertragung gemäß eines Protokollstandards für Phasor-Daten (IEEE C37.118-2005) weiter an Visualisierungssysteme. Es ermöglicht des Weiteren das manuelle Anlegen einer Kopie einzelner Teilmengen der Daten auf einem Hadoop-Cluster, wo Analyseanwendungen aufgebaut werden können. Auf die Daten kann hierzu über eine FTP Schnittstelle bzw. über VPN-Tunnel zugegriffen werden. Auch die TVA kommt nicht umhin, sich mit der Problematik der Skalierbarkeit befassen: Der Bestand an gesammelten PMU-Daten nimmt schnell zu, während

---

<sup>12</sup> LAN-zu-LAN VPN-Tunnel (von engl. *Virtual Private Network*, kurz VPN) ermöglichen eine gesicherte Verbindung zweier lokaler Computer-Netzwerke (von engl. *Local Area Network*, kurz LAN), so dass eine Datenübertragung zwischen Computern stattfinden kann, als wären diese innerhalb desselben Netzes, was physisch jedoch nicht der Fall ist, weswegen die Verbindung als *virtuell* bezeichnet wird..

laufend weitere PMUs integriert werden. Ende 2010 betrug die archivierte Datenmenge etwa 40 TB und es wurde eine Zunahme auf 500 TB innerhalb der darauffolgenden fünf Jahre erwartet (aktuelle Zahlen sind momentan nicht öffentlich verfügbar).

Im Rahmen des durch die *Grid Protection Alliance* (GPA) verwalteten Projekts *Open Source Phasor Data Concentrator* (openPDC) [289] wird eine quelloffene Software zur Echtzeit-Verarbeitung von Zeitreihen aus PMU-Messungen entwickelt. Sie soll die Verarbeitung mehrerer hundert PMU-Datenströme mit Erfassung, Archivierung und interner Sortierung nach GPS-Zeitstempeln ermöglichen. Bei der Entwicklung werden Skalierungs- und Kompatibilitätsaspekte berücksichtigt, indem Schnittstellen für gängige Übertragungsstandards für PMU-Daten entwickelt werden. Die Zeitauflösung des Systems ist jedoch mit einer zehnhundertstel-Erfassungsrate zu niedrig für tiefgehende Analysen wie die Detektion transienter Ereignisse.

Ein weiteres quelloffenes System zur Verarbeitung von PMU-Daten, basierend auf dem IEEE C37.118 Standard ist die Software *iPDC* [125]. Sie beinhaltet einen Datenbank-Server und Module zu PMU-spezifischen Simulationen und richtet sich an Studenten und Wissenschaftler, die sich mit Algorithmen- und Softwareentwicklung bezogen auf PMU-Daten befassen.

Das Software-Projekt *Lumberyard* [190], das sich in einer frühen Entwicklungsstufe befindet, beschäftigt sich mit der Verwaltung und Analyse von Zeitreihen aus Stromnetz-Messungen der TVA mithilfe von Hadoop. Es stellt eine Indizierung der Zeitreihendaten basierend auf einer symbolischen Repräsentation zur Verfügung und speichert den Index in *HBase*, einem spaltenbasierten Speichersystem für Hadoop. Hierfür nutzt *Lumberyard* die Symbolische Aggregat-Approximation [143]. Die Software wird jedoch seit Juli 2011 nicht gepflegt, ist nicht direkt lauffähig und der Hadoop-spezifische Verarbeitungs-Quellcode der TVA ist nicht zugänglich. Trotzdem scheint der Ansatz skalierbar und daher vielversprechend zu sein.

In [328] verwenden die Autoren ebenfalls eine symbolische Darstellung – es handelt sich de facto um die *Symbolische Aggregat-Approximation* (SAX) – um die Dimensionalität großer Mengen von Smart Meter-Daten zu reduzieren und damit klassifizierbar und voraussagbar zu machen. In [163] werden die Möglichkeiten der Kompression zur Reduktion der Datengröße in großen Archiven von Stromqualitätsdaten untersucht und eine von den Autoren vorgeschlagene Technik auf reale Datensätze angewandt. In [155] präsentieren die Autoren eine datenparallele Umsetzung ihrer PDFA-Algorithmus (von engl. *Parallel detrended fluctuation analysis*) auf einem Hadoop-Cluster und wenden diesen zur Detektion auffälliger Oszillationen des Stromnetzes von England, Schottland und Wales an. Die verwendeten Datensätze haben jedoch nur eine Größe von einigen Gigabytes, weshalb die Skalierbarkeit nicht zureichend beurteilt werden kann.



## 1.3 Fragestellung

Die explorative Analyse und Charakterisierung großer semistrukturierter Zeitreihendaten wie der am KIT gemessenen hochfrequenten Strommessungen erfordert neue Möglichkeiten der Verarbeitung, Organisation und interaktiven Visualisierung großer Datensammlungen sowie die Extraktion geeigneter Merkmale, welche als Basis nachgelagerter struktureller und statistischer Analysen und Ähnlichkeitssuchen dienen können. Hierzu muss ermöglicht werden, die Zeitreihendaten in ihrer Gesamtheit zu durchlaufen und zu analysieren.

Für sehr große Datenmengen skalieren die bekannten Methoden jedoch unzureichend mit der Datengröße, was vollständige Analysen derzeit zeitaufwendig und teuer bzw. ressourcenaufwendig macht. In der Praxis werden deshalb große Zeitreihenbestände bisher selten in ihrer Gesamtheit, sondern nur ausschnitthaft untersucht. Es werden neue Methoden für die interaktive Visualisierung von Zeitreihen und Verfahren zur Navigation und Manipulation benötigt, die auf die besonderen Skalierungsaspekte zugeschnitten sind.

Aufgrund dessen besteht ein Bedarf zur Erforschung geeigneter Methoden zur Datenreduktion und Datenrepräsentation und zur Entwicklung neuer skalierbarer Verfahren zur Zeitreihenexploration und vollständigen Analyse verteilt gespeicherter Zeitreihendaten auf allen Detailstufen.

Dazu müssen u. a. folgende Fragen geklärt werden:

- Mit welchen Methoden können große multivariate Zeitreihen analytisch charakterisiert werden und welche Repräsentationen der Daten eignen sich hierfür?

Dazu sind die folgenden Fragen von Interesse:

- Welche Datenrepräsentation eignet sich für strukturelle Analysen und für die Entdeckung von Ähnlichkeitsstrukturen bei unbekanntem Kategorien und inwiefern lassen sich existierende Analyse-Methoden auf diese Datenrepräsentation übertragen?
- Wie kann der zeitliche Verlauf auf verschiedenen Detailstufen kompakt beschrieben werden?
- Wie lassen sich große Zeitreihen vergleichen und durchsuchen?
- Wie können bekannte und unbekannte Muster gefunden werden und wie können große Zeitreihen in semantisch ähnliche Segmente unterteilt werden?
- Wie können große Zeitreihendaten organisiert werden, um skalierbare Auswertungen zu ermöglichen?  
Hierbei stellen sich folgende Teilfragen:
  - Wie können diese Daten trotz ihres Umfangs effizient im Sinne einer Analyse verarbeitet werden?
  - Welche Anforderungen ergeben sich auf Datei- und Architekturebene?

- Wie kann Metainformation aus den Daten extrahiert werden und wie kann damit eine semantische Referenzierung realisiert werden?
- Welche Verfahren und Werkzeuge werden für die explorative Analyse großer Zeitreihen benötigt und wie können diese unter Berücksichtigung der speziellen Skalierungs- und Architekturanforderungen effizient implementiert und integriert werden?  
Dafür sind die folgenden Fragen zu klären:
  - Wie können große multivariate Zeitreihendaten auf eine Weise interaktiv visualisiert werden, welche sowohl einen schnellen Überblick über die Charakteristik der Gesamtheit der Daten ermöglicht, als auch eine interaktive Navigation zu Details?
  - Wie kann eine Zusammenarbeit zwischen Experten bei der Analyse der Zeitreihendaten ermöglicht werden?
- Wie können die erarbeiteten Verfahren als Gesamtsystem leistungsfähig eingesetzt werden und worin liegt ihr besonderer Nutzen?  
Detailfragen hierfür sind:
  - Welche neuen Möglichkeiten ergeben sich für verschiedene Anwendungsbereiche, v. a. für die Analyse der am KIT gemessenen Energiedaten, und wie kann das aufgebaute Auswertungs-System auf diese angewandt werden?
  - Welche neuen Erkenntnisse über Stromnetze ergeben sich mithilfe der vorgestellten Methoden?

## 1.4 Ziele und Aufgaben

Das Ziel der vorliegenden Dissertationsschrift besteht darin, ein neues Konzept für ein skalierbares Datenmanagement und explorative Analysen großer Zeitreihendaten aufzustellen, umzusetzen und anhand umfangreicher Messdaten aus dem Stromnetz des KIT zu erproben und zu evaluieren. Dazu sind die folgenden wissenschaftlichen Teilziele zu bearbeiten:

- Neuer Konzeptentwurf zur skalierbaren Organisation und Auswertung großer Zeitreihendaten
  - Identifikation und Verteilung benötigter Komponenten und Datenflüsse
  - Entwurf einer neuen effizienten datenparallelen Analyse zur Merkmalsextraktion und Datenreduktion
  - Herleitung neuer Metadaten mit dimensionsreduzierten Zeitreihendaten
  - Entwurf einer integrierten Client-Software zur Darstellung, Suche und Exploration von Zeitreihen mit Anbindung an das verteilte Datenverarbeitungssystem
- Skalierbarkeitsbewertung bestehender und Ableitung neuer Zeitreihenanalyseverfahren
  - Ableitung und Auswahl geeigneter Zeitreihenanalyseverfahren zur Umsetzung auf der verteilten Zielarchitektur

- Ableitung interaktiver und Kombination geeigneter Zeitreihenexplorationswerkzeuge
- Neues Konzept zur verteilten Datenorganisation und -verarbeitung
  - Experimentelle Performanzuntersuchung der Architektur und Software zur datenparallelen Verarbeitung
  - Konzept für Schnittstellen und Datenflüsse in einem verteilten Verarbeitungssystem zur Ermöglichung einer integrierten Analyseumgebung
  - Neue und modifizierte Verfahren und Umsetzung als Software-Werkzeuge zur effizienten Organisation großer Datenmengen
- Neues Konzept zur verteilten explorativen Analyse großer Zeitreihendaten
  - Modifizierte Zeitreihenbeschreibung anhand struktureller Merkmale
  - Verteilte Algorithmen zur Merkmalsextraktion und Software-Design für das Zielsystem
  - Metadaten-Konzept für dimensionsreduzierte Zeitreihendaten und abgeleitete Merkmale zur semantischen Referenzierung sowie Umsetzung der datenparallelen Merkmalsextraktion auf dem Zielsystem
  - Modifizierte Algorithmen für ähnlichkeitsbasierte Zeitreihenvergleiche und Anpassung bestehender Distanzmaße
  - Verfahren und Softwarewerkzeuge zur effizienten Suche nach Teilstrukturen in großen Zeitreihendaten und Merkmalsdaten
  - Neue und angepasste Visualisierungswerkzeuge zur interaktiven Zeitreihendarstellung mit integrierter Datenauswahl und -Mustersuche zur verteilten Weiterverarbeitung
  - Design und Umsetzung aller Teilkomponenten und Schnittstellen und Integration zu einem leistungsfähigen Gesamtsystem
- Evaluation der Einzelkomponenten
- Anwendung auf umfangreiche Stromnetz-Messdaten des KIT.

Passend zu diesen Zielen gliedert sich die vorliegende Arbeit in sechs Kapitel, die im Folgenden kurz vorgestellt werden. Nach der erfolgten Einleitung in die Thematik, in welcher Motivation, Problemstellung und Ziele der Arbeit sowie der technische Entwicklungsstand erläutert werden, wird in Kapitel 2 das grundlegende Konzept der Arbeit in Form eines Entwurfs für ein System zur skalierbaren explorativen Zeitreihenanalyse vorgestellt. Die hierzu benötigten Verfahren und Strukturen aus den Bereichen der Zeitreihenanalyse, der Zeitreihenvisualisierung und des Data-Intensive Computings werden aufgrund ihrer Skalierbarkeit bewertet und ausgewählt. Anschließend wird in Kapitel 3 auf Design und Umsetzung des neuen Konzeptes eingegangen und das spezielle System aus Software und Hardware wird dargestellt. Letzteres wird schließlich gemeinsam mit den einzelnen Teilverfahren und Modulen in Kapitel 4 evaluiert. Außerdem wird das

entwickelte System in Kapitel 5 auf zwei Anwendungsfälle aus dem Bereich der Stromnetzanalyse angewandt und die Ergebnisse vorgestellt. Die Arbeit schließt mit einer Zusammenfassung und Schlussbetrachtung in Kapitel 6, in welcher die Ergebnisse zusammengefasst und diskutiert werden und ein Ausblick auf mögliche weitere Forschungsthemen geboten wird.

Im Anhang werden die Bewertung der Skalierbarkeit ausgewählter Verfahren zur Analyse und Exploration von Zeitreihen in Anhang A und der Anwendungsbereich der Stromnetzanalyse, auf den das System angewandt wird, in Anhang B detailliert beschrieben. Hierzu werden anwendungsspezifische Methoden und Systeme dargestellt und es wird auf hochfrequente Strommessungen am KIT und deren Verwaltung eingegangen.

## 2 Neues Konzept zur skalierbaren, explorativen Analyse großer Zeitreihendaten

Dieses Kapitel stellt das im Rahmen der vorliegenden Arbeit aufgestellte Konzept für eine skalierbare Verwaltung und Analyse von Zeitreihendaten vor. Es bezieht sich teilweise auf am KIT gemessene Energiedaten, ist jedoch bewusst sehr allgemein gehalten und auf beliebige Sequenz- und Zeitreihendaten anwendbar. Es beinhaltet Methoden aus den Bereichen des *Data-Intensive Computings*, des *Maschinellen Lernens*, des *Data-Minings* bezogen auf Zeitreihen sowie der *Explorativen Zeitreihenanalyse* bzw. *Zeitreihen-Visualisierung*.

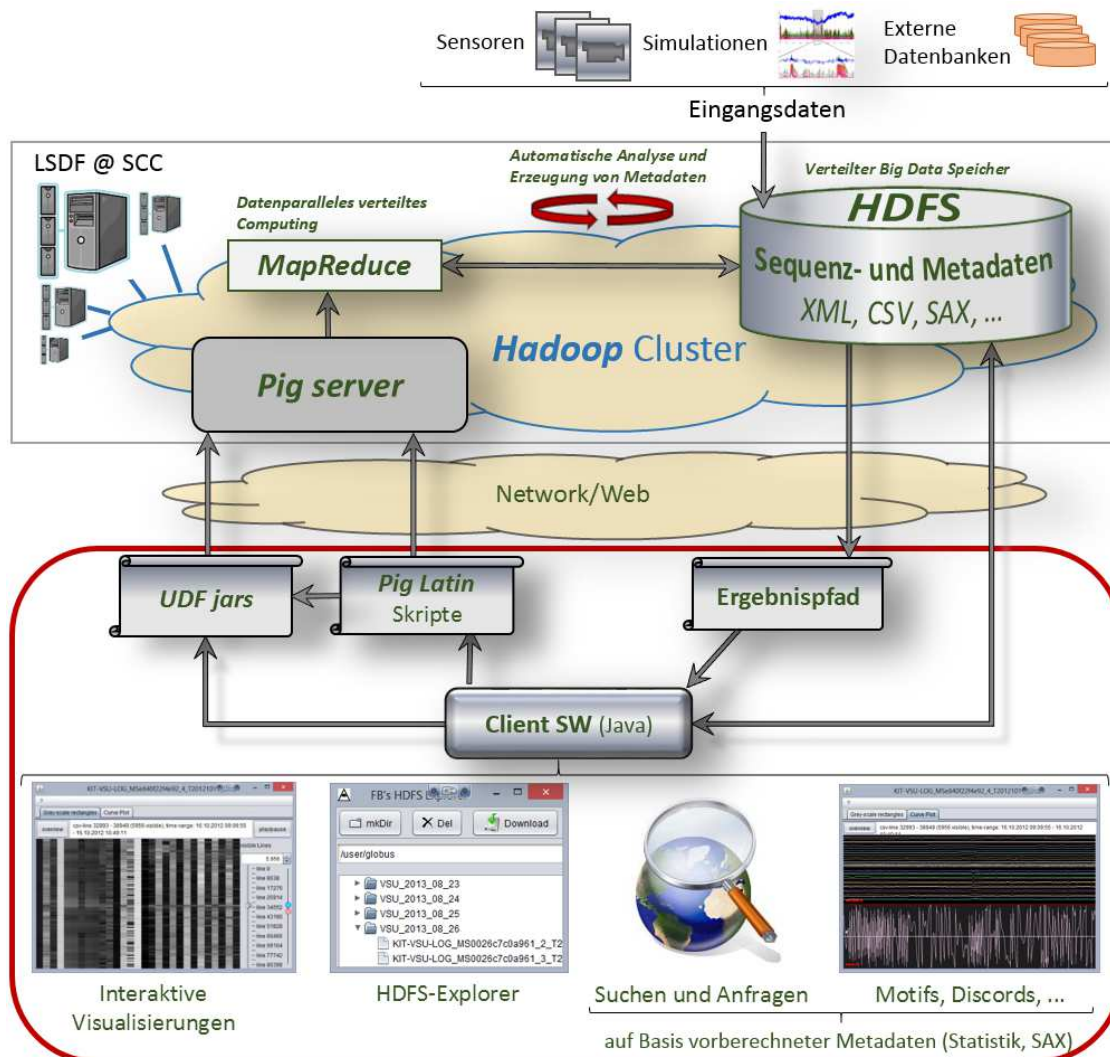
Das Konzept ist in Abbildung 2.1 dargestellt und basiert u. a. auf dem Prinzip der *Datenreduktion*, beziehungsweise der Ableitung weiterverwendbarer *Metadaten* durch datenparallele Verarbeitung auf einer verteilten Infrastruktur (oberer Abschnitt in Abbildung 2.1). Es umfasst skalierbare Methoden zur Verwaltung, Organisation und Verarbeitung sowie zur explorativen Analyse großer Zeitreihendatenbanken durch interaktive Visualisierungen und Suchen, welche als Client-Anwendung umgesetzt werden (unterer Abschnitt in Abbildung 2.1). Die Datenverarbeitungskette zur Ableitung der Metadaten, die oben in Abbildung 2.1 als Kreislauf dargestellt ist, wird in Abbildung 2.2 gezeigt und der Zusammenhang zwischen Anfrageschema und Datenverarbeitung ist in Abbildung 2.3 dargestellt. Beim Entwurf wurde ein besonderer Fokus auf Skalierbarkeit mit der Datengröße und -komplexität gelegt, um die Anwendung auf extreme Mengen an Zeitreihendaten zu ermöglichen, die aktuell in vielen Wissenschafts- und Industriebereichen anfallen und deren Analyse mit bekannten Methoden oft an Skalierungsproblemen scheitert. Daher werden geeignete Methoden und Verfahren auf ihre Skalierbarkeit und Parallelisierbarkeit hin bewertet und eine Auswahl getroffen (Details hierzu finden sich in Anhang A).

### 2.1 Komponenten und Datenflüsse

Eine Übersicht über Komponenten der hierzu benötigten verteilten Infrastruktur sowie die Datenflüsse und Schnittstellen des Systems für Benutzer und Datenquellen ist in Abbildung 2.1 dargestellt. Das Konzept sieht vor, eingehende Daten innerhalb des Hadoop-Clusters der LSDF (siehe Abs. 3.1) verteilt zu speichern, um sie dort unter Berücksichtigung der Datenlokalität effizient datenparallel verarbeiten zu können. Die Voraussetzungen hierfür werden in Abs. 3.1 untersucht

## Abs. 2.1 – Komponenten und Datenflüsse

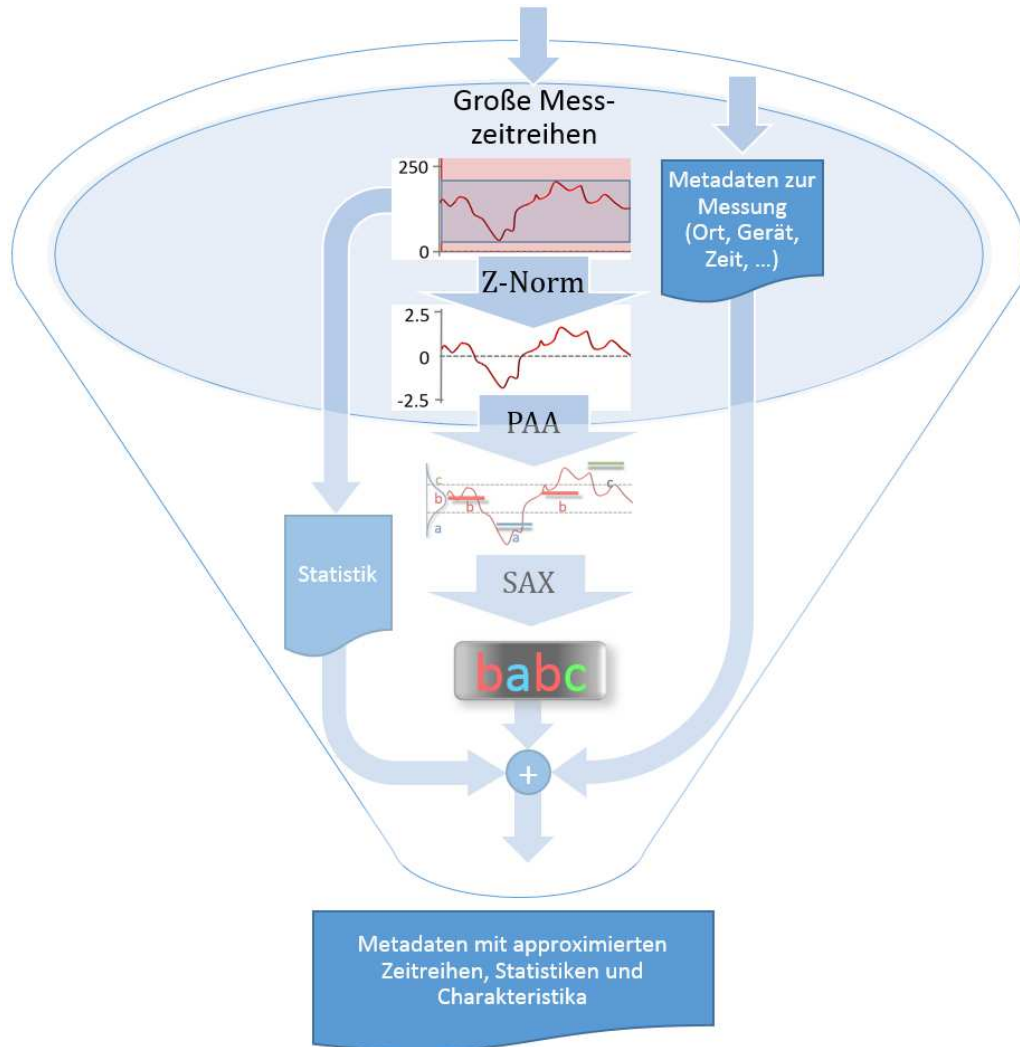
und benötigte Verfahren abgeleitet. Als Dateisystem kommt das horizontal gut skalierbare, Hadoop-spezifische HDFS (von engl. *Hadoop File System*) zum Einsatz, welches Daten in Blöcke gleicher Größe aufteilt und redundant auf Festplatten mehrerer Rechner verteilt speichert. Es ermöglicht hierdurch eine datenparallele Verarbeitung mittels des MapReduce-Programmiermodells, welche in Abs. 3.2 evaluiert und in Abs. 3.4 für die Metadatenableitung genutzt wird.



**Abbildung 2.1:** Konzept des Gesamtsystems –Komponenten der verteilten Datenverarbeitung, Datenflüsse und Benutzerschnittstellen (eigene Arbeiten rot hervorgehoben).

Jedoch birgt die Verwendung des Hadoop-Clusters durch die Verteilung der Daten einige Komplexität bezüglich des Datenzugriffs in sich, z. B. erfordern Dateimanipulationen Spezialwissen und sind bisher nur in Kommandozeilenumgebungen möglich. Daher werden entsprechende Softwarewerkzeuge wie z. B. ein HDFS-Dateiexplorer entwickelt, welche eine Navigation speziell in großen Datenmengen, Dateimanipulationen sowie Dateitransfers in einer benutzerfreundlichen grafischen Oberfläche ermöglichen (siehe Abs. 3.3). Außerdem müssen anwendungsspezifische Zeitreihenanalysen später häufig durch Spezialisten durchgeführt werden, die nicht mit

dem komplexen verteilten System vertraut sind. Daher werden Methoden und Softwarewerkzeuge wie z. B. ein Datenselektionswerkzeug entwickelt, die es erlauben, die in HDFS gespeicherten Zeitreihendaten auf Vollständigkeit zu prüfen und vollständige Daten über lange Messzeiträume auszuwählen, ohne die zugehörigen Einzeldateien manuell zu betrachten (siehe Abs. 5.2).



**Abbildung 2.2:** Konzept zur Datenreduktion großer Messzeitreihen und Ableitung durchsuchbarer Metadaten.

Als Hauptdatenquellen sind rechts oben in Abbildung 2.1 Zeitreihendaten aus Sensoren, Simulationen und externen Datenbanken dargestellt, es sollen aber beliebige weitere Zeitreihen- bzw. Sequenzdaten integriert werden können, was durch eine zu entwickelnde Datenschnittstelle ermöglicht wird. Die teilweise unstrukturierten Eingangsdaten können so umfangreich und strukturell komplex sein, dass keine Exploration bzw. Auswertung mit herkömmlichen Methoden möglich ist. Daher sollen die Rohdaten, wie in Abbildung 2.2 gezeigt, durch neue Methoden der Datenreduktion – wie z.B. der Symbolischen Aggregat-Approximation (SAX, siehe Abs. 2.3.1 und

Anhang A.1.8 zu Details) – und des Data-Minings – wie z.B. *Suffix-Trees*, *Discords* und *Motifs* – zu kleineren Metadaten aufbereitet werden und dadurch für eine explorative Analyse und interaktive Visualisierungen (siehe Abs. 3.6) erschlossen werden.

## 2.2 Datenparallele Auswertung und Merkmalsextraktion

In HDFS übertragene Daten sollen automatisch in Hadoop datenparallel ausgewertet werden können (siehe Abbildung 2.3). Die Datenparallelität und die Berücksichtigung der Datenlokalität bei der Verarbeitung ermöglichen durch eine Verteilung der Auswertung auch die Bearbeitung sehr großer Datensätze und Zeitreihensammlungen. Der Auswertemechanismus wird in Abs. 3.4 beschrieben. In diesem Analyseschritt werden Metadaten erzeugt, die wesentlich kleiner sind als die Originaldaten und über die aufgrund der Reduktion effizient auf die großen Originaldaten zugegriffen werden kann.

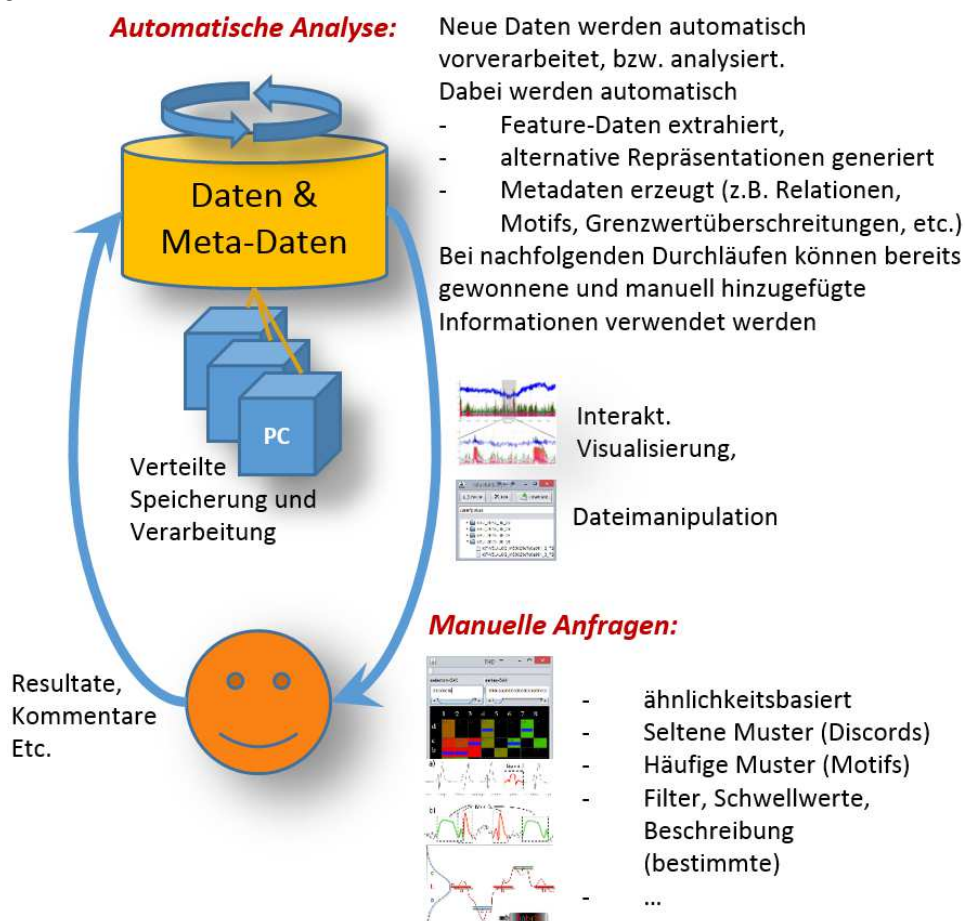


Abbildung 2.3: Automatische Metadatenextraktion und manuelle Suchanfragen.



Die erzeugten Metadaten beinhalten u. a. deskriptive Informationen, etwa über die Herkunft der Originaldaten, wie z. B. Sensor-ID, Zeitpunkt und Ort der Messung etc., statistische Zusammenfassungen von Zeitreihenabschnitten, aber auch dimensionsreduzierte Repräsentationen der Originalzeitreihen (siehe Abs. 3.4.3). Basierend auf diesen verhältnismäßig kleinen Metadaten sind weitere Auswertungen möglich, welche abstrakte strukturelle Charakteristika extrahieren, die den zeitlichen Verlauf der Daten sehr kompakt beschreiben und graphisch und textuell darzustellen vermögen, was in Abs. 3.7.3 beschrieben wird. Auf Basis der Metadaten können Zugriffe und Anfragen realisiert werden, die zuvor für die großen Originalzeitreihen so nicht möglich waren, wie z. B. durch die in Abs. 3.7 beschriebenen Verfahren, die u. a. auch das interaktive Aufspüren ungewöhnlicher und bisher unbekannter Muster, wie z. B. *Discords* ermöglichen.

## 2.3 Metadaten mit dimensionsreduzierten Zeitreihendaten für neuartige Anfragen

Die erzeugten Metadaten sollen eine gegenüber den großen Ursprungsdaten stark komprimierte Beschreibung der Originaldaten enthalten, so dass sie auch auf einem lokalen Client visualisiert und ausgewertet werden können, auch über sehr große Messzeiträume (siehe Abs. 4.5). Sie sollen außerdem eine statistische Beschreibung enthalten (siehe Abs. 3.4.2), die beispielsweise schwellwertbasierte Anfragen erlaubt (siehe Abs. 3.8), aber auch eine Visualisierung vereinfacht.

Dimensionsreduzierte symbolische Zeitreihenrepräsentationen ermöglichen beispielsweise unscharfe inhaltsbasierte Anfragen (siehe Anhang A.1.3.1 und die Umsetzungen in Abs. 3.7.4 und 3.8), abstrakte Strukturbeschreibungen sowie eine darauf basierende semantische Zeitreihensegmentierung (siehe Abs. 3.7.3) und damit eine inhaltliche Sicht auf die Daten. Eine besondere symbolische Repräsentation der Zeitreihen, die sog. Symbolische Aggregat-Approximation (SAX), erlaubt die Anwendung von Methoden aus den Bereichen der Bioinformatik und der Textanalyse, etwa zur Detektion struktureller Auffälligkeiten wie seltener und häufiger Muster (siehe Umsetzungen in Abs. 3.7.6 und 3.7.7) und zur Extraktion grammatikalischer Regeln (siehe Anh. A.1.3.9), was mit anderen Zeitreihenrepräsentationen so nicht möglich ist. Tabelle 2.1 stellt einige Repräsentationen hinsichtlich ihrer Komplexität, Anwendbarkeit, Approximationsgenauigkeit und Besonderheiten gegenüber. Die SAX-Codierung bietet klare Vorteile hinsichtlich der Skalierbarkeit mit der Datenmenge, der flexiblen Parametrisierbarkeit und der Anwendbarkeit für die Aufgabenstellung und soll daher als dimensionsreduzierte Zusammenfassung der Zeitreihendaten in den Metadaten zum Einsatz kommen.

Die Erzeugung der Metadaten erfordert das Lesen, Verarbeiten und Schreiben großer Datenmengen und ist daher sowohl rechen- als auch datenintensiv und kann sequentiell nicht bei akzeptabler Laufzeitkomplexität und -Dauer durchgeführt werden. Deshalb muss dieser Schritt komplett

datenparallel auf dem Hadoop-Cluster ausgeführt werden, um akzeptable Verarbeitungszeiten zu erreichen (siehe Abs. 3.4.1).

**Tabelle 2.1:** Relevante Eigenschaften im Data-Mining gebräuchlicher Zeitreihenrepräsentationen, Angaben teilw. aus [258] (identisch zu Tabelle A.2).

	Komplexität	Anwendung	Approximation	Besonderheiten
APCA	?	vielseitig einsetzbar, robust	je nach Parameterwahl und Signal	geeignet für energiereiche Signale, zwei Koeffizienten pro Segment, insgesamt aber wenige Koeffizienten
DFT	$O(n \log(n))$	Kompression, Periodizitätsanalyse, Signalbereinigung	sehr gut für viele oder energiereichste Koeffizienten, hinreichend für die ersten $n$ . Nicht auf alle Signale anwendbar (stark unperiodische)	<b>gute Genauigkeit</b> , manche Signale erfordern viele Koeffizienten für gute Approximation, <b>Indizierung komplex</b>
H	$O(n)$	Kompression, Periodizitätsanalyse	unter Einschränkungen einstellbar	einfachste Wavelet-Transformation, hierarchisch
PAA	$O(Nm)$ bei $N$ PAA-Segmenten und $m$ Sliding Windows	vielseitig einsetzbar, robust	Zeitlich beliebig einstellbar	einfach, <b>wenige Parameter</b>
PLA	$O(n \log(n))$	Signal-bereinigung, Trend-analyse	einstellbar	<i>Inkrementeller</i> Algorithmus, Fehler berechenbar
SAX	$O(PAA) + O(n)$	vielseitig einsetzbar, robust	<b>flexibel einstellbar</b>	<b>diskret (symbolisch)</b> , <b>wenige Parameter</b> , analytische Verfahren aus Textanalyse & Bioinformatik <b>nutzbar</b> (Markov Modelle, Suffix-Bäume, Regelextraktion, ...)
SVD	$\min \left\{ O(M^2 n), O(Mn^2) \right\}$ bei $M$ Sequenzen der Länge $n$	vielseitig, Suchen (z. B. PageRank)	optimal	nicht für einzelne Sequenzen, Neuberechnung nach Hinzufügen neuer Sequenzen notwendig

### 2.3.1 Zeitreihencodierung mit der Symbolischen Aggregat-Approximation (SAX)

In den letzten Jahren wurden viele neue Verfahren vorgeschlagen, welche durch die zunehmende Anwendung von Methoden des Data-Mining und des maschinellen Lernens für die strukturelle Analyse großer Datenmengen motiviert sind. Sie operieren im Gegensatz zu klassischen Verfahren der Zeitreihenanalyse nicht direkt auf reellwertigen Sequenzen sondern auf Symbolsequenzen. Viele dieser Verfahren stammen aus dem Umfeld der Biowissenschaften – spezifisch zur Analyse von DNA-Sequenzen bzw. Proteinen - und aus dem Bereich der Text-Analyse. Obgleich die Methoden nicht direkt auf die Analyse von ZR übertragbar sind, da sie nur für diskrete Daten definiert sind, existieren geeignete Zeitreihenrepräsentationen, die eine Anwendung derartiger Verfahren erlauben. Diese Ansätze basieren auf einer dimensionsreduzierten diskreten Repräsentation der Daten. Hierdurch sollen v. a. viele Aufgaben der ähnlichkeitsbasierten Zeitreihenanalyse (siehe Abs. A.1.3) effizienter durchführbar werden. Beispiele sind die Einordnung, bzw. Klassifikation, die Gruppierung (*Clustering*) und Indizierung von Zeitreihen für schnelle Zugriffs- und Suchverfahren, semantische Segmentierung sowie die Detektion häufiger und seltener Muster (sog. *Motifs* und *Discords*), vergl. [139], [244] und [86]. Für diese Analyseaufgaben wird ein einfach berechenbares Ähnlichkeitsmaß benötigt, das sich auf die Struktur der Zeitreihe bezieht. Es sollte dabei möglichst ähnliche Ergebnisse wie die Euklidische Distanz liefern und diese möglichst unterschreiten, um z. B. bei Suchanfragen keine falschen Negativen, sondern im schlechtesten Fall nur durch die Vergrößerung falsche Positive zurückzuliefern. Ein junger Vertreter dieser neuen Datenrepräsentationen für Zeitreihen ist die auf der *Paarweisen Aggregat-Approximation* (PAA) [151] aufbauende *Symbolische Aggregat-Approximation* (SAX) [176], die im Folgenden kurz erklärt wird. Weitere Details finden sich im Anhang Abs. A.1.8.

Die Idee hinter SAX ist es, die Dimensionalität auf der Zeitskala durch PAA und auf der Werteskala<sup>13</sup> durch Bereichsaufteilung und Symbolisierung zu reduzieren, wie in Abbildung 2.4 dargestellt. Für die Transformation einer reellwertigen Zeitreihe in die SAX-Repräsentation müssen dazu stets die folgenden drei Schritte durchgeführt werden:

1. *Abschnittsweise Aggregat-Approximation (PAA)*
2. *Z-Normalisierung*
3. *Aufteilung des Wertebereichs in möglichst gleich häufige Intervalle und Symbolzuweisung*

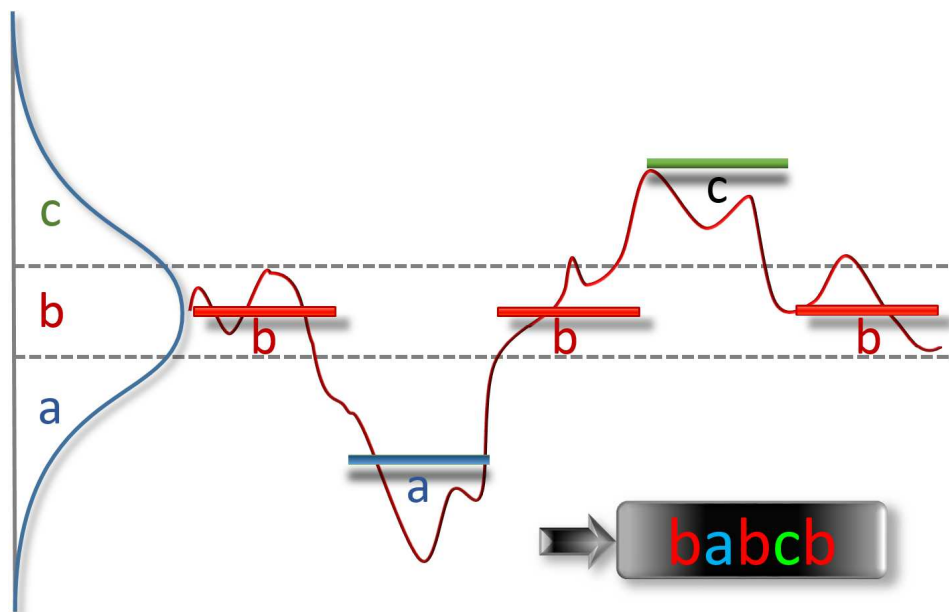
Zunächst wird eine *Paarweise Aggregat-Approximation* (PAA) durchgeführt (siehe Abs. A.1.7), die einer zeitlichen Aggregation entspricht (siehe Abs. A.1.1): Eine Zeitreihe  $X$  der Länge  $n$  wird

---

<sup>13</sup> Man spricht in diesem Zusammenhang von der *Kardinalität* der Zeitreihe. Der Begriff meint in der Mathematik die Anzahl der Elemente einer Menge.

in einen Vektor  $\bar{X} = (\bar{x}_1, \dots, \bar{x}_M)$  der Länge  $M$  überführt. Dadurch wird also zunächst die Dimensionalität von  $n$  auf  $M$  reduziert, indem für die Zeitreihe  $M$  Mittelwerte über gleich große aufeinanderfolgende Teilabschnitte gebildet werden.

Im nächsten Schritt wird die resultierende Zeitreihe normalisiert, so dass sich ein Mittelwert von 0 und eine Standardabweichung von 1 ergeben. Dieses Vorgehen wird häufig als *Z-Normalisierung* (von engl. *zero mean, unit variance*) bezeichnet. Es vereinheitlicht den Wertebereich zu vergleichender Zeitreihen und entfernt dabei automatisch Verschiebungen im Wertebereich (*Shift*) und unterschiedliche Skalierungen. Differenzen zwischen zwei Zeitreihen können damit weitgehend unabhängig von der jeweiligen Lage im Werteraum angegeben werden.



**Abbildung 2.4:** Codierung einer Kurve in SAX-Symbole anhand eines Beispiels. Quelle: Eigene Darstellung, in Anlehnung an [176].

Als letzter Schritt wird der Wertebereich um den Mittelwert der PAA-Zeitreihe herum in eine definierte Anzahl von Bereichen aufgeteilt, die wiederum Symbolen zugeordnet werden. Je nach Anwendung sind hier verschiedene Alphabete denkbar. In der ursprünglichen Veröffentlichung von SAX [176] wird eine Wertaufteilung gewählt, welche zu einer in etwa gleichen Auftretswahrscheinlichkeit aller Zeichen des Alphabets führt, sofern die ursprüngliche Zeitreihe näherungsweise gaußverteilt ist<sup>14</sup>. Dazu wird die Fläche unter einer Gaußkurve in gleich große Bereiche zerlegt. Die Anzahl der Bereiche entspricht der Größe des gewählten Alphabets. Der in Abbildung 2.4 dargestellte Mechanismus ermöglicht die Verwendung fester Nachschlagetabellen für

<sup>14</sup> Nach [176] weisen die meisten Zeitreihen eine Normalverteilung der Einzelwerte auf. Auch Zeitreihen, die andere Verteilungen aufweisen, können jedoch problemlos mit der Methode codiert werden.

die sog. Umbruchpunkte (engl. *Break Points*), was die Codierung schnell im Vergleich zu Methoden macht, bei denen die Tabelle für jede neue Zeitreihe dynamisch berechnet werden muss.

### 2.3.2 Ähnlichkeitsbewertung von Zeitreihen mit dem SAX-Distanzmaß MINDIST

SAX hat den entscheidenden Vorteil, dass ein Distanzmaß hierfür so definiert werden kann, dass es die sog. Kontrahierende Eigenschaft (engl. *Contractive Property* oder auch oft *Lower-Bounding Condition*) besitzt. Dies bedeutet, dass die Transformation  $T$  zusammen mit dem Abstandsmaß dazu führt, dass der Abstand  $D$  zwischen zwei Zeitreihen  $X$  und  $Y$  im Merkmalsraum  $M$  (engl. *feature space*) nie größer ist als der „tatsächliche“ Abstand (z. B. der Euklidische Abstand), dass also gilt:

$$D_{feature}(T(X), T(Y)) \leq D_{euklid}(X, Y) \quad (2.1)$$

Diese Eigenschaft wurde laut [258] von den Autoren des ersten veröffentlichten Ansatzes für eine Indizierung von Zeitreihen [1] erstmals erwähnt, als Beweis der korrekten Funktionsweise ihres Algorithmus. Agrawal, Faloutsos und Swami setzten für ihr Paper von 1993 DFT zur Reduktion der Dimensionalität ein, indem nur die ersten drei<sup>15</sup> Fourier-Koeffizienten als Merkmalsraum betrachtet wurden und führten eine auf dem sog. *F-index* basierende Ähnlichkeitssuche ein. Bei der Evaluation der Suche bemerkten sie, dass alle falschen Resultate falsche Positive waren aber keine falschen Negativen auftraten. Es konnte bewiesen werden [1], dass die Distanz im Merkmalsraum die Euklidische Distanz stets so annähert, dass letztere immer unterschritten wird (engl. *Lower Bounding*).

In der Erstveröffentlichung zu SAX [176] wird ein Ähnlichkeitsmaß namens *MINDIST* definiert. Die Definition von *MINDIST* für die SAX-Zeitreihen  $\bar{X}$  und  $\bar{Y}$  sieht folgendermaßen aus:

$$MINDIST(\bar{X}, \bar{Y}) \equiv \sqrt{\frac{n}{N} \sum_{i=1}^N (dist(\bar{x}_i, \bar{y}_i))^2} \quad (2.2)$$

Aufgrund der Z-Normalisierung während der SAX-Codierung kann die Distanz zwischen zwei SAX-Symbolen einfach einer vorberechneten Nachschlagetabelle  $dist(x, y)$  entnommen werden, ähnlich der Tabelle mit sog. Umbruchpunkten für die Symbol-Codierung. Dies ermöglicht schnelle Ähnlichkeitssuchen.

---

<sup>15</sup> Die ersten drei Fourier-Koeffizienten sind in realen Signalen, z. B. aus Messungen von Prozesseigenschaften, stets die signifikantesten, weshalb die Beschreibung des Ursprungssignals allein durch diese drei Koeffizienten das Originalsignal ausreichend gut approximiert.

Die Distanztabelle sieht für ein Alphabet  $A = \{a, b, c\}$  folgendermaßen aus:

**Tabelle 2.2:** SAX-Distanztabelle für ein Alphabet mit drei Symbolen.  
Entspricht Tabelle A.3

	$a$	$b$	$c$
$A$	0	0	0.86
$B$	0	0	0
$C$	0.86	0	0

Man beachte, dass der Abstand zwischen zwei benachbarten Symbolen stets gleich Null ist. Genau diese intuitiv falsche Unschärfe führt dazu, dass der Euklidische Abstand stets unterschritten wird, da somit auch der PAA-Abstand unterschritten wird, welcher selbst wiederum stets kleiner als der Euklidische Abstand ist. Diese sog. Kontrahierende Eigenschaft wurde von Lin u. a. gezeigt [181]; außerdem wurde sie von Ding u. a. detailliert untersucht [72] und insbesondere für nicht-periodische Zeitreihen wurde die Genauigkeit als besser bewertet als die des DFT-Ansatzes.

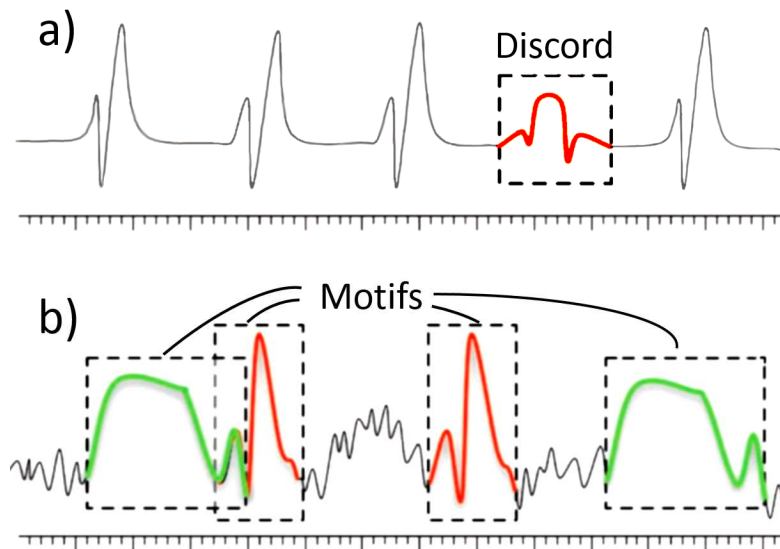
*MINDIST* ist keine echte Metrik der Distanz, sondern als Ähnlichkeitsmaß zu betrachten, da es lediglich die Symmetrieeigenschaft ( $d(x, y) = d(y, x)$ ) erfüllt, nicht jedoch die folgenden beiden Voraussetzungen für Distanzmaße:

- die Identitätseigenschaft ( $x = y \stackrel{!}{\Leftrightarrow} d(x, y) = 0$ ) und
- die Dreiecksungleichung ( $d(x, z) \leq d(x, y) + d(y, z)$ ),

die besagt, dass eine Dreiecksseite höchstens so lang wie die Summe der beiden anderen Seiten sein darf.

### 2.3.3 Detektion seltener und häufiger Muster – Discords und Motifs

Das Auffinden seltener Muster innerhalb einer Zeitreihe ist eine wichtige Analyseaufgabe, die auf Basis der vorberechneten Metadaten, insbesondere der SAX-Codierung durchführbar werden soll. Abbildung 2.5 a) zeigt ein ungewöhnliches Muster (Discord [143]) innerhalb einer Zeitreihe mit ansonsten periodischer Wiederholung eines charakteristischen Musters und Abbildung 2.5 b) zeigt häufige Muster (Motifs). Bislang werden Zeitreihendaten meist von Fachpersonal manuell überprüft, gängige automatische Detektionsverfahren berücksichtigen lediglich Einzelwerte und ob sie außerhalb eines definierten Normbereichs liegen. Letzteres stellt jedoch keine Form der Detektion von Discords, sondern der vergleichsweise trivialen Detektion von Ausreißern dar.



**Abbildung 2.5:** a) Idealisiertes ungewöhnliches Muster (*Discord*) in periodischer Zeitreihe und b) wiederkehrende Muster (*Motifs*). Identisch zu Abbildung A.3.  
Quelle: [79], modifiziert.

Um weitreichendere Analysen zu ermöglichen, wie etwa „ungewöhnliche“ *strukturelle* Muster ohne Personal in Messdaten zu detektieren, was notwendig wird, sobald die Anzahl und Länge auszuwertender Zeitreihen zunimmt oder keine Referenzwerte vorliegen, wird ein automatisches Verfahren benötigt. Eine besondere Schwierigkeit besteht im Umgang mit überlappenden Teilsequenzen, da man eindeutige Fundstellen eines jeden Discords als Resultate erhalten möchte. Außerdem dürfen kleine lokale Unterschiede sowie Rauschen und Ausreißer nicht überbewertet werden. Die Aufgabe der Detektion von Discords kann folgendermaßen formuliert werden [79]:

*Finde für alle Teilsequenzen  $Z'$  der Länge  $l$  einer Zeitreihe  $Z$  der Länge  $n$  mit  $n \geq l$  diejenigen  $m$  Teilsequenzen  $Z'_a$  und deren Positionen in  $Z$ , die sich am meisten von allen anderen Teilsequenzen unterscheiden.*

In [143] wird eine Berechnung von Discords vorgestellt, die lediglich die Teilsequenzlänge  $l$  als Parameter benötigt. Aus der resultierenden nach Unterschiedlichkeit geordneten Liste der Teilsequenzen kann dann die gewünschte Anzahl  $m$  entnommen werden. Darauf aufbauend wurde in [338] ein exakter Algorithmus zur Detektion von Discords fester Länge vorgestellt, welcher die Zeitreihe lediglich zweimal sequentiell durchlaufen muss und sich daher gut für große Datensätze eignet. Dieser Ansatz basiert auf der in Abs. 2.3 gewählten SAX-Repräsentation, die in Abs. 2.3.2 erläutert wird und kann so im Rahmen des hier erarbeiteten Konzepts optimal eingesetzt werden. Die Algorithmen werden außerdem zur Anwendbarkeit auf großen Zeitreihendaten hinsichtlich der Skalierbarkeit optimiert (siehe Umsetzung in Abs. 3.7.6 und Evaluation in Abs. 4.3).

Eine ähnliche Aufgabe stellt das Auffinden häufig wiederkehrender Muster – sog. *Motifs* – innerhalb einer Zeitreihe dar. Diese Analyseaufgabe wurde durch Gensequenz-Analysen im Bereich der Bioinformatik motiviert, wo es u. a. darum geht, unbekannte Verwandtschaftsbeziehungen aufgrund von Gemeinsamkeiten im Erbgut aufzudecken. Sie wird in dieser Arbeit auf Zeitreihen übertragen angewandt. Der Vorgang kann aufbauend auf [79] formal definiert werden als:

*Finde für eine gegebene Zeitreihe Z alle Teilabschnitte Z', die mehrmals in Z vorkommen und deren Positionen in Z.*

Ein Ansatz für die Suche nach Motifs wurde 2002 erstmalig in [180], [231] (derselben Autoren) als Suche nichtüberlappender „typischer“ Teilsequenzen beschrieben. Durch die Entdeckung der fehlenden Aussagekraft eines Clusterings reellwertiger Teilsequenzen 2003 durch Keogh u. a. (spätere Publikation als Zeitschriftenartikel in [153]) erfuhr das *Motif-Mining* besondere Aufmerksamkeit, da die Autoren darauf hinwiesen, dass Motifs zu einem sinnvollen Clustering beitragen können [79]. Außerdem können Motifs gefunden werden, die das „Normalverhalten“ vieler Zeitreihen beschreiben, wodurch eine vereinfachte Modellierung und eine semantische Segmentierung ermöglicht werden kann. Die Aufgabe der Zeitreihenklassifikation kann zudem gemäß [341] beschleunigt werden, indem klassenspezifische Motifs ermittelt und verwendet werden [79].

Nähere Details zur Detektion von Discords und Motifs sowie weiteren Analyseaufgaben finden sich zusammen mit einer Skalierbarkeitsbewertung in Anh. A.1.3.

## 2.4 Client-Software mit Hadoop-Schnittstelle

Eine zu entwickelnde Client-Software mit Schnittstelle zum Hadoop-Cluster über LAN bzw. Internet ermöglicht Benutzern eine intuitive Interaktion mit den Meta- und Originaldaten auf unterschiedlichen Abstraktionsebenen. Sie erlaubt interaktive explorative Analysen der Originaldaten durch diverse Visualisierungen (siehe Abs. 3.6), entfernte Datenmanipulationen und -Übertragungen (siehe Abs. 3.3), unscharfe Suchen und Anfragen über die vorberechneten Metadaten (siehe Abs. 3.7 und 3.8) sowie das Anstoßen verteilter Datenverarbeitung bzw. spezieller Analysen auf dem Hadoop-Cluster. Dabei sollen auch sehr große Zeitreihendaten betrachtet werden können, weshalb alle Teilmodule auf optimale Skalierbarkeit mit der Datenmenge und der Anzahl anzuzeigender Elemente ausgelegt werden.

Optional besteht die Möglichkeit, Teilmengen der Daten – v. a. der Metadaten, aber auch der Originaldaten – auf einen lokalen Client-Rechner zu übertragen, um sie dort zu analysieren, sofern eine lokal verarbeitbare Datenmenge nicht überschritten wird. Des Weiteren sollen getrennt von den zuvor genannten Metadaten fachspezifische strukturierte Kommentare von Experten zu



markierten Bereichen in den Zeitreihen erstellt werden (siehe Abs. 3.6.3) und optional auf den Hadoop-Cluster übertragen werden können, wo sie dann ausgewertet werden können und allen Nutzern zur Verfügung stehen. Die Markierungen und Kommentare können später für eine semantische Anreicherung der Visualisierung und der Ergebnisse einer inhaltsbasierten Ähnlichkeitssuche verwendet werden, sofern sie sich auf Bereiche beziehen, die ähnliche Charakteristika wie die gesuchten Daten aufweisen. Damit können prinzipiell durch Experten bereits gefundene und erklärte Auffälligkeiten in den Daten auf ähnliche Fälle übertragen werden, indem auf entsprechende Stellen zusammen mit den passenden Kommentaren verwiesen wird.

## 2.5 Evaluation und Anwendung auf hochfrequente Stromnetz-Messdaten

Das fertige Gesamtsystem und seine Einzelkomponenten werden in Kapitel 4 experimentell evaluiert, indem speziell vorbereitete Testdaten analysiert werden. Dabei wird die Leistungsfähigkeit herausgestellt, wobei der Skalierbarkeit ein besonderes Augenmerk zukommt.

Durch Anwendung des fertigen Systems auf umfangreiche hochfrequente Stromnetz-Messdaten des KIT in Kapitel 4 wird gezeigt, dass hiermit datenintensive Analysen durchgeführt werden können, die zuvor in diesem Bereich nicht effizient möglich waren. Des Weiteren wird nach anwendungsrelevanten Merkmalen und Auffälligkeiten in den Daten gesucht, die aufgrund aktuell eingesetzter Verfahren, welche auf der Norm *DIN EN 61000-4-30* bzw. *VDE 0847-4-30* beruhen, nicht skalierbar zu detektieren sind.



# 3 Design neuer skalierbarer, explorativer Verfahren zur Analyse von Zeitreihen als verteiltes Softwaresystem

In diesem Kapitel wird eine konkrete Umsetzung des in Kapitel 2 vorgestellten Konzepts für eine skalierbare Zeitreihenanalyse namens *FraScaTi* (von engl. **F**ramework for **S**calable **T**ime series **A**nalysis) vorgestellt. Hierzu werden zunächst die Voraussetzungen für eine Umsetzung, die verwendete technische Infrastruktur und die benötigte Funktionalität erarbeitet und dargestellt, um dann die Funktions- und Leistungsfähigkeit der neu entwickelten und modifizierten Verfahren und Methoden mithilfe einer prototypischen Implementierung auf Tauglichkeit und Skalierbarkeit zu testen. Hierauf folgt eine Darstellung der Entwicklung der Verfahren und Teilkomponenten und des Zusammenspiels als Gesamtsystem. Dabei wird insbesondere auf die Skalierungsproblematik bei Verwaltung, Exploration und datenparalleler Verarbeitung großer Zeitreihendaten in einer verteilten Infrastruktur eingegangen.

## 3.1 Voraussetzungen für verteilte Zeitreihenanalyse

Die wichtigste Voraussetzung für eine datenparallele verteilte Verarbeitung von Zeitreihendaten ist eine geeignete Infrastruktur, welche die Prinzipien der Datenlokalität und der Minimierung von Datentransfers umsetzt und außerdem geeignete Programmierschnittstellen bereitstellt, über welche eigene Client-Software die verteilte Datenverarbeitung effektiv steuern kann. Basierend auf einer bestehenden Installation von Hadoop wurde im Rahmen der vorliegenden Arbeit ein System zur Anwendung von DIC zur Zeitreihenanalyse und -Exploration entwickelt. Hierzu wurde zunächst lokal ein Rechencluster bestehend aus drei Knoten aufgesetzt, um sich mit der Installation und dem Betrieb von Hadoop vertraut zu machen und in einer selbst administrierten Umgebung experimentieren zu können. Anschließend wurde ein größeres Clustersystem mit 58 Knoten genutzt, welches als Teil der *Large-Scale Data Facility* (LSDF) betrieben wird, die am *Steinbuch Centre for Computing* (SCC) am Karlsruher Institut für Technologie (KIT) eingerichtet wurde (siehe Abs. 3.1.1). Die folgenden Beschreibungen des Hadoop-Clusters und der damit zusammenhängenden Komponenten der LSDF beziehen sich auf die Konfiguration jenes Clusters während des Zeitraums der Erarbeitung des *FraScaTi*-Systems, also auf den Stand von 2012 bis 2014. Dieser Hadoop-Cluster stellt die Infrastruktur und Architektur dar, welche für die Entwicklung eines skalierbaren Zeitreihenanalysesystems zur Verfügung stand. Mittlerweile wurde die

Konfiguration des Hadoop-Clusters mehrfach geändert, woran die entwickelte Software angepasst wurde, um die Lauffähigkeit in der neuen Umgebung zu gewährleisten und neu entstandene Möglichkeiten zu nutzen.

### 3.1.1 Die Large Scale Data Facility (LSDF)

Die LSDF wurde durch das Steinbuch Centre for Computing eingerichtet, um Dienste zur Speicherung und Verarbeitung großer Mengen wissenschaftlicher Rohdaten bereitzustellen. Für den vollständigen Aufbau der LSDF siehe z. B. [282]. Die Problematik der notwendigen und häufig vorgeschriebenen Langzeitarchivierung solcher Daten stand beim Aufbau der LSDF im Vordergrund. Entsprechende Anwendungsperspektiven werden z. B. in [88], [89], [204], [277], [278] beschrieben. Es wurde aber auch das Potential der sog. Datenwissenschaften zur zukünftigen Wissensextraktion aus großen Sammlungen wissenschaftlicher Daten (vergl. [109]) erkannt, wozu spezielle Methoden und Architekturen benötigt werden, welche sich für verteilte Auswertungen sehr großer Datenmengen (also DIC) eignen.

Es sollten auch Architekturen für HPC zur Ermöglichung aufwendiger Berechnungen geschaffen und für wissenschaftliche Projekte zur Verfügung gestellt werden, bei welchen große Datenmengen produziert oder analysiert werden. Der Fokus lag dabei jedoch auf rechenintensiver Verarbeitung. Um auch erste experimentelle Erfahrungen mit datenintensiver Verarbeitung unter Berücksichtigung der Datenlokalität zu sammeln, wurde auch ein Cluster mit der damals noch in einem relativ frühen Entwicklungszustand befindlichen Software Apache Hadoop eingerichtet.

Für den Aufbau der LSDF wurde u. a. auf die Erfahrungen mit der Datenorganisation für den *Large Hadron Collider* (LHC) der Europäischen Organisation für Kernforschung (CERN) [288], [199] und für das *Grid Computing Centre Karlsruhe* (GridKa) [97], [200], das deutsche *Worldwide LHC Computing Grid* (WLCG) Tier 1 Zentrum, zurückgegriffen. Die Motivation der Einrichtung der LSDF wird in [282] beschrieben. Mittlerweile stellt die LSDF auch die groß angelegten Online-Speicherdienste *bwSync&Share* und *bwFileStorage* für Benutzer und Institute des öffentlichen Dienstes in Baden-Württemberg bereit. Relevant für die Entwicklung des hier vorgestellten Softwaresystems unter Verwendung des Hadoop-Clusters waren v. a. die folgenden Komponenten der LSDF.

Hardware und Dienste der LSDF:

- Rechner-Infrastruktur (10 Racks, Kühlung, Management-Netzwerk)
- LAN-Netzwerke
  - SCC Basisnetzwerk
    - 10Gb/s redundanter Backbone

- Datenerfassungs-Netzwerk (10Gb/s)
  - angeschlossene KIT-Institute<sup>16</sup>: ITG, IPE, IAI, ANKA, GPI
- Datei-Server und Speicher
  - DDN → 750 TB (550 TB nutzbar), 8 Server
  - IBM → 2 PB (1.4 PB nutzbar), 6 Server
- Software-Dienste
  - NFS CIFS HTTP GPFS , CIFS, HTTP, GPFS, SoFS, Matlab, Hadoop, Integration mit KIT ADS
- Computing mit Hadoop
  - 58 Knoten mit je 8 Kernen, 36 GB RAM (2 Knoten mit 100 GB)
  - Direktanschluss an Speicher (GPFS/DDN)
  - 110 TB HDFS.

### 3.1.2 Der Hadoop-Cluster der LSDF am KIT

**Architektur und Hardware:** Der Hadoop-Cluster der LSDF besteht aus einem Verbund von 58 Rechenknoten mit insgesamt 464 physikalischen Rechenkernen. Jeder Knoten ist über das Netzwerk in die Infrastruktur der LSDF eingebunden und hat folgende Spezifikationen:

- 2 Intel Xeon CPU E5520, 2.27 GHz mit je 4 Kernen, vierfaches Hyperthreading (16 effektive Kerne)
- 36 GB RAM
- 2 TB Festplattenspeicher
- 1 Gigabit Ethernet (GigE) Netzwerkkadappter
- OS Scientific Linux 5.5 mit Linux Kernel 2.6.18.

Die Koordination des Clusters übernehmen zwei Namensknoten mit folgenden erweiterten Ressourcen:

- 2 Intel Xeon CPU E5520, 2.27 GHz mit je 4 Kernen, vierfaches Hyperthreading (16 effektive Kerne)
- 96 GB RAM
- 10 GE Gigabit Ethernet (GigE) Netzwerkkadappter
- OS Scientific Linux 5.5 mit Linux Kernel 2.6.18.

---

<sup>16</sup> Abgekürzte Institute des KIT: Institut für Toxikologie und Genetik (ITG), Institut für Prozessdatenverarbeitung und Elektronik (IPE), Institut für Angewandte Informatik (IAI), Angströmquelle Karlsruhe (ANKA) und Geophysikalisches Institut (GPI).

**Hadoop-Distribution und verfügbare Schnittstellen:** Die installierte Hadoop-Version war zunächst eine Cloudera-Hadoop (CDH) Distribution in der Version `Hadoop 0.20.1-cdh3u4` und wurde 2013 auf `Hadoop 0.20.2-cdh3u5` aktualisiert. Erst gegen Ende 2014 wurde ein neuer Cluster mit `Hadoop 2.5.0-cdh5.2.0` (u. a. auf Anraten des Autors) eingerichtet. Somit war die zur Verfügung stehende Version stets einige Jahre älter als die jeweils aktuelle Version und wurde teilweise nicht mehr offiziell von Cloudera unterstützt, was bei der Softwareentwicklung berücksichtigt werden musste. Außerdem waren nur wenige der neu hinzugekommenen Erweiterungen des „Hadoop-Ökosystems“ mit der vorhandenen Version kompatibel und konnten nicht genutzt werden. Es gab am SCC kaum verfügbares Personal für einen Support des Hadoop-Clusters, aus Sicherheitsgründen wurden jedoch auch keine administrativen Rechte für die kleine Zahl an Hadoop-Nutzern eingeräumt. Deshalb standen die in einer CDH-Umgebung üblichen Web-Benutzerschnittstellen, insbesondere *Apache Hue*, aufgrund fehlender Konfiguration leider nicht zur Verfügung. Entsprechende Funktionalität für die zur Verfügung stehende Umgebung wurde daher selbst entwickelt. Die im Folgenden vorgestellte Softwareentwicklung basiert zum größten Teil auf der Java-Hadoop-API in der Version `Hadoop 0.20.2-cdh3_u5`.

### 3.1.3 Benötigte Funktionalität und Softwaremodule

Die Hauptanwendung des zu entwickelnden Systems ist eine skalierbare Datenanalyse mit einem Fokus auf Zeitreihendaten. Insbesondere sollen große Mengen am KIT gemessener EDR-Stromnetz-Messdaten zugreifbar und untersuchbar gemacht werden und Werkzeuge für eine explorative Analyse der Daten entwickelt werden. Die hierfür benötigte Funktionalität umfasst:

- Einen übersichtlichen effizienten und effektiven Zugriff auf die Daten, sowohl auf
  - Datei- und struktureller Ebene, als auch auf
  - inhaltlicher Ebene, v. a. durch Visualisierungsmethoden;
- Eine Möglichkeit der effizienten Datenverarbeitung, um Analysen zu ermöglichen, die auch mit sehr großen Datenmengen gut skalieren.

Hieraus ergeben sich konkrete Softwaremodule, die – in einem Gesamtsystem integriert – den Benutzern alle benötigten Werkzeuge und Hilfsmittel für eine skalierbare Zeitreihenanalyse zur Verfügung stellen sollen, ohne dass spezielle Kenntnisse und Fähigkeiten für die Programmierung datenparalleler verteilter Verarbeitung und Clustersysteme benötigt werden. Die angestrebte Benutzerfreundlichkeit umfasst daher auch die transparente Nutzung der speziellen komplexen Infrastruktur auf einem hohen Abstraktionsniveau.

Die folgenden Softwaremodule werden benötigt:

- Ein Datei-Explorer zur Betrachtung und Manipulation entfernter, verteilter Daten

- Eine Visualisierungsumgebung für die interaktive Exploration und Analyse multivariater Zeitreihendaten
- Ein skalierbares datenparalleles Verarbeitungssystem, welches die folgenden daten- und teilweise auch rechenintensiven Teilschritte der Zeitreihenanalyse auf einem Hadoop-Cluster ermöglicht:
  - Vorverarbeitung
  - Statistische Auswertung
  - Aggregation und Dimensionsreduktion
  - Transformation etc.

## 3.2 Untersuchungen zu Performanz und Skalierung des Hadoop-Clusters der LSDF am KIT

Um die Tauglichkeit und Performanz von Infrastruktur und Softwareumgebung zu testen und festzustellen, in wieweit sie sich für Format und Größe der Zieldaten eignet, wurde zunächst eine verhältnismäßig einfache statistische Auswertung als Beispielanwendung realisiert und evaluiert<sup>17</sup>. Diese wurde auf die von 2012 bis 2015 am KIT verteilt aufgezeichneten EDR-Stromnetz-Messdaten (siehe Abs. 5.1) angewandt. Die EDR-Daten standen hierzu vollständig in HDFS bereit und es wurden täglich die neuesten Datensätze in einem automatisierten Vorgang hinzugefügt. Der Gesamtumfang der Daten betrug Ende 2014 ca. 45 TB an Rohdaten, hinzu kamen Simulations- und Metadaten sowie generierte Testdaten.

Bei einer ersten manuellen Begutachtung der Rohdaten, die bis dahin aufgrund des komplexen Zugriffs nur stichprobenhaft untersucht worden waren, wurden Mess- und Berechnungsfehler sowie fehlende Daten festgestellt, weshalb zunächst bereinigte Testdaten erzeugt wurden. Hierzu wurden fehlerfreie Datensätze ausgewählt und in diverse Ordner kopiert, welche dann EDR-Daten mit unterschiedlicher Messdauer und damit unterschiedlichem Datenvolumen und unterschiedlicher Anzahl an Einzeldateien enthielten. Für die in den Testdaten enthaltenen Zeitreihen sollten die folgenden Kenngrößen berechnet werden:

- Minimum
- Maximum
- Median
- Arithmetisches Mittel
- Standardabweichung.

---

<sup>17</sup> Realisierung und Evaluation der beschriebenen Beispielanwendung wurden teilweise bereits in [18] veröffentlicht und 2013 auf der Konferenz *PDP 2013* in Belfast, Nord-Irland präsentiert (*21st International Conference on Parallel, Distributed and Network-based Processing*).

Um die Performanz der datenparallelen MapReduce Datenverarbeitung mittels Pig mit der eines taskparallelen Multithread Programms vergleichen zu können, wurde die Auswertung für beide Systeme gleichermaßen umgesetzt:

**Das taskparallele Multithread-Programm** wurde in Java implementiert und operierte auf Testdaten auf einer lokalen Festplatte. Deshalb musste eine verhältnismäßig kleine Maximaldatenmenge gewählt werden, so dass die Größenbeschränkungen der Festplatte nicht überschritten wurden. Zwei verschiedene Rechner mit unterschiedlicher Anzahl an Rechenkernen standen als Testsysteme bereit (im Folgenden PC-1 und PC-2 genannt):

- PC-1: Intel i5-2500 CPU, 4 Kerne, 3.3 GHz, 8 GiB RAM, mit Windows 7 64Bit
- PC-2: 2 x Intel Xeon CPU, 8 Kerne, 3.0 GHz, 16 GiB RAM, mit Windows 7 64 Bit.

**Das datenparallele Pig-Programm** wurde ebenfalls als Java-Programm implementiert, welches jedoch ein in Pig Latin eingebettetes Pig-Skript enthielt. Durch das Skript wurden eigene, in Java implementierte Pig-UDFs aufgerufen, welche die spezielle Funktionalität enthielten. Im Testszenarium waren die Testdaten in HDFS gespeichert und der auszuführende Verarbeitungscode wurde von Hadoop und Pig intelligent an Datenknoten gesendet und dort ausgeführt, wo die zu verarbeitenden Daten vorlagen. Das Testsystem war der 58-Knoten Hadoop-Cluster der LSDF (siehe Abs. 3.1.2).

**Pig Skripte und UDFs:** Die Datenverarbeitung mittels Pig sollte direkt aus Java-Programmen heraus gestartet werden können. Deshalb wurde Pig Latin Code in Java-Code eingebettet, was den Vorteil hat, dass Parameter aus der Java-Umgebung zur Laufzeit an Pig übergeben werden können. Hierzu wurde die Apache Pig Java-API verwendet. Sie ermöglicht die Kommunikation mit einem entfernten Pig-Server. Unter Verwendung dieser Schnittstelle wurde eine Java-Methode implementiert, welche zunächst eine \*.jar-Datei mit UDFs am Server registriert, die angegebenen Daten festlegt und anschließend für jeden Eintrag die statistischen Berechnungen in einer UDF durchführt und die Ergebnisse am angegebenen Zielpfad speichert (siehe Code 3.1). Der Quellcode bleibt dabei kurz und leicht verständlich und enthält keine komplexen Anweisungen zur Steuerung der komplexen verteilten Infrastruktur, auf welcher die datenparallele Verarbeitung stattfindet. Für das Laden der Daten wird hier ein Lademechanismus für XML Daten namens `XMLLoader`<sup>18</sup> verwendet, dem als Argument der Name eines XML-Tags übergeben wird. Damit werden ausschließlich alle Tags mit entsprechendem Namen aus allen übergebenen Dateien (z. B. aller innerhalb eines Ordners) geladen. Der Inhalt jedes gefundenen Tags wird dann einer UDF übergeben, welche dort ausgeführt wird, wo die Daten lokal vorliegen.

---

<sup>18</sup> Der erwähnte Lademechanismus stellt selbst eine Pig-UDF dar, welche in einer Sammlung benutzerdefinierter Funktionen namens *PiggyBank* [236] bereits in Pig integriert ist.



```
public void runXMLQuery(PigServer pigServer, String inputPath,
                        String outputPath, String udfPath)
    throws IOException
{
    PigServer.registerJar(udfPath); // register jar with UDFs
    PigServer.registerQuery("A = load '" + inputPath +
        "' using org.apache.Pig.Piggybank.storage.XMLLoader('data') as
        (doc:chararray);");
    PigServer.registerQuery("B = foreach A generate UdfStats(doc);");
    PigServer.store("B", outputPath);
}
```

**Code 3.1:** Java Funktionsbeispiel, um ein Pig-Skript auf dem Hadoop-Cluster auszuführen.

Die UDF namens `UdfStats` aus Code 3.1 erhält als Argument den vollständigen Text innerhalb des Texts. Die EDR-Rohdaten enthalten für jede Sekunde und jeden Kanal ein XML-Tag namens 'data', in welchem die Einzelwerte in einer Base64-Codierung enthalten sind. Der Inhalt des Tags folgt folgendem Schema (siehe Abs. 5.1.2):

Die UDF `UdfStats` (siehe Code 3.3) dekodiert die Base64-codierten Werte (fett in Code 3.2) und führt in einem zweiten Schritt die statistischen Berechnungen (Mittelwert, Maximum, Median, ...) durch und gibt die Ergebnisse an Pig zurück, wo sie weiterverarbeitet oder gespeichert werden können. Hadoop führt im Hintergrund automatisch jegliche Verarbeitung auf den Rechenknoten aus, auf denen die zu verarbeitenden Daten gespeichert sind, um aufwendige Datenbewegungen zu vermeiden.

```
...
<data channel="La" samples="5000" scaleFactor="42.3053" unit="V">
  <![CDATA[TTe3Mvwu ... RgAA==]]>
</data>
...
```

**Code 3.2:** Darstellung des Aufbaus eines „data“-Tags der EDR-Rohdaten anhand eines Beispiels (mit gekürztem Datenblock).

Pig UDFs sind Java-Klassen, welche von Pig bereitgestellte Basisklassen erweitern. UDFs, die eine Funktion auf einzelne Felder eines Datensatzes anwenden sollen, werden beispielsweise als *Evaluationsfunktionen* implementiert (in Code 3.3 *EvalFunc* von engl. *Evaluation function*). Der Typ des Rückgabewertes kann generisch übergeben werden und ist hier ein `String`, also Text.

```
import org.apache.Pig.EvalFunc;
public class UdfStats extends EvalFunc<String> {
    public String exec(Tuple input) throws IOException {
        double[] values = decode(input);
        String result = compStats(values);
        ...
    }
}
```

**Code 3.3:** Aufbau einer Pig-UDF in Java.

Die gleiche Funktionalität wurde in Java als Multithread-Programm für taskparallele Verarbeitung realisiert und auf den zwei Mehrkernrechnern ausgeführt, um die Laufzeit mit der datenparallelen Verarbeitung mittels Pig für unterschiedliche Datengrößen vergleichen zu können. Die Ergebnisse sind in Tabelle 3.1 dargestellt.

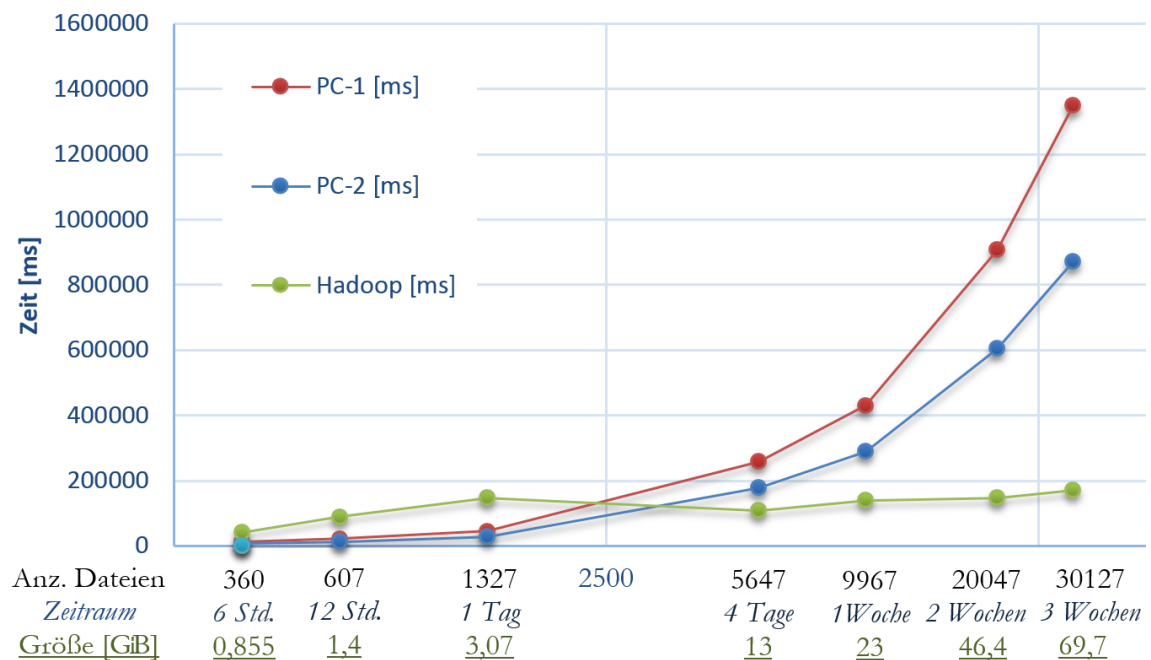
**Tabelle 3.1:** Laufzeitvergleich von Pig (Hadoop) und zwei Mehrkernrechnern.

Mess-Zeitraum	Anzahl der Dateien	Größe [GiB]	PC-1 [ms]	PC-2 [ms]	Hadoop [ms]
6 Std.	360	0,855	13 752	8 706	42000
12 Std.	607	1,400	22 163	13 672	89 000
1 Tag	1 327	3,070	45 488	29 250	147 000
4 Tage	5 647	13,000	258 996	177 827	108 000
1 Woche	9 967	23,000	430 382	289 921	139 000
2 Wochen	20 047	46,400	907 247	604 831	147 000
3 Wochen	30 127	69,700	1 350 670	871 402	171 000

Wie bereits erwähnt, war die maximal verarbeitbare Datenmenge durch die verfügbare lokale Festplattenkapazität von PC-1 und PC-2 beschränkt. Der größte Testdatensatz umfasste deshalb Spannungsmessungen über drei Wochen, was einem Datenvolumen von 69,7 GiB in 30 127 XML-Einzeldateien entspricht. Die Größenbeschränkung stellt dabei an sich bereits ein Ergebnis dar: Sobald die zu analysierende Datenmenge größer als der lokal zur Verfügung stehende Festplattenspeicher ist, kann sie nicht mehr auf einmal verarbeitet werden und muss beispielsweise über ein Netzwerk o. ä. nachgeladen werden. Dadurch wird die Gesamtverarbeitungszeit um die zusätzliche, erhebliche Übertragungsdauer verlängert. Ohnehin müssen Daten aus großen Zeitreihendatenbanken für eine lokale Verarbeitung auf einem PC stets zunächst dorthin übertragen

werden, so dass streng genommen die Übertragungsdauer auf die Gesamtverarbeitungszeit von PC-1 und PC-2 angerechnet werden muss. Im Gegensatz dazu verbleiben die Daten in einer verteilten Umgebung wie Hadoop normalerweise dauerhaft, um dort gemäß des Datenlokalitätsprinzips datenparallel verarbeitet werden zu können. Hadoop ist deshalb in der Lage, über horizontale Skalierung mit weitaus größeren Datenmengen umgehen zu können – übliche Größen sind mehrere hundert TB bis hunderte PB und prinzipiell bei ausreichender Anzahl an Datenknoten auch darüber hinaus.

In Abbildung 3.1 sind die gemittelten Ergebnisse von je drei Testläufen visualisiert. Für die Testdaten kleineren Volumens sind PC-1 und PC-2 schneller mit der Auswertung fertig als die verteilte Pig-Variante auf dem Hadoop-Cluster. Dies kann auf die Initialisierung von Pig und auf die Vorbereitung der Hadoop-Infrastruktur auf die Ausführung zurückgeführt werden, die sich komplexer gestaltet als für die PCs. Mit zunehmender Größe der Testdaten ist jedoch zu beobachten, wie die Laufzeiten für PC-1 und PC-2 mit ähnlicher Charakteristik exponentiell zunehmen, für Hadoop jedoch eher linear<sup>19</sup>. In Abbildung 3.1 ist die horizontale Achse logarithmisch skaliert, um dies besonders anschaulich zu machen. Sie enthält als Beschriftung neben der Datengröße auch den Messzeitraum und die Anzahl an Dateien. Für Testdaten, welche eine Größe von etwa



**Abbildung 3.1:** Vergleich der Laufzeiten mit Pig (Hadoop) und mit zwei Mehrkernrechnern.

<sup>19</sup> Um die Charakteristik des Verlaufs der Kurve für Hadoop besser erkennen zu können, müssen zusätzliche Durchgänge mit größeren Testdaten durchgeführt werden.

6,2 GiB übersteigen, ist die Hadoop-basierte Verarbeitung schneller als auf den PCs. Der Schnittpunkt der Kurven ist abhängig von der Leistungsfähigkeit der PCs und des Clusters sowie vom Verarbeitungsalgorithmus, sollte jedoch grundsätzlich in ähnlicher Weise für alle datenintensiven Berechnungen zu finden sein.

Während die Laufzeit für die Verarbeitung mit den beiden PCs bei mehreren Durchgängen kaum variierte, gab es größere Varianzen bei der Verarbeitung auf dem Hadoop-Cluster. Deshalb wurde die Varianz über mehrere Testläufe auf dem Hadoop-Cluster in einem weiteren Schritt genauer bestimmt. Die Ergebnisse sind in Tabelle 3.2 gelistet. Während der Verarbeitung traten zwei „Hadoop socket-Exceptions (*DataStreamer Exception*)“ für die Durchläufe *H2/1 Tag* und *H4/12h* auf (rot und umrahmt in Tabelle 3.2), welche zu wiederholten Berechnungen und längeren Laufzeiten führten. Ein Zusammenhang zwischen Datenvolumen und Laufzeit ist erkennbar, es treten jedoch starke Varianzen auf, die beispielsweise auch auf gleichzeitig ablaufende Verarbeitungs-Jobs zurückgeführt werden können, da mehrere Nutzer den Cluster gleichzeitig verwenden.

**Tabelle 3.2:** Varianz der Laufzeit in der Hadoop-Umgebung mit Pig über fünf Testläufe mit bedingter Formatierung der Zellohintergründe.

	Messdauer	H1	H2	H3	H4	H5	Median	Mittelw.
Datenvolumen	6 h	0:01:41	0:03:42	0:01:34	0:01:04	0:01:17	0:01:34	0:01:52
	12 h	0:01:26	0:01:07	0:01:17	0:03:17	0:01:20	0:01:20	0:01:41
	1 Tag	0:01:15	0:05:28	0:03:00	0:04:26	0:01:23	0:03:00	0:03:06
	4 Tage	0:04:05	0:01:46	0:01:36	0:05:42	0:02:11	0:02:11	0:03:04
	1 Woche	0:02:01	0:06:36	0:03:49	0:01:50	0:01:48	0:02:01	0:03:13
	3 Wochen	0:05:11	0:05:49	0:03:26	0:06:09	0:06:30	0:05:49	0:05:25

Insgesamt kann aufgrund der Ergebnisse die taskparallele Verarbeitung auf Mehrkernrechnern als ungeeignete und die datenparallele Verarbeitung mit Pig auf einem Hadoop-Cluster durchaus als geeignete Architektur für die Verarbeitung großer Mengen an Zeitreihen-Daten eingeschätzt werden.

### 3.3 Neuer Dateexplorer für Hadoop’s Dateisystem

Um eine komfortable und benutzerfreundliche Interaktion mit dem Hadoop Dateisystem HDFS zu ermöglichen, wurde eine entsprechende Software mit graphischer Benutzeroberfläche in Java entwickelt, die in Abbildung 3.2 dargestellt ist. Das Dateisystem HDFS, in welchem sich die zu

analysierenden Daten befinden, ist ein verteiltes Netzwerk-Dateisystem und unterscheidet sich deshalb in vielen Punkten von den gebräuchlichen Benutzerdateisystemen. Deshalb galt das Hauptaugenmerk der Entwicklung einer einfach zu bedienenden Anwendung, welche eine effiziente Betrachtung und eine schnelle Navigation in den großen Datenmengen mit zahlreichen Dateien und Ordnern ermöglicht und dabei die Komplexität der Verteilung für den Nutzer transparent macht. Auch sollten grundlegende Dateioperationen ermöglicht werden, wie das Erstellen von Ordnern, das Löschen, Verschieben, Umbenennen und das Hoch- bzw. Herunterladen von Dateien zwischen dem lokalen Dateisystem und HDFS.

### 3.3.1 Schnittstelle zu HDFS und Benutzeroberfläche

Die verwendete Hadoop Java-API bietet eine Dateisystem-Schnittstelle namens `org.apache.hadoop.fs` an, welche Funktionalität bereitstellt, die den Zugriff auf Daten in HDFS vereinfacht. Über die API kann auf einem abstrakten Niveau mit HDFS kommuniziert werden, so dass die Komplexität der Infrastruktur, wie z. B. die Speicherung großer Dateien in verteilten, redundanten HDFS-Blockdateien gleicher Größe, nahezu verborgen bleibt.

Abbildung 3.2 zeigt die Benutzeroberfläche des HDFS-Explorers. Über die Dropdownliste im oberen Bereich können HDFS-Pfade eingegeben werden oder es kann aus der Liste bereits genutzter Pfade gewählt werden. In der Abbildung ist aktuell der HDFS-Pfad `/user/globus` geöffnet. Dies ist der EDR-Wurzelordner, in welchem die EDR-Messdaten in einer flachen Ordnerstruktur gespeichert sind. Die Darstellung der Verzeichnisstruktur erfolgt in der Hauptansicht über einen „Dateibaum“, eine Metapher, die für diesen Zweck in vielen Betriebssystemen Verwendung findet, sich jedoch nicht immer optimal skalierbar für große Datenmengen zeigt. Deshalb bietet der HDFS-Explorer erweiterte Ansichten der Verzeichnisstruktur mit unterschiedlichen Schwerpunkten an, die in Abs. 3.3.2 und 3.3.3 beschrieben werden. Das Textfeld im unteren Bereich zeigt die Anzahl enthaltener Dateien und Ordner sowie Datenvolumen, Zugriffsrechte und den Zeitpunkt der letzten Änderung für den aktuell selektierten HDFS-Pfad an. Über die Toolbar im oberen Bereich und über das Kontextmenü können Dateioperationen gestartet werden, welche direkt im entfernten HDFS durchgeführt werden. Außerdem können Einzeldateien, welche Zeitreihendaten enthalten, direkt aus dieser Umgebung heraus visualisiert werden (siehe Abs. 3.6).

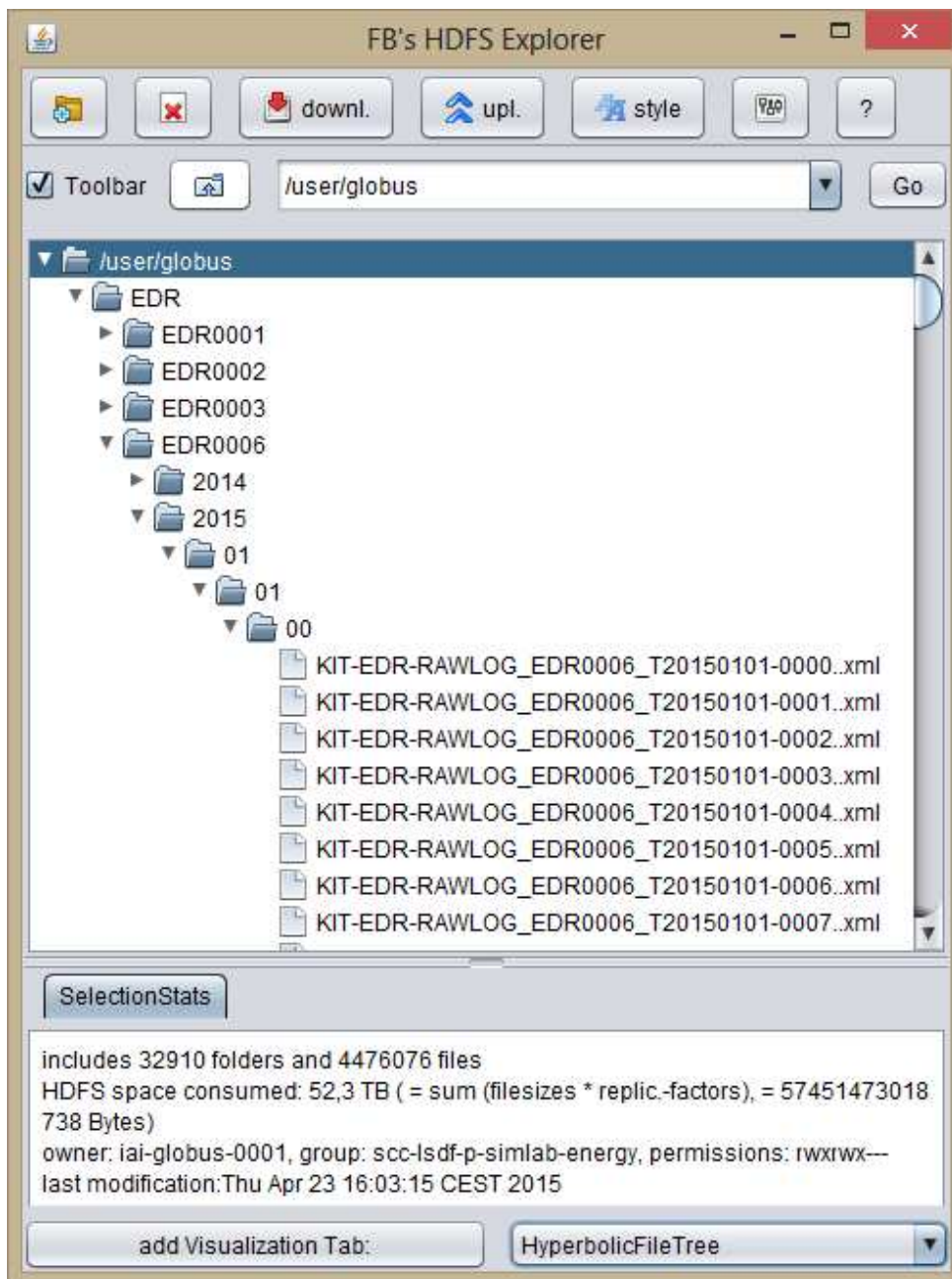


Abbildung 3.2: Oberfläche mit Dateihierarchie des entwickelten HDFS-Explorers.

### 3.3.2 Visualisierung großer komplexer Dateihierarchien

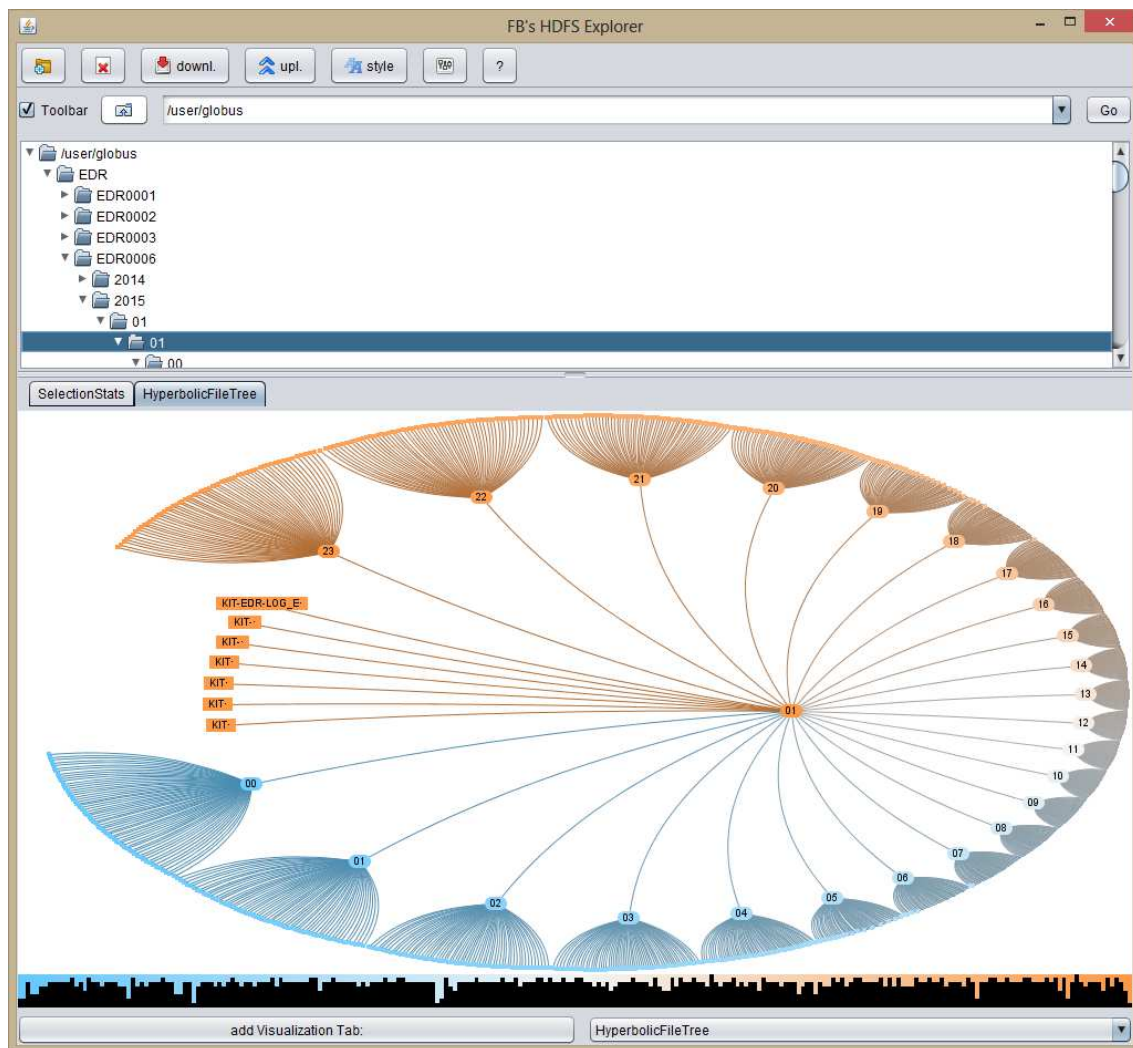
Eine besondere Herausforderung bei der Darstellung der EDR-Daten auf Dateiebene ist die große Anzahl an Dateien und das über die Jahre mehrmals geänderte Format von Ordner- und Dateistrukturen. Die Visualisierung der Dateihierarchie erfolgt als eine Baumdarstellung, wie man sie aus gängigen Dateixplorern unter den Betriebssystemen MS Windows und Apple OSX oder

Linux kennt. Bei der Interaktion mit und der Navigation in den EDR-Daten muss aufgrund der großen Anzahl an Dateien in vielen Ordnern sehr viel vertikaler Bildlauf veranlasst („gescrollt“) werden. Deshalb wurden alternative Darstellungsoptionen entwickelt, welche jeweils eigene Vorteile bei der Visualisierung von Dateihierarchien bieten und über den Button „*add Visualization Tab:*“ im unteren Bereich in Abbildung 3.2. hinzugefügt werden können.

So bietet beispielsweise die Darstellung als *hyperbolischer Baum* (nach [169]) eine Möglichkeit, typischen Darstellungsproblemen zu begegnen, die bei der Baumdarstellung hierarchischer Daten auftreten können. Baumdarstellungen wirken z. B. oft visuell zu voll und überladen und damit unübersichtlich, sobald die Anzahl an Knoten, welche für typische Strukturen mit absteigender Hierarchiestufe exponentiell steigt, und/oder die Tiefe der Hierarchiestufen ansteigt. Derartige Darstellungen benötigen in einem gewöhnlichen Dateibaum so viel Platz, dass der darzustellende Zweig nicht vollständig sondern nur ausschnitthaft auf dem Bildschirm darstellbar ist. Somit kann die Struktur der Hierarchie nicht vollständig vom Betrachter erfasst werden, obwohl hierin häufig semantisch bedeutsame Informationen enthalten ist.

Hyperbolische Bäume nutzen zur Darstellung den hyperbolischen Raum, in welchem sich auf gleicher Fläche mehr Knoten anzeigen lassen als im Euklidischen Raum. Außerdem ist bei einer interaktiven Implementierung, wie der in Abbildung 3.3 gezeigten, eine intuitive Navigation möglich, bei welcher der Benutzer mit der Maus an einer Stelle im hyperbolischen Baum klickt und den Baum nun interaktiv bewegen kann. Dabei verhält sich die Darstellung, als würde der Benutzer eine Kugel drehen, auf welcher Objekte in der Mitte der gewölbten Fläche vergrößert – und an den Rändern verkleinert erscheinen. In der Mitte werden darüber hinaus auch detailliertere Informationen dargestellt als an den Rändern, beispielsweise die vollständigen Ordnernamen als Text. Die hyperbolische Darstellung kann über eine Auswahlliste im unteren Bereich optional als Reiter hinzugefügt werden. Abbildung 3.3 zeigt als Beispiel die hyperbolische Darstellung der Dateihierarchie eines Tagesordners mit EDR-Messdaten des Gerätes „*EDR0006*“.

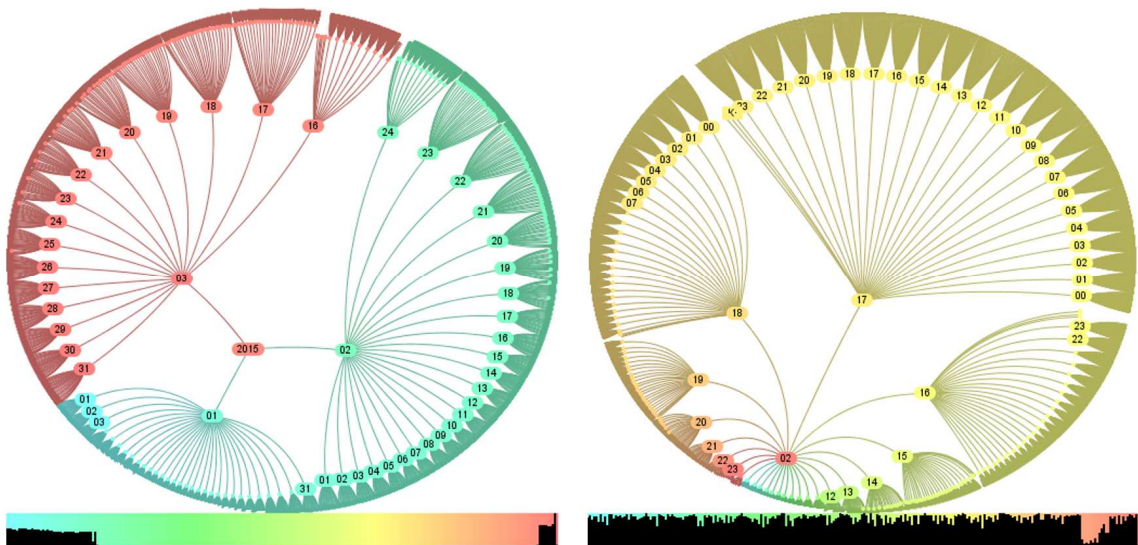
Der Wurzelknoten in der Mitte stellt den Tagesordner dar, umgeben von 24 Stundenordnern („00“ bis „23“), welche jeweils 60 Rohdaten-Dateien für je eine Minute an Messungen im EDR-XML-Format enthalten, und sieben CSV-Dateien, welche die berechneten Feature-Daten für diesen Tag enthalten (für die drei Phasen L1-L3 der Spannung und LI4-LI7 für den Strom aller drei Phasen und des Neutralleiters). Die Knoten repräsentieren dabei Dateien oder Ordner, welche jeweils farblich in einer Weise gekennzeichnet sind, welche das relative Alter der Daten seit deren Erstellung kodiert (blau kodiert ältere –, rot kodiert Dateien neueren Datums). Eine Legende am unteren Bildrand stellt ein Histogramm über die Verteilung der Daten nach Alter dar. Ein Mausklick auf die Legende erlaubt eine erweiterte Parametrisierung der farblichen Codierung, um weitere Dateieigenschaften wie Größe, Besitzer etc. darzustellen.



**Abbildung 3.3:** Darstellung eines EDR-Tagesordners für den 01.01.2015 im HDFS-Explorer mit hyperbolischer Baumdarstellung der Dateihierarchie.

Um die visuelle Skalierbarkeit aufzuzeigen, wird der Visualisierung eines Tagesordners in Abbildung 3.3 nun die Visualisierung eines Jahres- und eines Monatsordners in Abbildung 3.4 gegenübergestellt. Trotz der zusätzlichen Jahres- bzw. Monatebene ist die Dateihierarchie noch gut erkennbar und die hyperbolische Darstellung erlaubt einen schnellen Überblick über den gesamten Messzeitraum. Die gezeigte interaktive Umsetzung des Hyperbolischen Baums basiert teilweise auf Java-Quellcode des Projekts *Treeviz* [294] und wurde für das Auslesen und die Darstellung des HDFS-Dateisystems entsprechend angepasst und in den HDFS-Explorer integriert.





**Abbildung 3.4:** Vorgeschlagene hyperbolische Baumdarstellung der Dateihierarchie im HDFS-Explorer, links: EDR-Jahresordner 2015 mit drei Monaten, rechts: EDR-Monatsordner für Februar 2015.

### 3.3.3 Navigation in langen Dateilisten

Eine weitere Herausforderung für die Darstellung großer Datenmengen stellen lange Aufzählungen wie etwa Dateilisten dar. Da die Auswahl von Elementen aus Listen eine häufig durchzuführende Aktion ist, wird hierfür eine spezielle Darstellungs- und Interaktionsform benötigt, welche in der Lage ist, mit ungewöhnlich langen Listen umzugehen. Als Konzept bietet sich das Fischaugen-Menü [25] an, das – ähnlich der hyperbolischen Baumdarstellung – fokussierte Objekte größer darstellt als solche, die sich weiter entfernt vom aktuellen Fokus befinden.

Die „Fischaugen-Dateiliste“ wurde auf Basis des quelloffenen *Prefuse*-Frameworks [108] entwickelt und kann als Alternative zur klassischen Baumdarstellung der Ordnerstruktur zusätzlich optional ausgewählt werden, ähnlich der zuvor beschriebenen Komponente zur Navigation in komplexen Dateihierarchien. Im unteren Bereich der Oberfläche wird sie bei Bedarf über das Auswahlménü als Reiter hinzugefügt. Das Fischaugenménü stellt dann alle enthaltenen Elemente des aktuell in der Baumdarstellung selektierten Ordners dar und erlaubt ein sehr schnelles visuelles Durchsuchen, indem mit dem Mauscursor die Liste überfahren wird, wodurch Objekte nahe des Cursors vergrößert dargestellt werden (siehe Abbildung 3.5).

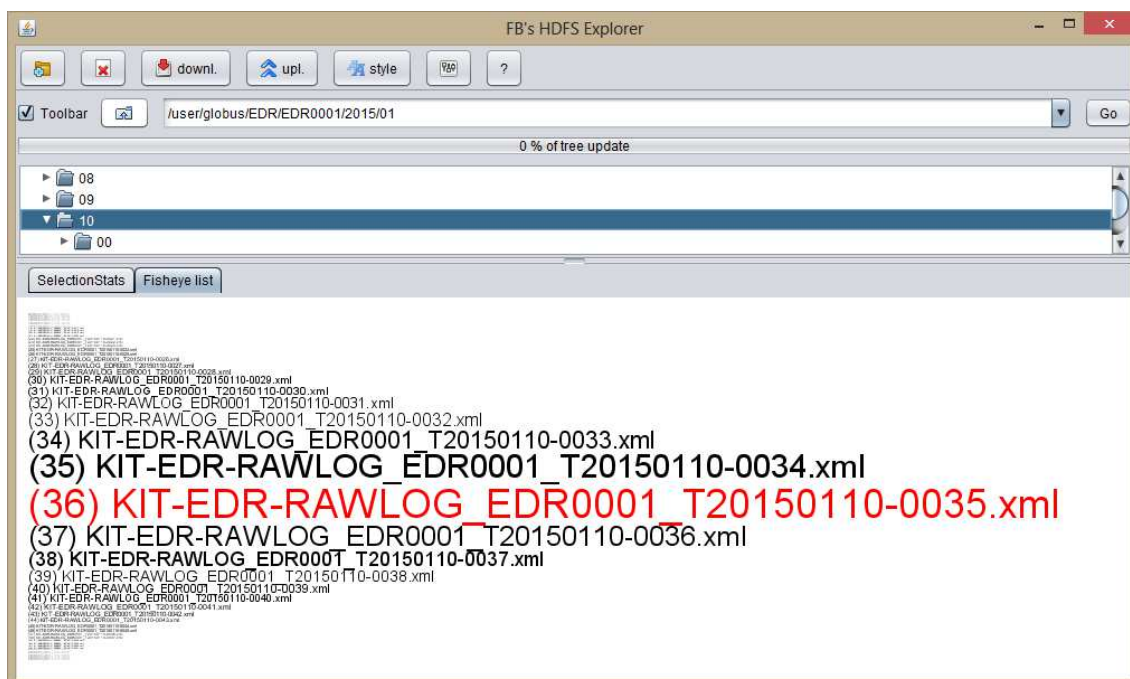


Abbildung 3.5: Interaktive Darstellung langer Dateilisten als Fischaugen-Menü im HDFS-Explorer.

### 3.4 Neue datenparallele Erzeugung von Zeitreihen-Metadaten in Hadoop

Um die Zeitreihendaten weitergehenden Analysen durch die Methoden des Data-Mining und des maschinellen Lernens zugänglich zu machen, werden Metainformationen benötigt, welche die Zeitreihen zusammenfassen und strukturell bzw. semantisch beschreiben. Dazu werden einerseits Statistiken über große Zeitbereiche berechnet (etwa um die Stationarität einer Zeitreihe zu bestätigen, u. ä.) und andererseits eine alternative symbolische Repräsentation der Daten. Beides soll automatisch aus den Rohdaten ermittelt und in Form von Metadaten gespeichert werden, die ein wesentlich kleineres Datenvolumen aufweisen.

Über die Metadaten ist ein semantischer Zugriff auf die Daten möglich und es kann über ermittelte Eigenschaften des Inhalts eine Datenauswahl bzw. Suche realisiert werden. Häufig benötigte Zugriffsmuster bei Zeitreihenanalysen sind beispielsweise die Suche nach Zeitreihen, welche bestimmte Teilsequenzen enthalten, ähnlichkeits- und abstandsbasierte unscharfe Suchen, die Detektion von Schwellwertunter- und -Überschreitungen und von periodischen Mustern etc. [205]. Außerdem ist eine Visualisierung der Metadaten möglich, die eine Exploration des Inhalts ähnlich der Visualisierung der Merkmalsdaten erlaubt. Die Metadaten-Dateien sollen ein möglichst

geringes Datenvolumen aufweisen, so dass lokale Kopien auch für Messdaten über größere Zeitabschnitte auf einem Client gespeichert und lokal analysiert werden können.

Im Folgenden wird ein datenparalleles Hadoop-basiertes Design der Metadatenerzeugung mittels Apache Pig vorgestellt. Dabei werden die entwickelten Pig UDFs erläutert, der Aufbau der XML-basierten Metadateien präsentiert und auf die symbolische SAX-Codierung und die Besonderheiten im Umgang mit der verteilten Infrastruktur eingegangen.

#### 3.4.1 Pig-Funktion für datenparallele Metadaten-Erzeugung

Da die Metadaten im Extremfall für sehr große Zeitbereiche (Tage, Monate, Jahre) und hochfrequente Zeitreihen (z. B. die hochfrequenten EDR-Messdaten) erzeugt werden müssen, wurde für die Auswertung eine datenparallele Verarbeitung mittels Apache Pig entwickelt. Die Erzeugung der Metadaten wird im Folgenden bezogen auf die EDR-Merkmalssdaten im CSV-Format dargestellt, welche selbst bereits abgeleitete Metainformationen für jeweils einen Kanal eines EDR-Gerätes über einen Tag darstellen.

Es wurden eine Verarbeitungskette und eine spezielle Pig UDF entwickelt, welche die Verarbeitung jeweils einer CSV-Datei durchführt. Die EDR-Merkmalssdaten sind Zeitreihendaten in tabellarischer Form – mit Überschriftentext in der ersten Zeile und den Fließkommawerten in den darauffolgenden 86 400 Zeilen (24 Std. \* 60 Min. \* 60 Sek. = 86 400 Sek.). Für jede Sekunde existiert also eine Zeile, welche in der ersten Spalte einen Zeitstempel und in den weiteren Spalten die entsprechenden Werte enthält. Das Format erlaubt kein zeilenweises Verarbeiten der Zeitreihen, da eine Zeile keine Zuordnung zu einer EDR-ID erlaubt<sup>20</sup>, weshalb in jeder UDF-Instanz die Daten einer kompletten<sup>21</sup> CSV-Datei gelesen werden.

Der Ablauf der UDF gestaltet sich so, dass bereits während des Einlesens Informationen zur Erzeugung grundlegender Statistiken wie Minimum, Maximum, Mittelwert, Standardabweichung und Median für jede Zeitreihe erzeugt werden, optional auch für zeitliche Aggregate, also etwa minutliche und stündliche Mittelwerte etc. Sind die Daten einer Datei vollständig gelesen, liegen diese als Arrays von Zeitreihen, Statistiken und Zeitstempeln vor. Für jede Zeitreihe wird schließlich eine dimensionsreduzierte symbolische Repräsentation berechnet (siehe Abschnitt 3.4.3),

---

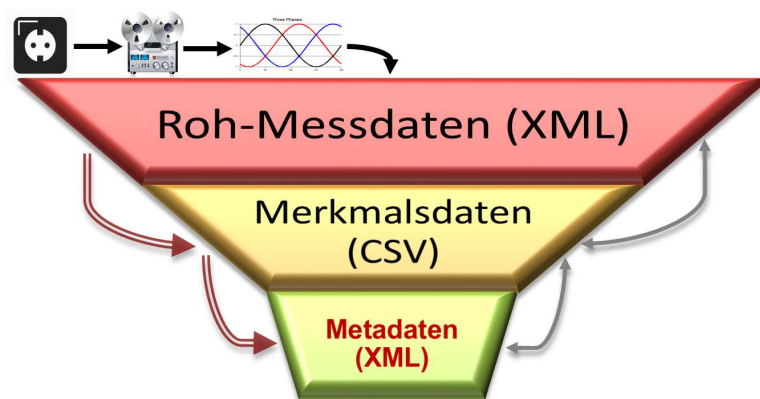
<sup>20</sup> In neueren Versionen von Apache Pig kann aus einer UDF heraus auf den Dateinamen der Datei zugegriffen werden, aus der die aktuell verarbeiteten Daten stammen. Diese Funktionalität konnte unter der während der Entwicklung der UDF verfügbaren Pig-Version (pig-0.8.1-cdh3u5) jedoch leider nicht verwendet werden.

<sup>21</sup> Die HDFS-Blockgröße wurde daher an die Größe der CSV-Dateien angeglichen, um die Effizienz der datenparallelen Verarbeitung nicht durch kleine Einzeldateien zu verringern (vergl. [133], [186], [196]).

welche insbesondere die Anwendung von Data-Mining Methoden zur ähnlichkeitsbasierten Suche ermöglichen, welche für die identifizierten Analyseaufgaben benötigt werden. Die berechneten Informationen werden in einer XML-Datei gespeichert, welche nach den einzelnen Merkmalen strukturiert ist und im folgenden Abschnitt 3.4.2 genauer dargestellt wird.

### 3.4.2 Ableitung und Aufbau der neuen Metadaten-Dateien

Für jeden Kanal eines jeden EDR-Gerätes werden Metainformationen aus den Merkmalsdaten für einen Tag extrahiert und als separate Metadaten-Datei gespeichert. Die erzeugten Metadaten werden als XML-Dateien gespeichert, da dieses Format auf einem verbreiteten Standard basiert und ein flexibles selektives Auslesen einzelner Abschnitte erlaubt. Für jede EDR-CSV-Datei mit Merkmalsdaten wird eine entsprechende Metadaten-Datei erzeugt, die direkt auf die CSV-Datei und hierüber wiederum auf die Rohmessdaten verweist. Die Verweise sind über die Dateinamen möglich, welche einheitlich eine ID des Messgerätes und den Messzeitraum beinhalten (siehe Abbildung 3.6).



**Abbildung 3.6:** Darstellung der Ableitungs- (rote Doppelpfeile links) und Verweiskette (einfache graue Pfeile rechts) bei der Erzeugung von XML-Metadaten aus EDR-Stromnetz-Messdaten (oben).

Abbildung 3.7 zeigt einen beispielhaften Ausschnitt aus einer Metadaten-Datei. Diese beginnt mit einem Informations-Block zur Datei selbst, einer Versionsinformation, der Zeit der Erzeugung, dem abgedeckten Messzeitraum sowie der SAX-Parameter, die zur SAX-Codierung verwendet wurden. Darauf folgt für jedes Merkmal ein Abschnitt mit Statistiken (hier nur über den Gesamtzeitraum der ausgewerteten Datei) und die berechneten SAX-Sequenzen.



- Standardabweichung
- Median.

Die SAX-Sequenzen sind für jedes Merkmal in drei unterschiedlichen Genauigkeitsstufen enthalten (klein, mittel, voll), die sich für unterschiedliche Aufgaben eignen. Dies betrifft die SAX-Alphabetgröße und die PAA-Größe, also den Grad zeitlicher Aggregation (siehe Tabelle 3.3).

**Tabelle 3.3:** Genauigkeitsstufen der SAX-Sequenzen und SAX-Parameter.

<i>Detailgrad</i>	<i>Alphabetgröße</i>	<i>PAA-Größe</i>	<i>Erklärung</i>
<i>klein</i>	4	24	24 Stunden
<i>mittel</i>	4	1440	1440 Minuten
<i>voll</i>	16	86400	86400 Sekunden

### 3.4.3 SAX-Codierung in Hadoop mit Lauflängencodierung

Die Implementierung der SAX-Codierung zur datenparallelen Ausführung auf dem Hadoop-Cluster wurde in eine Pig-UDF integriert. Unter Verwendung von Quellcode aus der quelloffenen *jMotif* [292] Bibliothek wurde die SAX-Codierung zunächst für die lokale Ausführung in Java umgesetzt und erprobt. Für die Ausführung unter Hadoop wurde eine Minimalversion des Codierungscodes erstellt und daraus eine Jar-Datei erzeugt, aus der alle unnötigen Abhängigkeiten und jegliche nicht benötigte Funktionalität entfernt wurden. Das war einerseits wichtig, um die Dateigröße gering zu halten, da die Datei auf alle eingesetzten Datenknoten des Clusters über das Netzwerk verteilt werden muss und andererseits, um von möglichst wenigen zusätzlichen Bibliotheken abhängig zu sein, die ebenfalls verteilt werden müssen und möglicherweise zu Kompatibilitätsproblemen mit der Java-Laufzeitumgebung<sup>22</sup> des Clusters führen können. Außerdem wurde die PAA-Codierung durch eine optimierte Version mit geringerem Speicherbedarf ersetzt, die durch den Hauptentwickler von *jMotif* – Pavel Senin – bereitgestellt und mittlerweile in *jMotif* integriert wurde. Auf die Originalzeitreihen wurde zunächst eine Z-Normalisierung angewandt, anschließend die zeitliche PAA-Aggregation zu Zeitbereichs-Mittelwerten und dann die Diskretisierung des Wertebereichs, wodurch eine SAX-Symbolsequenz erzeugt wurde.

Da die SAX-Repräsentationen vieler Zeitreihen typischerweise viele lange Zeichenketten enthalten, in denen keine Veränderung stattfindet, die also eine Wiederholung desselben Symbols darstellen, wurde eine Lauflängencodierung (kurz RLE von engl. *Run-length encoding*) für SAX-Sequenzen entwickelt, die für eine zusätzliche Datenkompression ohne Verlustbehauptung sorgt.

<sup>22</sup> Auf dem Hadoop-Cluster der LSDF stand zur Zeit der Entwicklung der UDF zur Metadatenerzeugung das verhältnismäßig alte Java 6 (Version 1.6.33) zur Verfügung.

Diese RLE-Codierung (siehe Code 3.4 und das Beispiel in Code 3.5) wird auf jede der SAX-Sequenzen angewandt, das Ergebnis wird jedoch automatisch nur dann verwendet, wenn es tatsächlich weniger Zeichen enthält als vor der RLE-Codierung. Bei der Decodierung kann dann über die Prüfung, ob das erste Zeichen eine Zahl darstellt, effizient ermittelt werden, ob eine codierte oder eine uncodierte Sequenz vorliegt.

```
RunLengthEncoding.encode(  
    SAXFactory_minimal.ts2string(  
        TSUtils.ts2String(  
            TSUtils.optimizedPaa(TSUtils.zNormalize(ts), paaSizeSmall),  
            alphabet,  
            alphabetSizeSmall  
        )  
    )  
);
```

**Code 3.4:** Java-Code des geschachtelten Aufrufs zur neuen SAX-Codierung inkl. RLE-Codierung bei der Metadatenerzeugung auf dem Hadoop-Cluster.

Die Länge der resultierenden Zeichenketten der einzelnen Zeitreihen kann somit stark unterschiedlich sein. In Anlehnung an die *Kolmogorov-Komplexität* und die *Minimale Deskriptionslänge* (MDL) [309] kann die Länge als grobes Maß der strukturellen Komplexität der Zeitreihe dienen, was für spätere Analysen ausgenutzt werden kann.

```
SAX-Beispiel 1: abcdabaaaaabbbbbbbcccccccdcba  
RLE-codiert:   1a1b1c1d1a1b6a7b7c1d1c1b1a  
Resultat:     1a1b1c1d1a1b6a7b7c1d1c1b1a  
  
SAX-Beispiel 2: abcdcbabcdcbabcdcbabcdcbabcdcb  
RLE-codiert:   1a1b1c1d1c1b1a1b1c1d1c1b1a1b1c1d1c1b1a1b1c1d1c1b1a1b1c1d1c1b  
Resultat:     abcdcbabcdcbabcdcbabcdcbabcdcb
```

**Code 3.5:** RLE-Codierung zweier SAX-Beispielsequenzen mit automatischer Auswahl der resultierenden kürzeren Zeichenkette, falls die RLE-Codierung länger ist.

Oftmals werden lange Zeitreihen abschnittsweise in einzelnen Datensätzen bzw. Dateien gespeichert (wie z. B. die EDR-Messdaten). Bei der SAX-Codierung solcher Zeitreihenabschnitte muss berücksichtigt werden, dass Mittelwert und Standardabweichung des vorliegenden Abschnitts die Codierung beeinflussen, was zur Folge hat, dass stark voneinander abweichende Zeitreihenab-

schnitte nicht einheitlich codiert werden. So ist es z. B. nicht zulässig, durch Verkettung der beiden SAX-codierten Sequenzen eine SAX-Sequenz zu erzeugen, die beide Abschnitte repräsentieren soll.

Hier zeigt sich eine wichtige Eigenschaft der SAX-Codierung: Sie ist lokal datenadaptiv. Das ermöglicht zahlreiche interessante Anwendungen für die Untersuchung und den Vergleich lokaler Strukturen, unabhängig vom Wertebereich der gesamten Zeitreihe, etwa durch eine Fensterverschiebungstechnik mit Codierung des Fensterinhalts. Hierbei wird die jeweilige lokale Form der Kurve durch eine SAX-Sequenz repräsentiert, was zum selben Codierungsergebnis führt wie die Codierung eines Fensterinhalts mit gänzlich unterschiedlichem Wertebereich, jedoch gleicher lokaler Kurvenform. Andererseits muss die Datenadaptivität berücksichtigt werden, wenn lange Zeitreihen codiert werden, deren Stationarität unbekannt ist, da es ansonsten zur Codierung von Abschnitten mit stark unterschiedlichem Mittelwert kommen kann. Für manche Anwendungen, wie z. B. die Mustersuche in SAX-Sequenzen, ist dies eher unproblematisch. Es muss jedoch bei der Auswertung und Entwicklung SAX-basierter Analyseanwendungen berücksichtigt werden, dass aus SAX-Sequenzen unterschiedlicher Zeitreihen nicht ohne weiteres auf die Originalzeitreihen geschlossen werden kann, lediglich auf deren innere Struktureigenschaften.

### 3.5 Einheitliche Schnittstelle für Zeitreihendaten

In diesem Abschnitt wird eine Schnittstelle vorgestellt, welche das Einlesen von EDR-Stromnetz-Messdaten und weiteren Sequenzdaten unterschiedlichster Formate in eine einheitliche Datenstruktur ermöglicht. Das hat den Vorteil, dass der Datenzugriff über die Schnittstelle stets auf gleiche Weise erfolgt und für unterschiedliche Softwaremodule wiederverwendet werden kann. Sollen Zeitreihendaten in einem bislang unbekanntem Format mit dem vorgestellten System verarbeitet werden, muss hierzu lediglich die Daten-Schnittstelle für den Zugriff auf die Daten implementiert werden und es sind keine Anpassungen weiterer Software notwendig. Enthalten sind beispielsweise Funktionen zum Lesen und Schreiben der Zeitreihen, Zeitreihenbezeichnungen, Zeitstempel, Darstellungsparameter für die einzelnen Kurven etc. aber auch zur Verknüpfung mit Nutzerkommentaren und der Markierung gefundener Discords bzw. Motifs.

Die Schnittstelle wird als *Java-Interface* entworfen, das die Funktionsköpfe ohne implementierte Funktionsrümpfe enthält und so die zu übergebenden Argumente und Rückgabetypen spezifiziert. Eine Datenklasse, welche die Schnittstelle umsetzt, muss alle Funktionen konkret implementieren. Anschließend können die Daten von allen *FraScaTi*-Softwaremodulen gelesen und verarbeitet werden. Die Schnittstelle wurde bereits implementiert für EDR-Rohdaten (XML-Dateien), EDR-Merkmalen (CSV-Dateien), die extrahierten EDR-Metadaten (XML-Dateien)



und EDR-Mehrtagesdaten (mehrere konsekutive CSV-Dateien als zusammenhängende Sequenz von Merkmalsdaten desselben EDR-Gerätes und derselben Phase). Sie kann ohne großen Aufwand für Sequenzdaten weiterer Formate realisiert werden. Der Datenaustausch in den *FraScaTi* Softwaremodulen findet ausschließlich über diese Schnittstelle statt, wodurch gewährleistet ist, dass jedes neue Datenformat sofort gelesen werden kann und mit allen Modulen zusammenarbeitet.

## 3.6 Neue interaktive Zeitreihenexploration und skalierbare visuelle Analyse

Zu Beginn einer jeden Zeitreihenanalyse steht eine Betrachtung der Daten, um einen Überblick über Charakteristik und zeitlichen Verlauf der Zeitreihen zu gewinnen. Eine Gesamteinschätzung der Charakteristik stellt die Grundlage für die Auswahl geeigneter weiterer Analysen dar. Bereits für verhältnismäßig kleine Zeitreihendaten ist es schwierig für einen menschlichen Betrachter, die Zeitreihen ohne eine kompakte visuelle Gesamtdarstellung zu erfassen. Hierfür gibt es zahlreiche Ansätze, die sich je nach Art der Daten und Analysebedarf unterschiedlich gut eignen.

Die älteste und verbreitetste Form der Zeitreihenvisualisierung ist die zweidimensionale geometrische Kurvendarstellung im kartesischen Koordinatensystem, bei der die kontinuierlich fortlaufende Zeit meist als horizontale und die Messwerte als vertikale Achse fungieren. Die Datenpunkte werden meist verbunden, so dass sich eine geschlossene Kurve ergibt. Durch Hinzufügen effizienter Echtzeit-Interaktionsmöglichkeiten wird es beispielsweise möglich, durch „Zoomen“ schnell zwischen einem Überblick über einen großen Zeitbereich und Detailansichten interessanter Teilabschnitte zu wechseln und eine rasche Navigation in den Daten durchzuführen. Für die Exploration großer Mengen an Zeitreihendaten stellt daher eine interaktive Visualisierung die beste Möglichkeit zur explorativen Untersuchung auf allen Detailstufen dar, die es auch erlaubt, unterschiedliche Visualisierungen zu kombinieren. Hierzu gibt es bereits einige vielversprechende Ansätze und auch gute, etablierte Software (siehe Anhang A.2).

Für die Visualisierung sehr großer Zeitreihendaten ergeben sich jedoch besondere Skalierungsanforderungen, da auch sehr große Zeitreihendaten mit mehreren Millionen an Datenpunkten effizient und effektiv angezeigt werden sollen und die Interaktionsfähigkeit in Echtzeit dabei gewährleistet sein muss. Außerdem ergeben sich anwendungsspezifische Anforderungen und Besonderheiten, die in der Vergangenheit zu zahlreichen Nischenlösungen geführt haben, die nur in bestimmten Bereichen verwendet werden und auf die Durchführung spezieller Aufgaben ausgerichtet sind. Es wurde ein einheitliches, flexibles Visualisierungssystem für beliebige Sequenzdaten benötigt, das eine skalierbare und anwendungsübergreifende Exploration und visuelle Ana-

lyse umfangreicher multivariater Zeitreihendatenbanken erlaubt und dabei außerdem die Komplexität verteilter Infrastrukturen, insbesondere der Segmentierung und Verteilung der Zeitreihendaten für Anwender transparent macht.

Daher wurde für das *FraScaTi*-System ein Softwaremodul zur flexiblen und skalierbaren visuellen Exploration und Analyse von Zeitreihendaten namens *ViAT* (kurz für engl. *Visual Analysis of Time series*) entwickelt, das sich nahtlos in die Gesamtanalyseumgebung integriert.

### 3.6.1 Visualisierungsmodul ViAT

Das Visualisierungsmodul *ViAT*<sup>23</sup> stellt mehrere unterschiedliche Ansichten auf Zeitreihen zur Verfügung, zwischen denen jederzeit gewechselt werden kann (siehe die *Reiter* – im Folgenden wird der in der Softwareentwicklung gebräuchliche engl. Begriff *Tabs* verwendet – in Abbildung 3.8). Dabei hat jede Ansicht ihre eigenen Vor- und Nachteile im Hinblick auf das Sichtbarmachen bestimmter charakteristischer Merkmale der Zeitreihendaten. Zu den wichtigsten Visualisierungen gehören die in Abbildung 3.8 gezeigte Kurvendarstellung, bei der Werte über die vertikale Position kodiert werden, und eine tabellarische Farbfeldvisualisierung (siehe Abbildung 3.18), bei der Werte farblich kodiert werden. Beide folgen dem Zeitstrahlkonzept und stellen den zeitlichen Verlauf der Zeitreihen auf einer linearen Zeitskala dar.

Jede Interaktion in einer der beiden Ansichten wird mit der jeweils anderen synchronisiert, so dass beispielsweise die aktuelle Position innerhalb der Zeitreihen, der sichtbare Ausschnitt, ausgewählte Bereiche usw. stets in beiden Ansichten identisch sind. Dies erlaubt es beispielsweise, für eine Analyse manche Teilaufgaben in der Kurvendarstellung durchzuführen und dann in die tabellarische Ansicht zu wechseln und dort weiterzuarbeiten, sollte die jeweils andere Ansicht hierfür vorteilhaft sein.

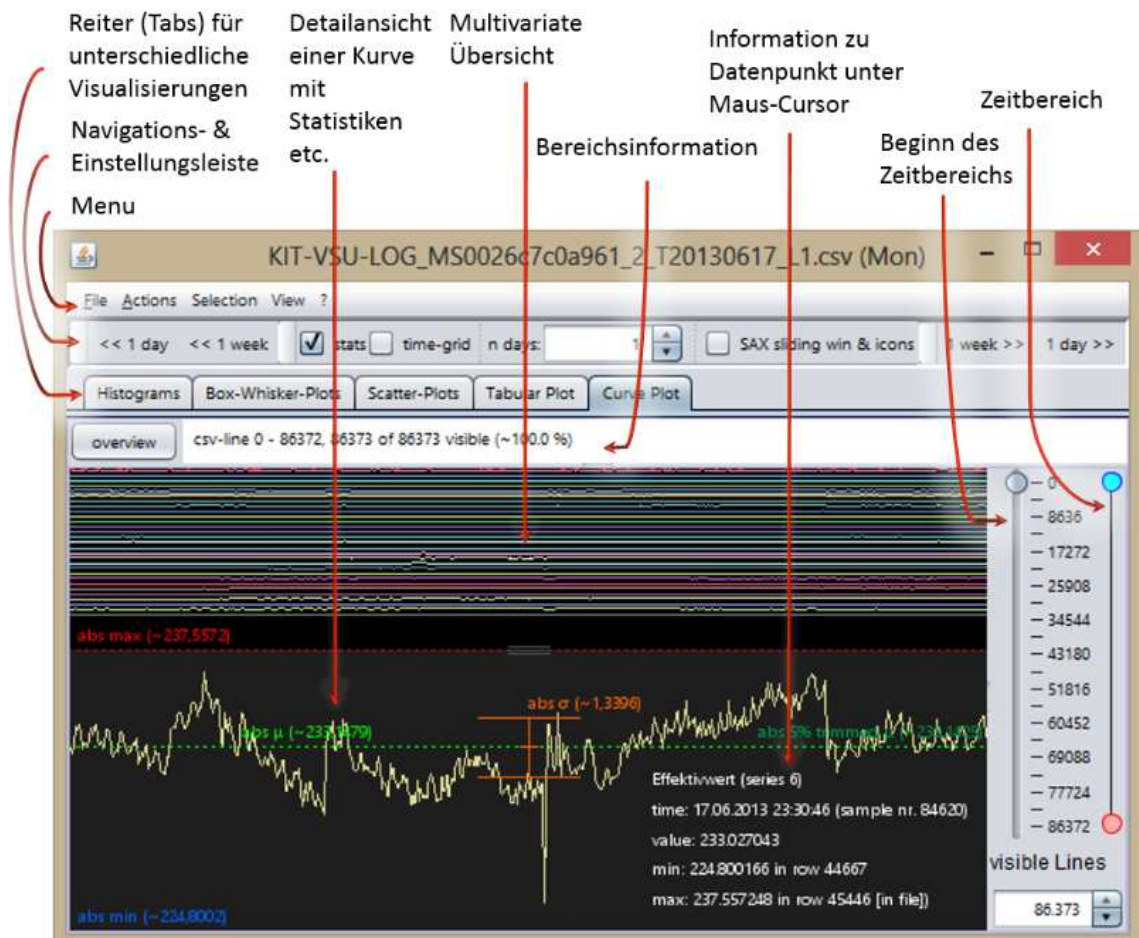
Neben der Kurven- und tabellarischen Visualisierung stehen als weitere Sichten auf die Daten *Reiter* zur Verfügung: Spezielle Zeitreihen-Histogramme, Box-Whisker Plots und Scatter-Plots. Diese Visualisierungen werden per Default nicht initialisiert, um Ressourcen zu sparen und erst bei Bedarf erstellt, so dass die Visualisierung auch bei Darstellung großer Datenmengen schnell startet.

Des Weiteren existieren zahlreiche weitere interaktive visuelle Schnittstellen, welche v. a. in Kombination mit der Anwendung von Data-Mining Methoden genutzt werden können und diese vi-

---

<sup>23</sup> Eine erste Beschreibung der grundlegenden Elemente des Visualisierungsmoduls *ViAT* wurde vom Autor bereits in [194] publiziert.

suell stützen. Sie können auf ausgewählte Bereiche, ganze Zeitreihen oder alle Zeitreihen angewandt werden und können aus dem Menü oder Kontextmenü heraus gestartet werden. Die einzelnen Visualisierungen werden in den folgenden Abschnitten näher behandelt. Das Visualisierungsmodul *ViAT* nutzt die in Abs. 3.5 eingeführte Schnittstelle für Zeitreihendaten. Somit können prinzipiell alle Sequenzdaten eingelesen und dargestellt werden, für die diese Schnittstelle implementiert ist und es können mit allen weiteren Softwaremodulen effizient Daten ausgetauscht werden. Außerdem kann von Drittsoftware über die API effizient, einheitlich und formatunabhängig auf die Zeitreihendaten sowie auf alle verfügbaren Statistiken zugegriffen werden.



**Abbildung 3.8:** Zeitreihenvisualisierung in *ViAT* mit ausgewählter Kurvendarstellung, kommentierter Bildschirmabzug.

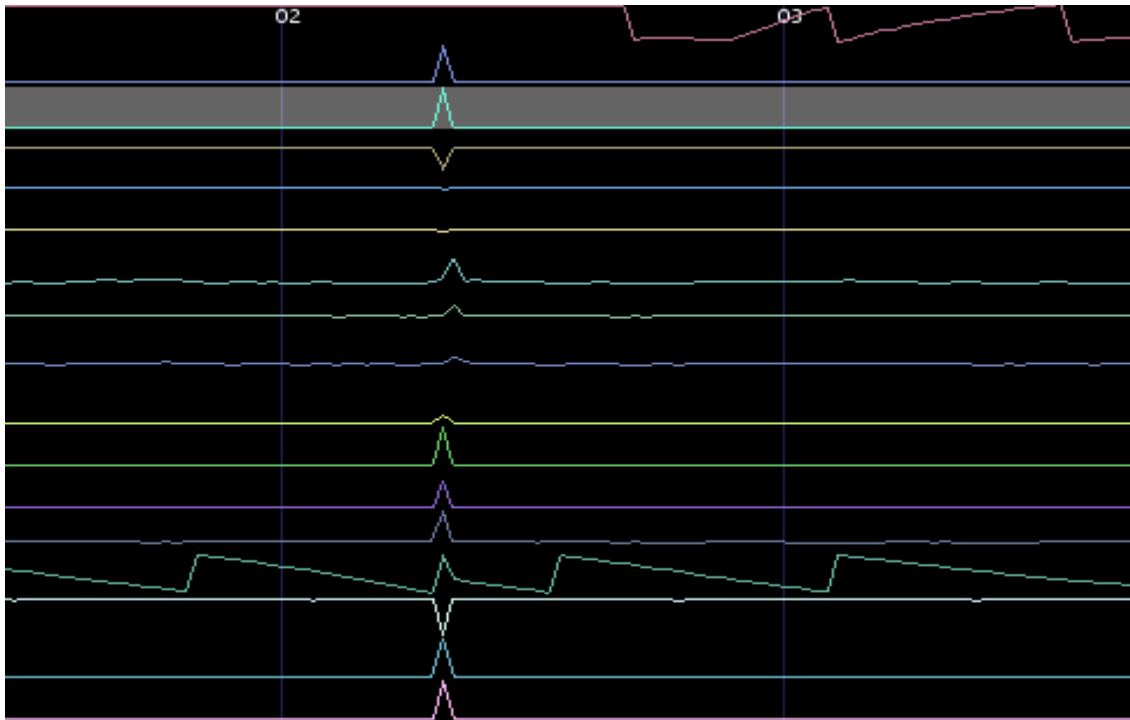
*ViAT* geht über ein bloßes Visualisierungsmodul hinaus, indem es diverse Analysemöglichkeiten des Data-Mining (z.B. Detektion von Discords und Motifs, Ableitung und Darstellung von Funktionen der Originalzeitreihen etc.) direkt integriert und interaktiv mit der Visualisierung kombiniert. Dadurch wird eine intuitive, experimentelle Arbeitsweise ermöglicht, bei der Hypothesen schnell getestet werden können, indem Teilergebnisse aus Analyseschritten sofort visuell rückgemeldet werden.

### 3.6.2 Skalierbare interaktive Kurvendarstellung

Die Kurvendarstellung in *ViAT* ist als Standardvisualisierung bereits beim Start von *ViAT* ausgewählt. Sie bietet eine flexible, kompakte und gleichzeitig leicht verständliche multivariate Darstellung aller in den geladenen Daten enthaltenen Zeitreihen. Hierzu müssen die darzustellenden Zeitreihen einen gemeinsamen Zeitbezug aufweisen, was entweder explizit über gemeinsame oder individuelle Zeitstempel erfolgen kann oder implizit, z. B. beim Laden typischer CSV-Formate wie den EDR-Merkmalen (siehe Abs. 5.1.3), so dass beispielsweise alle Werte in derselben Textzeile unterschiedlichen Zeitreihen und allen Zeilen fortlaufende äquidistante Zeitstempel zugeordnet werden. Sind Zeitstempel vorhanden und wird das Format erkannt, können die Zeitstempel auch als überlagerter Text angezeigt werden und es können zusätzlich Zeitbereiche wie Minuten, Stunden, usw. durch vertikale Begrenzungen dargestellt und durch entsprechende Zeitbereichs-Statistiken ergänzt werden. Außerdem können fehlerhafte Zeitstempel markiert werden.

**Multivariate Ansicht:** Alle Zeitreihen-Kurven werden gestapelt in einer multivariaten Ansicht dargestellt, in der jede Kurve einen gleich großen Raum einnimmt (siehe Abbildung 3.8, Abbildung 3.9 und Abbildung 3.10). Die Reihenfolge der dargestellten Zeitreihen entspricht der Reihenfolge beim Lesen der Daten. Die Stapelansicht macht es möglich, multivariate Zeitreihen unabhängig von ihren konkreten Wertebereichen auf ähnliche Weise darzustellen, so dass etwaige Interkorrelationen im Verlauf sofort sichtbar werden (siehe Abbildung 3.9). Dazu erhält jede Zeitreihe ein eigenes Koordinatensystem mit angepasster Skalierung. Das hat gegenüber der häufig anzutreffenden Darstellung aller Kurven in einem gemeinsamen Koordinatensystem den Vorteil, dass Zeitreihen mit stark unterschiedlichen Wertebereichen (beispielsweise *a*)  $[-230, 230]$  und *b*)  $[-0,01, -0,02]$ ) trotzdem visuell verglichen werden können und sich bei geeigneter Skalierung die Kurven nicht überlappen. Außerdem können die einzelnen Kurven so nicht nur über ihre Farbe o. ä. identifiziert werden, sondern aufgrund ihrer Position. In der Summe führt dies zu einer verbesserten Übersichtlichkeit.

**Detailansicht:** Die multivariate Ansicht kann über den gesamten Darstellungsbereich angezeigt werden oder es kann optional eine einzelne Kurve (z. B. per Doppelklick) fokussiert und in einer erweiterten Detailansicht im unteren Bereich angezeigt werden, wie in Abbildung 3.8 zu sehen ist. Die Höhe der Detailansicht kann durch Ziehen einer entsprechenden Markierung jederzeit beliebig angepasst werden. Die aktuell fokussierte Kurve wird in der multivariaten Übersicht stets grau hinterlegt, um den Kontext zu erhalten und die Position innerhalb der Reihung der anderen Zeitreihen hervorzuheben.

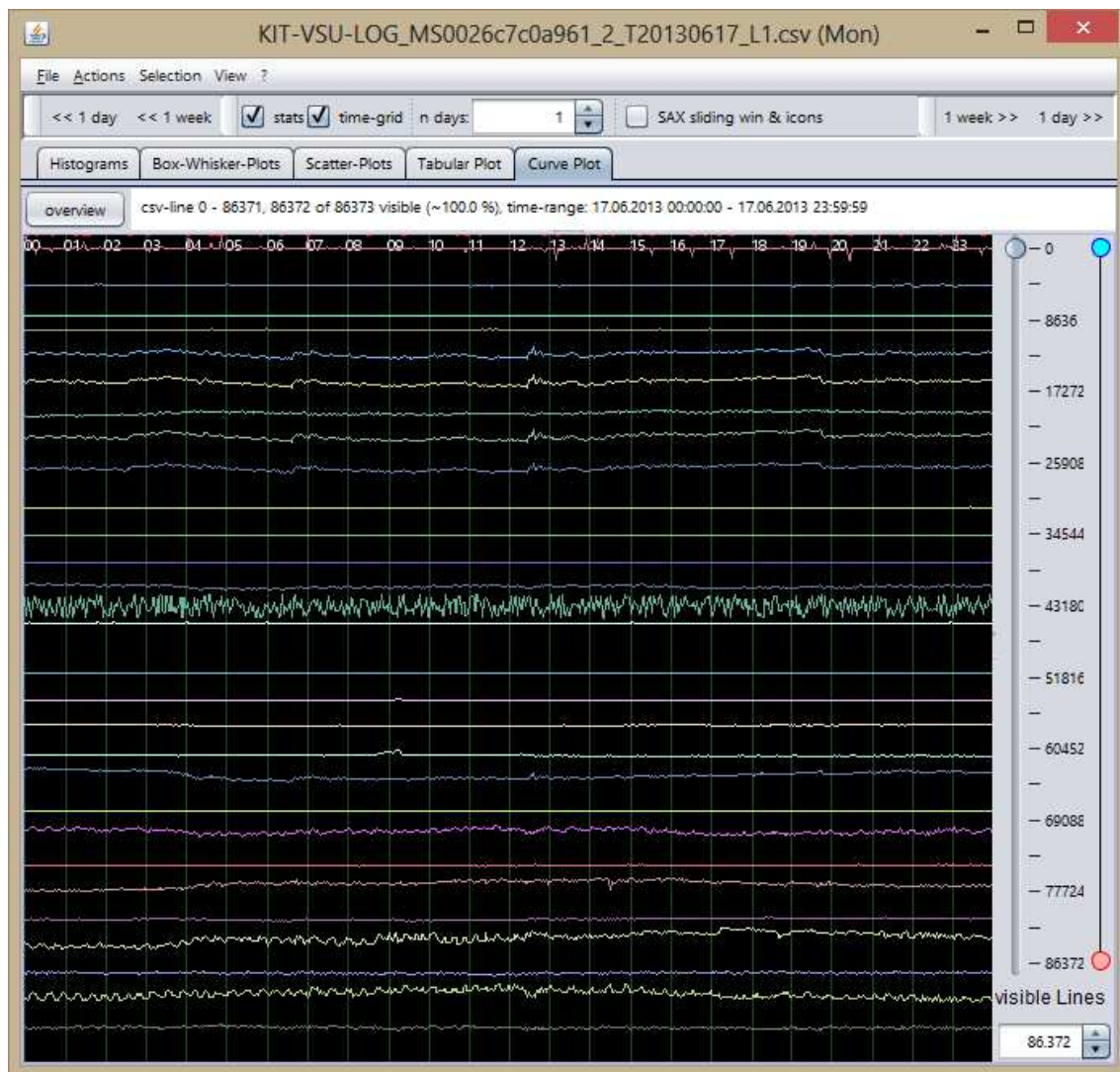


**Abbildung 3.9:** Ausschnitt einer multivariaten Kurvenansicht mit einem Ereignis, das im Verlauf fast aller Kurven sichtbar ist.

**Integrierte Statistiken:** In der Detailansicht werden zusätzlich Statistiken angezeigt, die entweder beim Lesen der Daten berechnet werden oder gegebenenfalls aus bereits bestehenden Metadaten ausgelesen werden können. Dies sind einerseits globale – d. h. über die gesamte Zeitreihe berechnete – Statistiken und andererseits optionale Zeitbereichsstatistiken, falls Zeitstempel zur Verfügung stehen, die „natürliche“ Zeitbereiche wie Sekunden, Minuten und Stunden definieren. So wird unten als gestrichelte blaue Linie das Minimum, mittig das arithmetische Mittel (grün) und ein „getrimmtes“ *arithmetisches Mittel*<sup>24</sup> (dunkelgrün) sowie optional der Median (gelb) und oben das Maximum (rot) angezeigt. Zusätzlich wird um den Mittelwert die Standardabweichung dargestellt (orange).

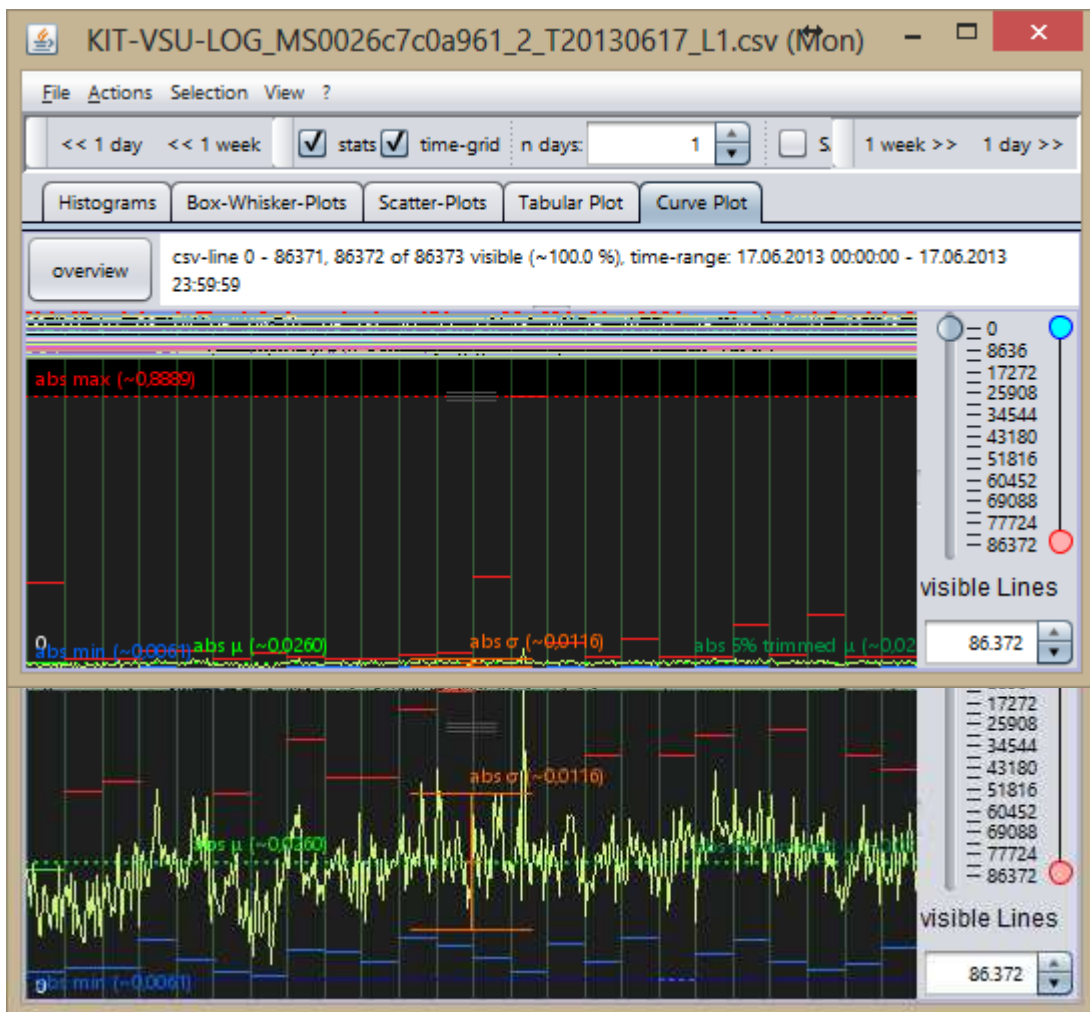
---

<sup>24</sup> Das sog. *getrimmte arithmetische Mittel* begegnet dem Problem des einfachen arithmetischen Mittels, dass Ausreißer den Mittelwert stark verändern können, mit einer Bereinigung der Daten vor der Mittelwertbildung. Hierzu wird ein festzulegender Anteil der größten und kleinsten Messwerte entfernt, so dass diese nicht in den Mittelwert eingehen. Für ein „5 % getrimmtes Mittel“ werden beispielsweise 5 % der höchsten und 5 % der kleinsten Messwerte entfernt. Das getrimmte Mittel liefert in den meisten Fällen sehr ähnliche Ergebnisse wie der Median.



**Abbildung 3.10:** Multivariate Tagesübersicht über EDR-Merkmalen mit gestapelter Kurvendarstellung und aktiviertem Zeitraster.

Die optionalen Zeitbereichsstatistiken werden adaptiv, d. h. passend zum gewählten Zeitausschnitt als durchgezogene Linien visualisiert. Wird beispielsweise die Zeitreihe eines Tages angezeigt, werden lediglich die stündlichen Statistiken angezeigt wie im oberen Teil von Abbildung 3.11. Vergrößert der Benutzer nun den angezeigten Bereich auf wenige Stunden, so werden auch die minutlichen Statistiken für diesen Bereich angezeigt (wie im unteren Teil von Abbildung 3.11) usw. Die Farbgebung entspricht den globalen Statistiken.



**Abbildung 3.11:** Zwei Kurvendarstellungen derselben Zeitreihe: Mit Min-Max Skalierung (oben) ist die Kurvencharakteristik aufgrund zahlreicher Ausreißer kaum erkennbar, jedoch mit  $\mu \pm 2\sigma$  Skalierung (unten).

**Werteskalierung:** Wie bereits erwähnt, können die Messwerte dargestellter Zeitreihen unterschiedlich skaliert werden. Hierzu steht eine *Min-Max Skalierung*, mehrere *mittelwertzentrierte Skalierungen* und *nichtlineare Skalierungen* zur Verfügung. Die Standardskalierung ist eine Skalierung zwischen Minimum und Maximum der Zeitreihe, bei welcher der Wertebereich auf die zur Verfügung stehende Anzeigehöhe abgebildet wird, so dass sich der globale Maximalwert ganz oben und der Minimalwert ganz unten im jeweiligen Anzeigebereich wiederfinden (im Folgenden *Min-Max Skalierung* genannt). Bei Zeitreihendaten ohne signifikante Messfehler oder Ausreißer ist diese Skalierung ideal. Problematisch wird sie allerdings bei Zeitreihen mit extremen Ausreißern. Speziell bei großen Zeitreihendaten muss davon ausgegangen werden, dass solche Ausreißer auftreten. Die Standardskalierung hat den Vorteil, dass alle Zeitreihen eine garantierte, feste Höhe

zugewiesen bekommen und sich nicht überlappen. Sind in einer Zeitreihe jedoch viele Ausreißer fern des hauptsächlichen Wertebereichs enthalten, führen sie zu einer enormen Vergrößerung des Wertebereichs. Eine Kurve wird dann oft innerhalb des Anzeigebereichs so klein dargestellt, dass ihr Verlauf kaum noch erfasst werden kann.

Da insbesondere umfangreiche Zeitreihen leider häufig derartige Ausreißer enthalten, wurde als alternative Skalierungsmöglichkeit eine mittelwertzentrierte „ $\mu \pm \sigma$ “ Skalierung integriert, welche dafür sorgt, dass Kurven um den Mittelwert  $\mu$  zentriert dargestellt werden (siehe Vergleich in Abbildung 3.11). Für die vertikale Skalierung wird die Standardabweichung  $\sigma$  herangezogen, die mit einer wählbaren Konstante  $c$  multipliziert wird.

Es ergibt sich folgende Berechnungsformel:

$$x' = \frac{(x - (\mu - c \times \sigma))}{\left| \frac{((\mu + c \times \sigma) - (\mu - c \times \sigma))}{h_{ZR}} \right|} \quad (3.1)$$

Die Höhe des Anzeigebereichs einer Zeitreihe  $h_{ZR}$  wird dabei genutzt, um einen möglichst großen Teil der Zeitreihe darzustellen.  $x'$  ist der vertikale Versatz in Pixeln, welcher für jeden Datenpunkt mit dem Wert  $x$  zur Anfangsposition des Anzeigebereichs addiert wird. Je nach Charakteristik der Zeitreihe sind unterschiedliche Werte für  $c$  sinnvoll. Durch Ausprobieren findet sich meist sehr schnell eine geeignete Darstellung. Hierzu sind für typische Werte von  $c$  im Kontextmenü entsprechende Einträge für eine sofortige Skalierung enthalten. Wird  $c$  zu klein gewählt, erscheint die Darstellung zu groß und es kann zur Überlappung mehrerer Kurven kommen. Wird  $c$  hingegen zu groß gewählt, erscheint die Kurve kleiner.

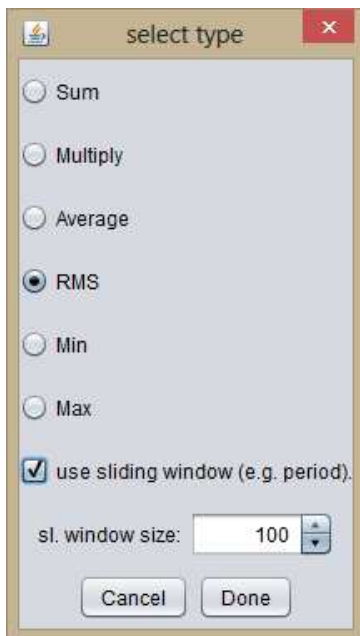
Des Weiteren wurden Möglichkeiten für eine nichtlineare Skalierung in *ViAT* integriert, da eine exponentiell verzerrte Darstellung des Wertebereichs manchmal sinnvoll sein kann, um bestimmte Charakteristika einer Zeitreihe leichter sichtbar zu machen. Hierzu kann ein Exponent festgelegt werden, mit dem jeder Wert der Zeitreihe potenziert wird.



**Abbildung 3.12:** Erzeugung eines neuen Benutzerkommentars zur aktuellen Bereichsauswahl über das Kontextmenü.



**Bereichsselektion:** Durch Ziehen bei gedrückter linker Maustaste kann ein Bereich markiert werden. Der ausgewählte Bereich wird durch eine Umrandung und einen leicht helleren Hintergrund visuell hervorgehoben. Er umfasst sowohl die Selektion eines zeitlichen Bereichs als auch die Auswahl aufeinanderfolgender Zeitreihen. Innerhalb der Umrandung werden Informationen zur Auswahl angezeigt, die es ermöglichen, eine sehr präzise Selektion durchzuführen. Hierzu gehört der Auswahlbereich in Indizes, also z. B. für CSV-Daten Beginn und Ende des Bereichs in Zeilen und Spalten und deren jeweilige Anzahl, falls verfügbar, die Namen der einzelnen ausgewählten Zeitreihen sowie, falls Zeitstempel verfügbar sind, der ausgewählte Zeitbereich. Die Bereichsselektion ist damit ein nützliches Hilfsmittel, um zusätzliche Detailinformationen anzuzeigen (s. Abb. 3.12). Darüber hinaus können Nutzer hierüber bei der Datenanalyse kooperieren, indem Kommentare zu Datenbereichen hinzugefügt und anderen zugänglich gemacht werden.

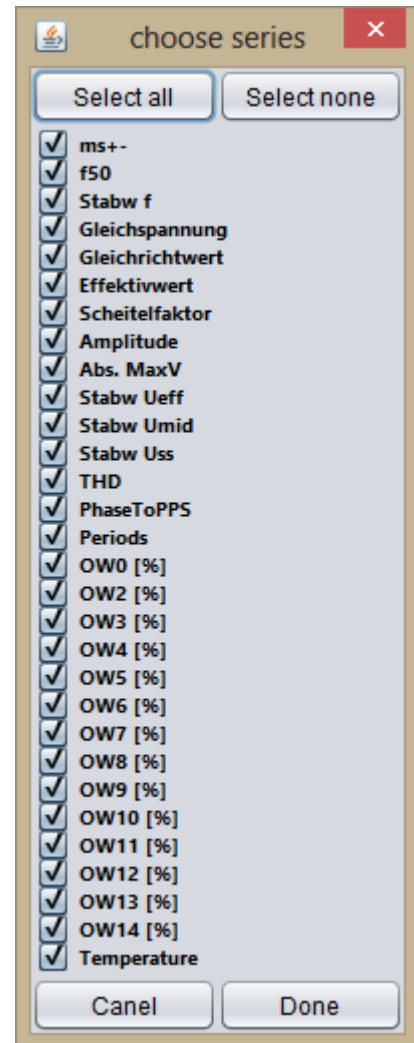


**Abbildung 3.13:** Auswahl der durchzuführenden Transformation bei der Erzeugung einer neuen Zeitreihe.

#### Sichtbarkeit einzelner

**Kurven:** Manchmal kann es sinnvoll sein, nur einen Teilbereich der Kurven darzustellen, etwa wenn die Darstellung sonst zu voll erscheint oder man sich bewusst auf einzelne Kurven konzentrieren will. Hierzu können über einen komfortablen Auswahldialog (siehe Abbildung 3.14), welcher die Zeitreihen namentlich auflistet, sofern Namensinformationen verfügbar sind, einzelne Zeitreihen ein- und ausgeblendet werden.

**Erzeugung zusätzlicher Kurven:** Oftmals müssen Analyseschritte im Kontext der Ursprungsdaten visualisiert werden, beispielsweise, wenn mehrere Zeitreihen summiert, gleitende Mittelwerte gebildet werden etc. Hierzu wurden grundlegende Analysefunktionen (s. Abbildung 3.13) direkt in das



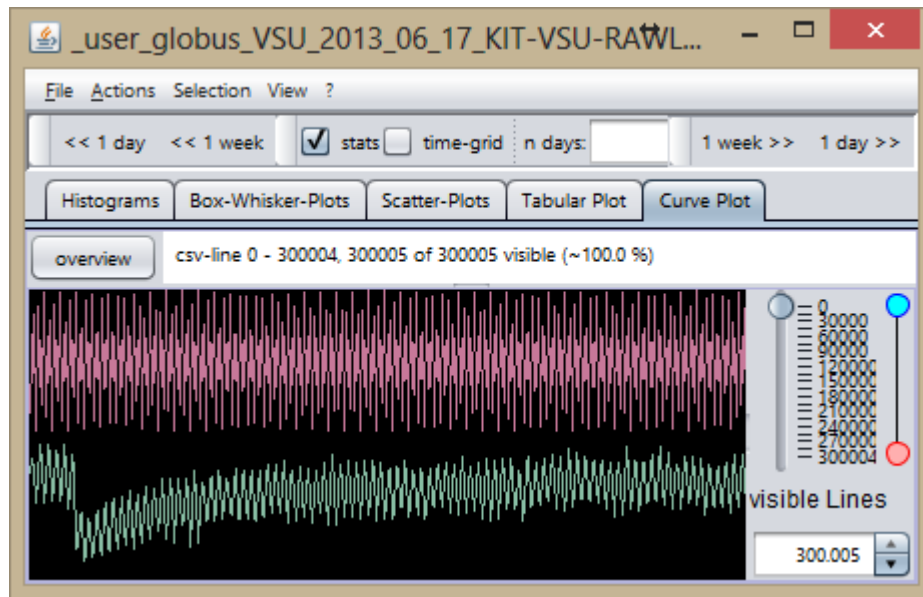
**Abbildung 3.14:** Auswahldialog über sichtbare Kurven.

Visualisierungsmodul *ViAT* integriert, so dass über einen weiteren Auswahldialog Eingabezeitreihen und gewünschte Transformationen gewählt werden können (siehe Abbildung 3.15). Ein Beispiel für eine häufig benötigte Analysefunktion ist die Berechnung gleitender Mittelwerte (siehe Abbildung 3.16). Hierbei können unterschiedliche Fenstergrößen bzw. Zeitbereiche gewählt werden, für die eine Berechnung des gleitenden Mittelwertes zu unterschiedlichen Ergebnissen führen kann, die sofort im Kontext der *ViAT*-Kurvendarstellung überprüft werden können.

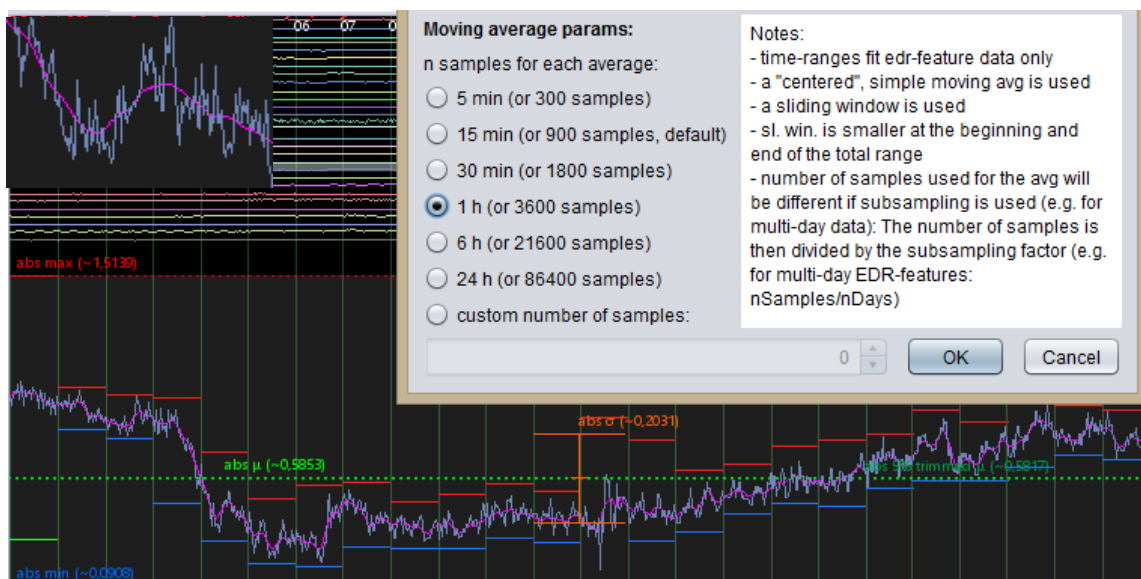
Während es für die Darstellung gleitender Mittelwerte sinnvoll erscheint, sie als Überlagerung der Originalkurve darzustellen, weisen komplexere Transformationen und Kombinationen mehrerer Kurven abgesehen vom gemeinsamen Zeitbezug eine weniger direkte Beziehung zu einer konkreten Kurve auf. Solche Kurven werden deshalb als neue Zeitreihe am Ende des Kurvenstapels eingefügt. Zur Erzeugung neuer Kurven können zunächst Zeitreihen als Eingabedaten ausgewählt werden. Hierzu wird ebenfalls der in Abbildung 3.14 gezeigte Dialog genutzt. Anschließend wird über den in Abbildung 3.13 gezeigten Dialog die Art der Transformation gewählt. Die Liste umfasst momentan Transformationen, die häufig bei der Analyse von Stromnetz-Messdaten benötigt werden, kann jedoch prinzipiell um weitere, für andere Anwendungen benötigte, Transformationen erweitert werden

Eine bei der analytischen Exploration von EDR-Rohmessdaten häufig benötigte Kurve ist z. B. die Wurzel quadrierter Mittelwerte (engl. Root Mean Square, kurz RMS), welche über ein gleitendes Fenster gebildet wird, dessen Größe ein Vielfaches einer Periode umfasst. Hierdurch können beispielsweise Spannungseinbrüche sichtbar gemacht werden, die ansonsten verborgen bleiben. Der RMS wird im Energiekontext meist als Effektivwert bezeichnet. Für sinusförmige Wechselgrößen beträgt er das  $1/\sqrt{2}$ -fache der Amplitude. Er besitzt eine besondere Bedeutung bei der Berechnung der effektiven Spannung und des effektiven Stroms (siehe Abs. 5.1.3). Abbildung 3.15 zeigt eine Darstellung von EDR-Spannungs-Messdaten mit erster Phase (oben) und erzeugter RMS-Kurve (unten), in welcher deutlich ein Spannungseinbruch sichtbar ist, der in der oberen Originalkurve nicht erkannt werden kann. Alle anderen enthaltenen Kurven wurden zugunsten besserer Übersichtlichkeit ausgeblendet.

**Skalierungsspezifische Probleme und Lösungsansätze:** Auch bei der Kurvendarstellung und allgemein bei der Datenvisualisierung sind besondere Anforderungen bezüglich der Skalierbarkeit mit großen Datenmengen zu berücksichtigen. Einerseits betrifft dies das Laden, Verarbeiten und Speichern von Daten. Diesem Aspekt wird jedoch bereits durch die in Abs. 3.5 beschriebene Datenschnittstelle begegnet. Weitere skalierungsspezifische Probleme machen sich bemerkbar, sobald eine große Anzahl von Elementen auf einem Computerbildschirm dargestellt werden soll.



**Abbildung 3.15:** Darstellung der ersten Phase der Spannungsmessung (obere Kurve) und des erzeugten RMS (untere Kurve), in welchem sich deutlich ein Spannungseinbruch zeigt, der in der oberen Originalkurve nicht zu sehen ist.



**Abbildung 3.16:** Berechnung und Darstellung eines gleitenden Mittelwertes, mit Dialog zur Auswahl des Summationsbereichs (rechts oben), Kurve mit überlagertem gleitenden Mittelwert (pink, unten) und Detaildarstellung der beiden Kurven (links oben).

Bei einer multivariaten Kurvendarstellung müssen hierbei zwei Dimensionen berücksichtigt werden: Die Anzahl anzuzeigender Kurven und die Anzahl an Datenpunkten in jeder der Zeitreihen.

Beispielsweise umfasst eine typische EDR-Merkmalstabelle für die Spannung einer Phase über einen Messtag ca. 30 Merkmalszeitreihen à 86 400 Werte (die Anzahl an Sekunden eines Tages). Es müssen also insgesamt 2 592 000 Einzelwerte für einen Messtag angezeigt werden. Zur Visualisierung großer Datenmengen werden ohnehin meist große, hochauflösende Displays eingesetzt. Das Display, das während der Entwicklung des Visualisierungsmoduls *ViAT* verwendet wurde, hat eine Auflösung von  $1920 \times 1080$ , also 2 073 600 Bildpunkten. Selbst wenn jeder einzelne Pixel zur Darstellung der Zeitreihen genutzt wird, können nicht alle Elemente angezeigt werden und ein Informationsverlust ist unumgänglich. Das bringt einige Probleme mit sich:

**Visuelles Aliasing:** Durch „Unterabtastung“, hier also durch Auslassung einzelner Messwerte bei der Darstellung, kann es zu unerwünschten Artefakten kommen. Sobald die Abtastrate unterhalb der Nyquistfrequenz der darzustellenden Kurve liegt, wird sie nicht mehr korrekt dargestellt. Gerade bei sinusähnlichen Kurven kann es dadurch zur Darstellung neuer, verzerrter Sinuskurven kommen, die nicht länger der Charakteristik der Zeitreihe entsprechen.

**Detailgrad:** Bei der Auslassung zu vieler Einzelwerte geht schlichtweg Information bei der Kurvendarstellung verloren. Beispielsweise werden möglicherweise bedeutsame lokale Extremwerte nicht als Kurvenabschnitt dargestellt, was etwa den falschen Eindruck vermittelt, die Kurve verlaufe stetig weiter.

**Geschwindigkeit:** Die Geschwindigkeit von Darstellung und Interaktion wird möglicherweise stark sinken, sobald eine kritische Anzahl an Elementen dargestellt werden muss, welche die verfügbaren Ressourcen des Computers voll ausschöpft. Gerade die Interaktivität der Darstellung ist jedoch grundlegend für eine erfolgreiche explorative Analyse. Vor allem muss es stets möglich sein, flüssig in der Zeitreihendarstellung zu navigieren und den dargestellten Zeitbereich zu verändern (zeitlich zu „zoomen“).

Den genannten Problemen wurde in *ViAT* folgendermaßen begegnet. Zunächst wurde ein Mechanismus integriert, welcher es ermöglicht, bereits beim Laden der Daten einzelne Werte auszulassen, was solange die Charakteristik der Kurven erhält, solange die Nyquistfrequenz dabei nicht unterschritten wird. Hierzu kann entweder eine Schrittweite festgelegt werden oder ein Maximalwert für die Anzahl an Zeitschritten. Für dieses Vorgehen muss die Nyquistfrequenz der Daten jedoch bereits bekannt sein, so dass es nur für bekannte Daten eingesetzt werden sollte und nur, falls die Anzahl darzustellender Elemente ansonsten inakzeptabel ist. Eine Alternative ist die Aggregation der Daten, z. B. zu Mittelwerten. Das hat jedoch den Nachteil, dass die Kurven dadurch geglättet werden und somit die Kurvencharakteristik verändert werden kann.

Dem Problem des visuellen Aliasing und des Fehlens von Information bei der Überblicksdarstellung zahlreicher Elemente wird mit hierarchischen Zeitbereichstatistiken begegnet, deren zusätz-

liche Einblendung eine Einschätzung ermöglicht, in welchem Wertebereich sich die Kurve innerhalb eines Zeitbereichs aufgehalten hat, ohne dass die aufgrund der Auslassungen fehlerhaft dargestellte Kurve es verrät. Beispielsweise ist ein verhältnismäßig großer Bereich zwischen Minimum und Maximum ein Hinweis auf viel Dynamik und einen komplexen, ungewöhnlichen Gesamtverlauf des Kurvenabschnitts. Durch Zoomen kann der Ausschnitt dann sofort näher untersucht werden.

Die mit Zunahme der anzuzeigenden Datenmenge abnehmende Geschwindigkeit und Flüssigkeit von Darstellung und Interaktion ist ein allgemeines Problem, das nicht vollständig gelöst werden kann, da es stets eine kritische Datenmenge geben wird, deren Überschreitung inakzeptable Reaktionszeiten eines Visualisierungssystem auslösen wird. Eine interaktive Visualisierung, die speziell für die Exploration großer Daten ausgelegt ist, sollte diese kritische Datenmenge jedoch voll ausschöpfen. Das kann durch eine performante und ressourcensparende Implementierung und durch Datenreduktion erreicht werden.

Statt auf bestehende, bewährte Visualisierungs-Frameworks aufzusetzen, wurden deshalb die Kurvendarstellung und auch die tabellarische Farbflächendarstellung in *ViAT* von Grund auf neu entwickelt. Verbreitete und mächtige Frameworks zur interaktiven Zeitreihenvisualisierung wie *jFreeChart* [131] zeigten sich bei Tests als ungeeignet zur Anzeige großer multivariater Zeitreihendaten. Beispielsweise reagierte während der Darstellung einer einzigen Minute an EDR-Messdaten (sieben Zeitreihen mit je ca. 300 000 Messwerten) in einem Test die Anzeige beim Zoomen ca. fünfzehn Sekunden lang nicht mehr. Eine Recherche nach Optimierungsmöglichkeiten erbrachte ein geringes Verbesserungspotential<sup>25</sup>, das ausgeschöpft jedoch noch immer zu unakzeptablen Reaktionszeiten führte. Dies ist allgemein auf den objektorientierten Programmier-Stil solcher Frameworks zurückzuführen, da auf diese Weise Grundelemente der Visualisierung wie Datenpunkte, Kurven, Achsen etc. als Klassen realisiert sind, von denen sehr viele Objekte instanziiert werden müssen. Die Kurven- und tabellarische Farbdarstellung wurden daher auf Grundlage von Grafik-Bibliotheken des *Processing*<sup>26</sup>-Frameworks [242] neu entwickelt, um durch Aussparung nicht benötigter Funktionalität und ressourcenschonende Abläufe möglichst große Zeitreihendaten interaktiv visualisieren zu können. So werden in *ViAT* Datenstrukturen stets so

---

<sup>25</sup> In den *jFreeChart*-Foren (unter <http://www.jfree.org/phpBB2>) fanden sich Vorschläge von Benutzern und Entwicklern, wie man das Rendern der Diagramme beschleunigen kann. Hieraus wurden für die verwendeten Diagramme eigene optimierte Renderer implementiert, welche die Darstellung teilweise performanter machten. Für mehrere Millionen Datenpunkte waren die Reaktionszeiten jedoch immer noch zu lange (teilweise über 20 Sekunden).

<sup>26</sup> *Processing* ist eine auf Java basierende Programmiersprache und Entwicklungsumgebung, welche speziell für die Erstellung interaktiver Darstellungen entwickelt wurde. Für *ViAT* wurde jedoch lediglich eine Grafik-Bibliothek aus *Processing* verwendet, die Software wurde komplett in Java entwickelt. Die Java-Syntax unterscheidet sich in einigen Punkten leicht von der vereinfachten *Processing*-Syntax, weshalb z. B. zum Testen von *Processing*-Code aus Tutorials einige Modifikationen notwendig waren.

durchlaufen, dass nur tatsächlich in der Visualisierung sichtbare Elemente besucht werden. Dadurch sinkt die Last bei einer Überblicksansicht, bei der ein großer Datenbereich auf kleinem Raum dargestellt werden soll, da die meisten Datenpunkte aufgrund der beschränkten Auflösung nicht dargestellt werden können und somit auch nicht durchlaufen werden müssen. In entsprechenden Schleifen wird deshalb eine Schrittweite verwendet, die dem Zoomlevel entspricht. Andererseits sinkt die Last auch bei einem hohen Zoomlevel, da Datenpunkte außerhalb des Darstellungsbereichs nicht dargestellt werden.

Des Weiteren sind die Interaktionsmöglichkeiten und die Möglichkeiten der Echtzeit-Synchronisation zwischen verschiedenen Darstellungen in Frameworks wie *jFreeChart* u. a. mehr als unbefriedigend. Das betrifft vor allem Navigation und Zoom in den Daten, da diese Aktionen bei der Exploration großer Zeitreihen sehr häufig durchgeführt werden müssen. Daher wurden effektivere Möglichkeiten für Zoom und Navigation entwickelt und in *ViAT* integriert, die in der Kurvendarstellung sowie in der tabellarischen Farbdarstellung gleichermaßen verwendet werden können. Beide Visualisierungen werden dabei in Echtzeit synchronisiert, so dass Position, Zoom, Bereichsselektion etc. für die jeweils andere Ansicht übernommen werden.

**Zoom:** Visualisierungsframeworks wie *jFreeChart* bieten sehr eingeschränkte Möglichkeiten zur Vergrößerung einzelner Diagrammabschnitte und lassen sich auch nicht ohne weiteres um die benötigte Funktionalität erweitern. Eine Erhöhung des Zoomlevels in einer *jFreeChart*-Kurvendarstellung funktioniert so, dass der Nutzer mit der Maus einen rechteckigen Bereich markiert, welcher dann über die volle Darstellungsfläche präsentiert wird. Dabei muss das Auswahlrechteck mit gedrückter Maustaste von oben nach unten gezogen werden. Ein Herauszoomen ist umgekehrt möglich, indem also ein Rechteck von unten nach oben gezogen wird, wodurch der dargestellte Bereich auf den vollständige Zeit- und Wertebereich festgelegt wird. Diese Form der Veränderung des Detailgrades ist unflexibel und ineffizient, da sie keine Navigation bei Beibehaltung des Zoomlevels und keinen graduellen Zoom ermöglicht. Außerdem kann das Rechteck während der Interaktion nicht korrigiert werden, so dass eine geringfügige Erweiterung des dargestellten Ausschnitts ein vollständiges Herauszoomen, das erneute Finden der zu betrachtenden Stelle und ein erneutes Hereinzoomen erfordert. Außerdem wird das Bild der Kurven vergrößert, also gleichzeitig im Zeitbereich und im Wertebereich gezoomt, obwohl beides häufig unabhängig voneinander benötigt wird, um eine sinnvolle Exploration zu ermöglichen.

Für *ViAT* wurde deshalb eine Vergrößerung entwickelt, die sich nur auf den Zeitbereich bezieht, da der Wertebereich in der Detailansicht ohnehin in der Darstellungshöhe angepasst werden kann. Die Vergrößerung wurde so realisiert, dass der Anzeigebereich durch Drehen am Mousrad

bei gehaltener `Strg`-Taste<sup>27</sup> um die Position des Mauszeigers zentriert vergrößert und verkleinert, so dass genau an die Stelle gezoomt werden kann, auf welche gerade mit der Maus gezeigt wird, ohne dass die aktuelle Position verlorengeht. Dieser Ablauf bietet den Vorteil, dass Vergrößerung und Verkleinerung auf gleiche Weise funktionieren und während des Zoomens Korrekturen möglich sind. Beispielsweise kann durch Bewegung des Zeigers während des Zoomens der Ausschnitt entsprechend verschoben, also gleichzeitig gezoomt und navigiert werden. Außerdem kann die Zoomgeschwindigkeit erhöht werden, indem zusätzlich die `Shift`-Taste gehalten wird, so dass je nach Bedarf gleichermaßen grob und fein gearbeitet werden kann. Die Vergrößerung ist außerdem datenadaptiv, also abhängig von der Anzahl dargestellter Datenpunkte, so dass in einer Überblicksdarstellung über Millionen von Datenpunkten schneller gezoomt wird als in einer vergrößerten Detailansicht. Ist ein Bereich selektiert, so kann außerdem der selektierte Zeitbereich über das Kontextmenü als neuer Darstellungsbereich ausgewählt werden. Durch Drücken der Taste `O` (kurz für engl. *Overview*) wird der Darstellungsbereich auf den vollständigen Zeitbereich der geladenen Daten erweitert.

**Navigation:** Die Navigation kann analog zur Vergrößerung ebenfalls mit dem Mausrad (ohne Halten der `Strg`-Taste) durchgeführt werden oder alternativ mit den Pfeiltasten links/rechts. Das zusätzliche Halten der `Shift`-Taste beschleunigt die Scrollgeschwindigkeit. Durch Drücken der Taste `S` (kurz für *Start*) wird unter Beibehaltung der Größe des Darstellungsbereichs sofort zur Anfangsposition verschoben.

### 3.6.3 Kooperation durch Benutzerkommentare

In der Kurvendarstellung und der tabellarischen Farbdarstellung ist eine Selektion von Bereichen möglich, die über das Kontextmenü mit Nutzerkommentaren versehen werden können. Die Nutzerkommentare sind natürlich rollen- und anwendungsabhängig, weshalb ein generisches System entwickelt wurde, das unterschiedliche Anwendungsfälle abdecken kann. Die Nutzerkommentare stellen selbst weitere Metadaten dar und werden in einem XML-Format (siehe Code 3.6) in einer Datenbank gespeichert. Enthalten sind für jeden Kommentar die kommentierten Daten, der Verzeichnispfad, Beginn und Ende des kommentierten Bereichs (Uhrzeiten und Position der Messwerte), Ersteller und Erstellungszeitpunkt, Angaben über die letzte Änderung (wer, wann), Angaben zum Messort und zum verwendeten Messgerät sowie zu den kommentierten CSV-Spalten.

---

<sup>27</sup> Das Zoomen mit `Strg`-Taste und Mausrad findet sich in zahlreichen bekannten Applikationen, weshalb diese Methode von Nutzern nicht neu erlernt werden muss. Auch das „zentrierte Zoomen“ wird in mancher Software auf ähnliche Weise verwendet, wie z. B. in fortschrittlicher Bildbearbeitungssoftware.

```

<?xml version="1.0" ?>
<data.db.xmlDB.timeRangeCommentsAndEvents.TR_EDR_FeatureCsv_Event>
  <describedFileNames>
    KIT-VSU-LOG_MS0026c7c0a961_2_T20130617_L1.csv
  </describedFileNames>
  <describedFileDirPath>
    E:/_down_test/_DATA_/original data/20130617_SpannungseinbrKitN14Uhr
  </describedFileDirPath>
  <startTime>
    2013-07-17 05:16:48.0 UTC
  </startTime><timeRangeMillis>
    21547000
  </timeRangeMillis>
  <createdBy>
    jc3561
  </createdBy>
  <lastChangedBy>
    jc3561
  </lastChangedBy>
  <timeOfCommentCreation>
    2015-05-12 11:21:20.820 UTC
  </timeOfCommentCreation>
  <timeOfLastCommentChange>
    2015-05-12 11:25:18.828 UTC
  </timeOfLastCommentChange>
  <header></header>
  <comment>Ungewöhnliche Oberschwingungen</comment>
  <type>eventComment</type>
  <edrUnitName>MS0026c7c0a961</edrUnitName>
  <channel>21</channel>
  <gpsLatitude>49,096</gpsLatitude>
  <gpsLongitude>8,433</gpsLongitude>
  <gpsAltitude>145,2</gpsAltitude>
  <csvStartCol>21</csvStartCol>
  <csvEndCol>29</csvEndCol>
  <csvStartRow>26193</csvStartRow>
  <csvEndRow>47740</csvEndRow>
  <csvCols>
    <string>a</string>
    <string>b</string>
    <string>c</string>
  </csvCols>
</data.db.xmlDB.timeRangeCommentsAndEvents.TR_EDR_FeatureCsv_Event>

```

**Code 3.6:** Resultierende XML-Metadaten aus einem Bereichskommentar in EDR-Merkmalen, mit letzter Änderung, Autor, Bereich, Kommentar etc.



Je nach Rolle des Nutzers und Art geladener Daten wird automatisch ein bestimmter Kommentartyp genutzt. Neue Kommentartypen können leicht durch Ableitung vorhandener konkreter oder abstrakter Kommentartypen als Java-Klasse hinzugefügt werden.

Es wurde ein System entwickelt, das aus der Datenstruktur generische Eingabedialoge zu erzeugen vermag, so dass auch für zukünftig integrierte Datenstrukturen Eingabedialoge und entsprechende Kommentardaten generiert werden können. Für jeden Kommentartyp werden zur Dateneingabe spezielle anwendungsabhängige Eingabemasken generisch aus der jeweiligen Java-Klasse erzeugt<sup>28</sup>, welche typabhängige Felder für alle Daten des Kommentartyps enthalten. So können z. B. ganzzahlige Werte über einen `JSpinner` eingegeben werden, Text über Textfelder usw.

Abbildung 3.17 zeigt ein Beispiel für eine Eingabemaske, über die Nutzer Kommentare zu bedeutsamen Ereignissen in EDR-Merkmalen festhalten können. Felder, deren Inhalt sich bereits aus Bereichsauswahl, der Art der Daten und aus der Benutzeranmeldung schließen lässt, werden automatisch ausgefüllt, um den Anwendern unnötige Eingaben zu ersparen, gleichzeitig aber möglichst viel Kontextinformation zu erfassen. Automatisch ausgefüllte Felder sind in Abbildung 3.17 z. B. Start- und Endzeitreihe (Spalte im CSV-Dokument) sowie Start- und Endposition des Auswahlbereichs, die Namen der gewählten Zeitreihen, Angaben über den Messort (ausgelesen aus den Metadaten der zugehörigen Rohmessdaten), Startzeit und Dauer des Auswahlbereichs, Benutzername, Erstellungszeit bzw. letzte Änderung des Kommentars usw. Bereits durch Benutzer kommentierte Bereiche werden in der Visualisierung automatisch markiert, indem am oberen Bildrand für jeden kommentierten Bereich eine gelbe Linie eingeblendet wird, deren Länge proportional zur Dauer des kommentierten Zeitbereichs ist.

Die Kommentarfunktion ermöglicht so eine nachvollziehbare Zusammenarbeit mehrerer Wissenschaftler bei der explorativen Zeitreihenanalyse. Hierdurch wird z. B. eine Arbeitsteilung möglich, bei der mehrere Analysten zunächst in einer frühen Analysephase unbekannte Zeitreihendaten explorieren und erste Erkenntnisse und Besonderheiten kommentieren. Zu einem späteren Zeitpunkt kann ein Fachspezialist dann gezielt die als Auffälligkeiten markierten und kommentierten Bereiche unter fachlichen Gesichtspunkten weiter untersuchen und bei Bedarf Rücksprache mit den Kommentatoren halten.

---

<sup>28</sup> Für die generische Erzeugung der Eingabemasken wird die Java-Reflection API genutzt, um die einzelnen Felder auszulesen und automatisch typabhängige Eingabefelder zu erzeugen.

Abs. 3.6 – Neue interaktive Zeitreihenexploration und skalierbare visuelle Analyse

csvStartCol	21
csvEndCol	29
csvStartRow	26.193
csvEndRow	47.740
csvCols (String-Array):	OW6 [%], OW7 [%], OW8 [%], OW9 [%], OW10 [%], OW11 [%], OW12 [%], OW13 [%], OW14 [%]
edrUnitName	MS0026c7c0a961
edrMACAdr	
channel	21
gpsLatitude	49,096
gpsLongitude	8,433
gpsAltitude	145,2
describedFileNames	KIT-VSU-LOG_MS0026c7c0a961_2_T20130617_L1.csv
describedFileDirPath	E:/_down_test/_DATA_/original data/20130617_SpannungseinbrKitN14Uhr/KIT-VSU-LOG_MS0026c7c0a961_2_T20130617_L1.csv, E:/_down_test/_DATA_/original data/20130617_SpannungseinbrKitN14Uhr
startTime	2013/07/17, 07:16:48:0
timeRangeMillis	21547000
createdBy	jc3561
lastChangedBy	jc3561
timeOfCommentCreation	2015/05/12, 12:52:57:151
timeOfLastCommentChange	2015/05/12, 12:52:57:151
header	
comment	
keywords	
URL	
type	eventComment

OK

**Abbildung 3.17:** Beispiel einer Eingabemaske für Nutzerkommentare zu einer Bereichsauswahl – hier für Ereignisse in EDR-Merkmaldateien, mit bereits automatisch vorausgefüllten Feldern.

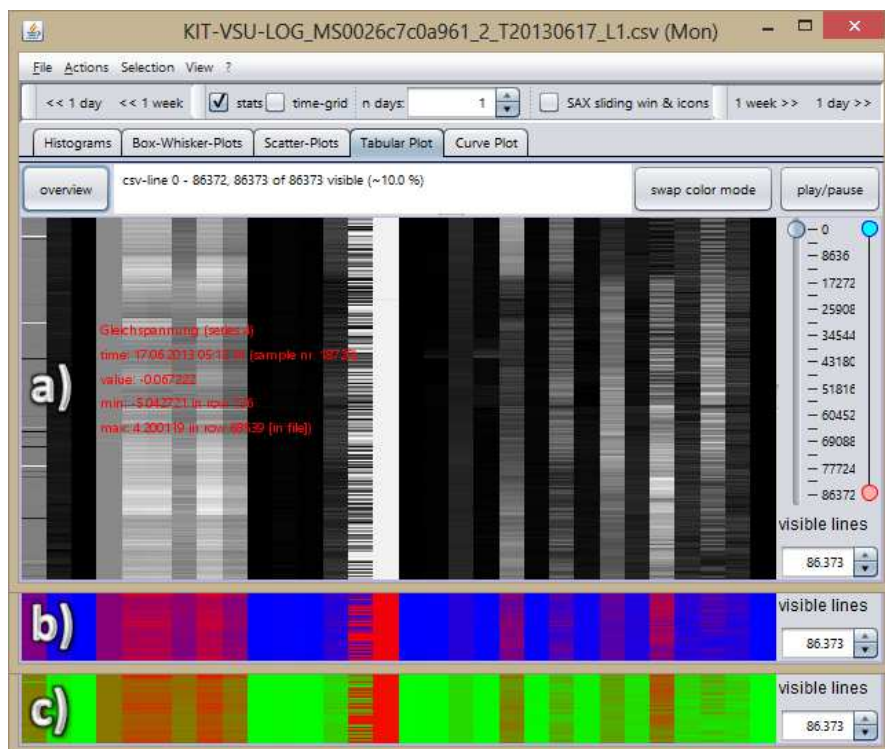
#### 3.6.4 Tabellarische Farbdarstellung

Die tabellarische Farbdarstellung wurde als Alternative zur Kurvendarstellung entwickelt, da sie v. a. den Vorteil bietet, dass sehr viele Zeitreihen parallel dargestellt werden können. Da in der Kurvendarstellung die Werte räumlich kodiert werden und der Darstellungsbereich der multivariaten Ansicht zwischen allen Kurven aufgeteilt werden muss, ist die Anzahl gleichzeitig darstellbarer Kurven hier beschränkt. Je nach Größe des Displays können bis zu ca. 100 Kurven gestapelt werden, ohne die Darstellung unkenntlich zu machen. Die Detaildarstellung als Kurve bleibt davon unberührt, da die Anzeigegröße hier unabhängig angepasst werden kann.

Gerade eine multivariate Ansicht vieler Kurven erlaubt jedoch oftmals eine schnelle Erkennung von Interkorrelationen. Daher wurde eine weitere Zeitreihen-Visualisierung entwickelt, welche die Werte einer Zeitreihe nicht räumlich, sondern farblich kodiert. Die tabellarische Darstellung skaliert sehr gut mit der Anzahl anzuzeigender Zeitreihen. So reicht theoretisch ein einzelner Pixel aus, um einen einzelnen Datenpunkt einer Zeitreihe darzustellen. Besser erkennbar ist jedoch eine Farbflächendarstellung. Die Zeitachse verläuft in dieser Ansicht vertikal und die Zeitreihen sind wiederum von oben nach unten in der Reihenfolge angeordnet, in der die Daten gelesen wurden. Die tabellarische Farbdarstellung ist somit für CSV-basierte Zeitreihenformate interessant, da die tabellarische Darstellung hier exakt die Anordnung der Daten widerspiegelt.

Es stehen drei unterschiedliche Farbdarstellungen zur Verfügung (siehe Abbildung 3.18). Alle haben gemeinsam, dass Werte bei der Abbildung auf Farben zwischen zwei Farbtönen verteilt werden, so dass beispielsweise bei der Grauwertdarstellung und bei Min-Max Skalierung die niedrigsten Werte schwarz und die höchsten Werte weiß angezeigt werden. Ansonsten ist die Bedienung ähnlich wie in der Kurvendarstellung. Es steht dasselbe Kontextmenü zur Verfügung und auch die Navigationsleisten am rechten Rand und die Information im oberen Teil sind identisch.

Eine Detailansicht steht jedoch nicht zur Verfügung, da durch die Farbcodierung der Werte eine Anzeige auf größerer Fläche keinen Vorteil bietet. Die Probleme durch Aliasing und fehlende Information bei der Darstellung vieler Elemente auf kleinem Raum können in dieser Ansicht nicht wie in der Kurvenansicht durch Zeitbereichsstatistiken gelöst werden. Daher sollte zu starke Unterabtastung hier generell vermieden werden. Dadurch müssen die Zeitreihen in der Darstellung in zeitlicher Richtung manuell durchlaufen werden, etwa durch Drehen des Mousrads, Drücken der Pfeiltasten oder Benutzung der Zeitbereichs- und Positionsleisten („scrollen“), um eine vollständige Exploration zu ermöglichen, bei der keine Elemente ausgelassen werden. Da dies umständlich und ermüdend sein kann, wurde außerdem eine Möglichkeit zum automatischen Scrollen geschaffen. So können die Zeitreihen optional als „Film“ betrachtet werden, indem der Button `Play/Pause` gedrückt wird.



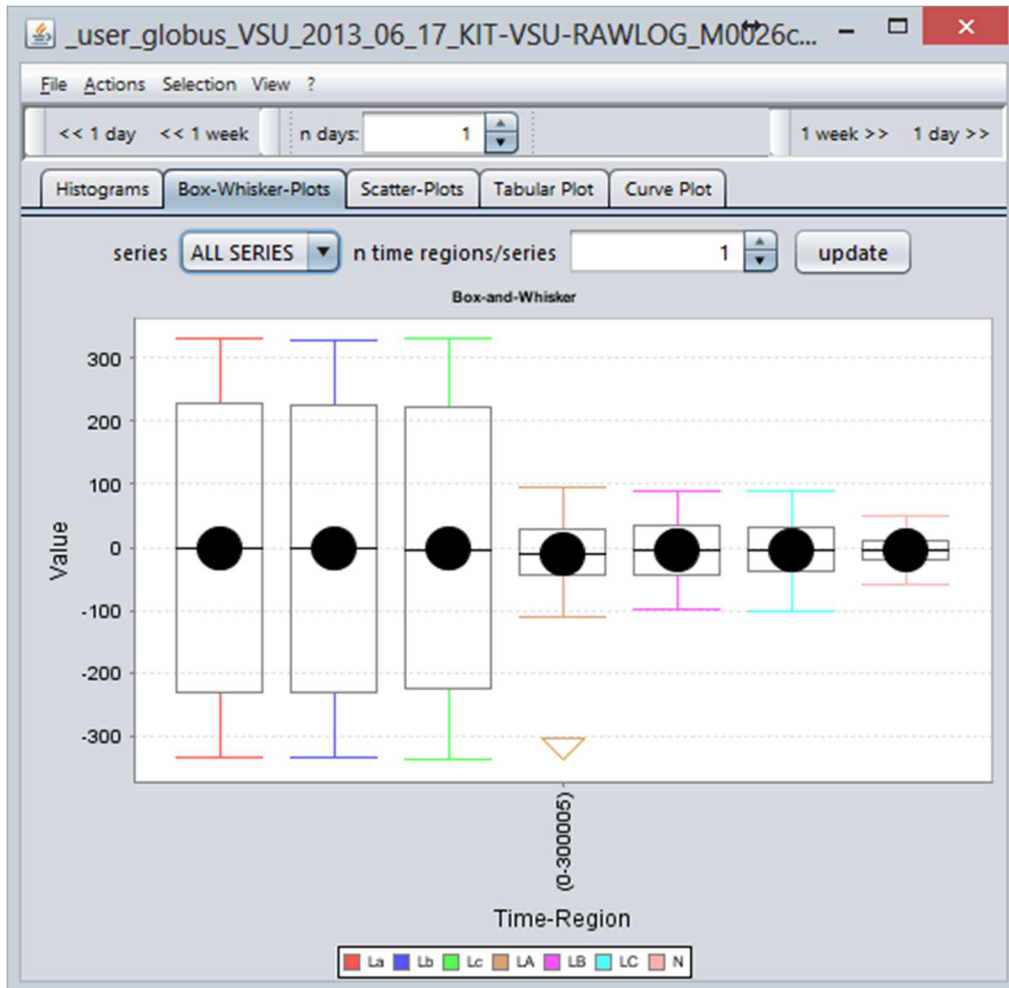
**Abbildung 3.18:** Multivariate tabellarische Farbflächendarstellung in ViAT, a) mit Grauwerten, b) mit Blau-Rot- und c) mit Grün-Rot-Verlauf.

### 3.6.5 Interaktiv parametrisierbare Box-Whisker-Plots zur Wertebereichs- und aggregierten Verlaufsdarstellung

Als eine weitere Zeitreihenvisualisierung wurden Box-Whisker-Plots [298] in ViAT integriert, da sie zahlreiche interessante Möglichkeiten bieten, aggregierte Zeitreihen zusammen mit Aggregat-Statistiken effizient anzuzeigen. Außerdem eignen sie sich hervorragend für eine Übersicht und einen Vergleich der Wertebereiche mehrerer Zeitreihen. Als Beispiel hierfür werden in Abbildung 3.19 die Wertebereiche der Zeitreihen aus einem EDR-Messdatensatz für eine Minute als Box-Whisker-Diagramm gezeigt. Es können darin klar drei Gruppen von Wertebereichen unterschieden werden: Die ersten drei Wertebereiche gehören zur dreiphasigen Spannung, die nächsten drei zum dreiphasigen Strom und der letzte repräsentiert den Strom im Neutralleiter.

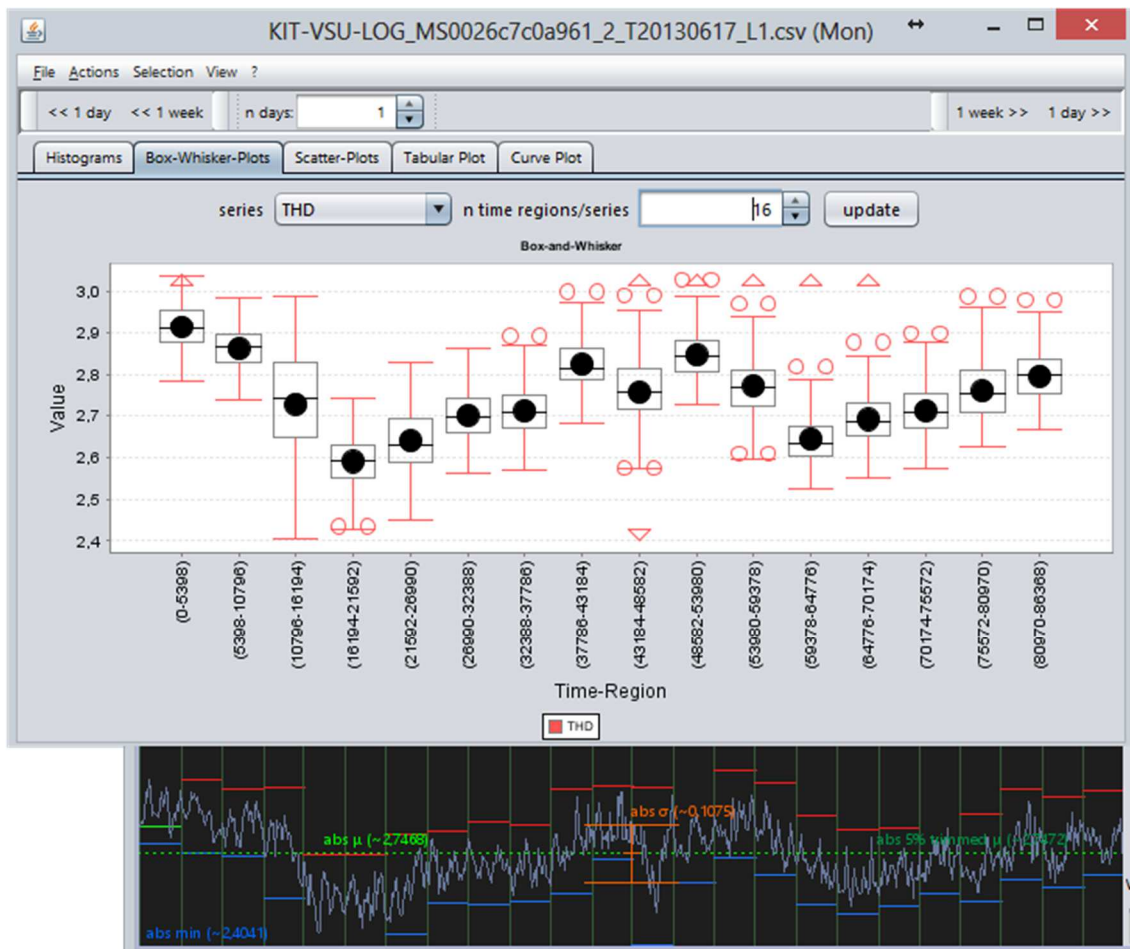
Die Box-Whisker-Darstellung aggregierter Zeitreihen bietet ähnliche Vorteile wie die Zeitbereichsstatistiken in der Kurvendarstellung und wurde daher für ViAT implementiert. Durch eine horizontale Sequenz von Box-Whisker-Zeitbereichsstatistiken wird nicht nur der grobe Verlauf der Kurve übersichtlich und kompakt, mit einstellbarer zeitlicher Auflösung dargestellt, sondern darüber hinaus auch der Verlauf der Extremwerte, der Werteschwankungen und der Ausreißer.

Eine zeitlich hochaufgelöste Ansicht, wie sie die Kurvendarstellung bietet, ist hier jedoch nicht vorhanden, ebenso sind die Interaktionsmöglichkeiten beschränkt.



**Abbildung 3.19:** Box-Whisker-Plot der unterschiedlichen Wertebereiche von sieben Zeitreihen (hier EDR-Rohdaten mit Spannung und Strom in je drei Phasen und Strom im Neutralleiter).

Eine Zeitreihe wird für die Darstellung zunächst in eine frei wählbare Anzahl gleich großer Abschnitte unterteilt. Jedes der Zeitaggregate wird dabei als einzelnes Box-Whisker-Element dargestellt, welches Median, arithmetisches Mittel, Quartile und Ausreißer darstellt. Die abschnittsweise Boxplot-Darstellung eignet sich dadurch besonders gut, um langfristige Entwicklungen der Zeitreihe unabhängig von lokalen Details einschätzen zu können. Beispielsweise können Trends erkannt werden, die nicht nur Mittelwerte umfassen, wie es etwa bei gleitenden Mittelwerten der Fall ist, sondern auch die zeitliche Entwicklung von Wertebereich und Ausreißern. Sollen sehr

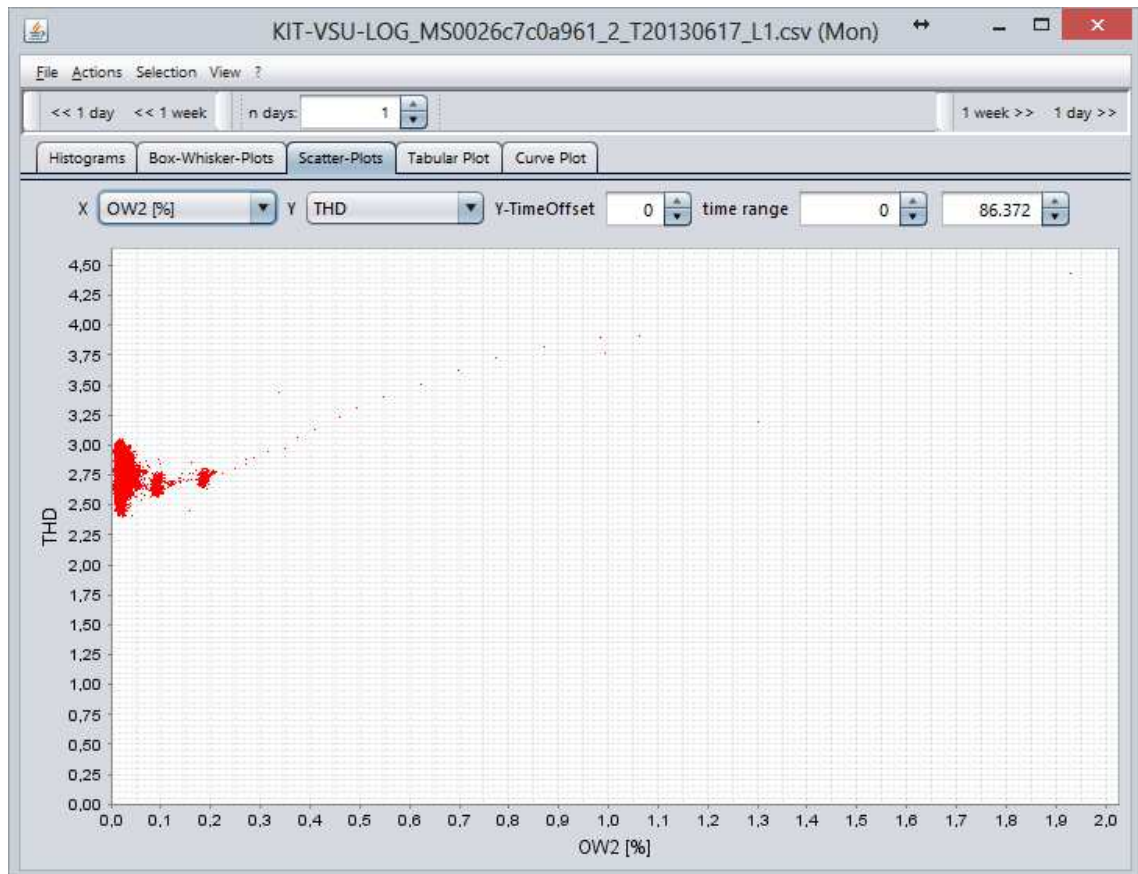


**Abbildung 3.20:** Box-Whisker Zeitreihendarstellung der harmonischen Gesamtverzerrung (THD) über einen Messtag mit 16 Zeitbereichen (oben) im Vergleich zur Kurvendarstellung mit stündlichen Zeitbereichsstatistiken (unten).

viele Messdatensätze mit unterschiedlicher Datenqualität als zusammenhängende Zeitreihe untersucht werden, ist diese Ansicht ideal, um die Entwicklung der Datenqualität zu beurteilen und einzelne Datensätze qualitativ miteinander zu vergleichen, da stark fehlerbehaftete Messungen sofort anhand der großen Zahl an Ausreißern erkannt werden. Ebenso können Einbrüche und Bereiche mit viel Dynamik und starken Werteschwankungen oft leichter erkannt werden als in der Kurvendarstellung, wie der Vergleich in Abbildung 3.20 zeigt. Zur Erzeugung der Box-Whisker-Diagramme sowie der Scatter-Plots und der Histogramme wurde die quelloffene Java-Bibliothek *jFreeChart* verwendet. Die Anzahl darzustellender Elemente ist hier durch die zeitliche Aggregation im Vergleich zur Kurvendarstellung eher überschaubar, weshalb Skalierungsprobleme nur bei einer sehr hohen Anzahl an Zeitbereichen zu erwarten sind. Trotzdem wurde der Quellcode von *jFreeChart* in der Version 1.0.14 teilweise modifiziert, um die Anzeige vieler Elemente zu optimieren und schnelle Reaktionszeiten zu ermöglichen, indem die Aufrufe der Zeichnungsfunktionen für jedes einzelne Diagramm zusammengefasst wurden.

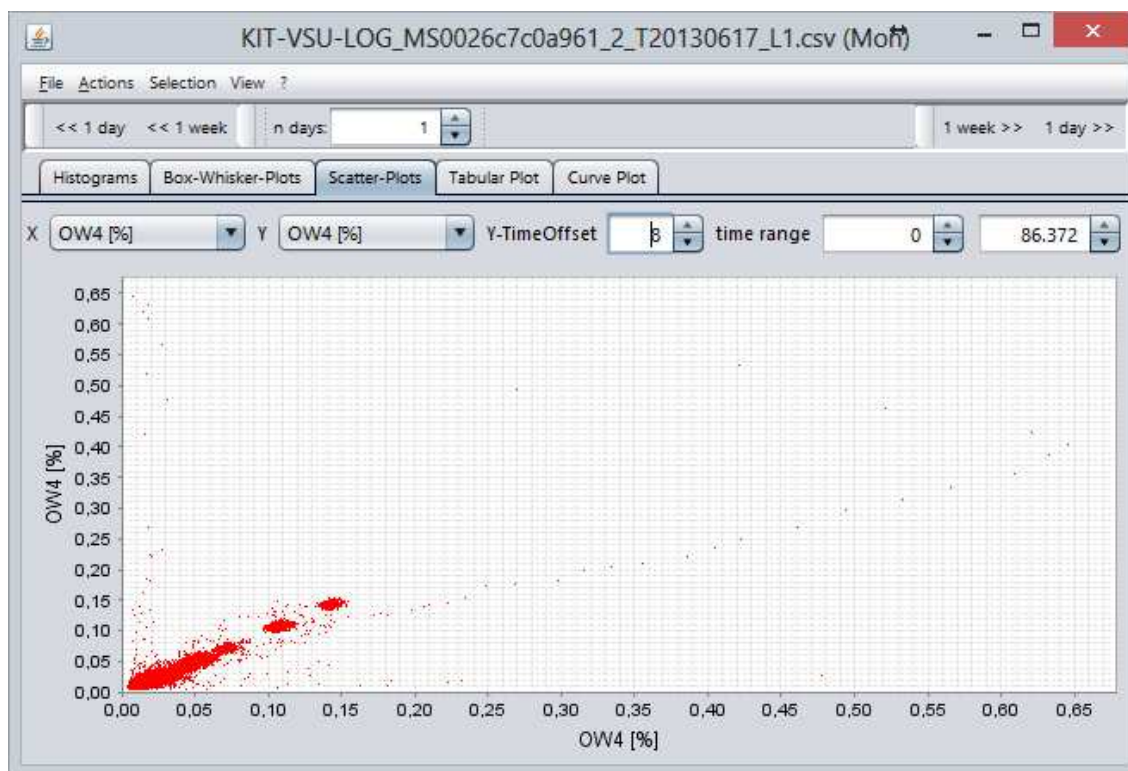
### 3.6.6 Scatterplots zur interaktiven Korrelations- und Autokorrelationsanalyse

Soll die Beziehung zwischen zwei zusammenhängenden Variablen visualisiert werden, eignen sich Streudiagramme (engl. Scatterplots [85]). Daher wurde in *ViAT* ein konfigurierbares Streudiagramm integriert, über welches die Wertestreuung von je zwei Zeitreihen in einer gemeinsamen Ansicht dargestellt werden kann (siehe Abbildung 3.21).



**Abbildung 3.21:** Streudiagramm zur Visualisierung des Zusammenhanges zwischen der prozentualen zweiten Oberschwingung (OW2) und der harmonischen Gesamtverzerrung (THD) der Spannung aus EDR-Merkmalen für einen Tag, ohne Zeitversatz.

Wertepaare aus zwei Merkmalen werden als Punkte in einem gemeinsamen kartesischen Koordinatensystem dargestellt, dessen Achsen die Wertebereiche der jeweiligen Daten vollständig zeigen. Bei starker Korrelation ergibt sich eine diagonale Linie, bzw. eine diagonal gestreckte Streuung (wie in Abbildung 3.22), während lose Punktwolken auf Zeitreihen ohne signifikante zeitliche Korrelation hindeuten (wie in Abbildung 3.21).



**Abbildung 3.22:** Streudiagramm als Lag-Plot der vierten harmonischen Schwingung, mit zeitlichem Versatz von acht Einzelwerten (für OW4). Die diagonal verteilte Streuung deutet auf eine nicht-zufällige Wertefolge hin.

Im oberen Bereich des Scatterplots wurden Steuerelemente zur Auswahl der darzustellenden Zeitreihen, zur Einstellung eines zeitlichen Versatzes (als Anzahl von Werten) und des berücksichtigten Bereichs integriert. Änderungen dieser Einstellungen werden sofort in der Visualisierung sichtbar, so dass schnell alle Variablen nacheinander durchlaufen und visuell auf Korrelationen geprüft werden können. Die Möglichkeit eines fixen zeitlichen Versatzes ist für gewöhnlich nicht in Streudiagrammen enthalten und wurde hier v. a. deshalb hinzugefügt, um die Zufälligkeit einer einzelnen Zeitreihe visuell von chaotischem Verhalten unterscheidbar zu machen. Man spricht bei dieser Darstellung von einem sog. *Poincaré*-<sup>29</sup> [101] oder *Lag-Plot* (siehe Abbildung 3.22).

Es werden dabei analog zum Streudiagramm zwei Zeitreihen gegenübergestellt, die hier jedoch zwei Varianten derselben Zeitreihe sind, wobei eine der beiden um eine wählbare konstante Schrittweite (typischerweise um einen Einzelwert) zeitlich verschoben wird. Ergibt sich hierbei eine diagonal gestreckte Streuung, deutet das darauf hin, dass die Einzelwerte einer Zeitreihe in ihrer Abfolge starke Abhängigkeiten aufweisen (wie in Abbildung 3.22), während eine breite

<sup>29</sup> Dem Namensgeber Henri Poincaré (1854 – 1912) wird u. a. eine frühe Verwendung des *Lag-Plots* zugeschrieben [101].



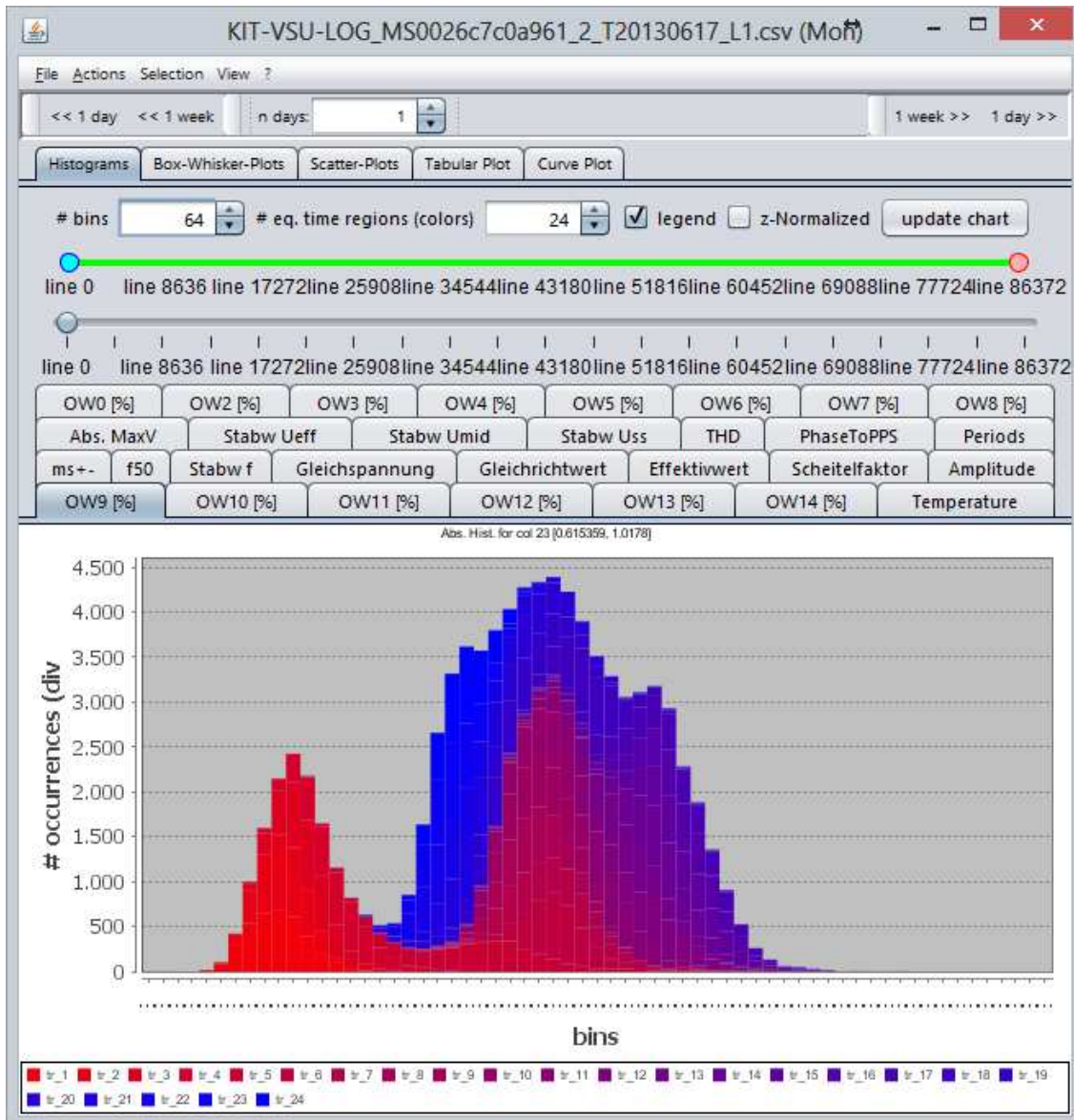
Streuung oder stark horizontal oder vertikal gestreckte Streuungen auf eher zufällige Werte und fehlende zeitliche Korrelation hinweisen. Diese Prüfung ist beispielsweise relevant für die Modellwahl, sobald eine Zeitreihe modelliert werden soll. Für deutlich als nicht-zufällig befundene Sequenzen eignen sich beispielsweise autoregressive Modelle gut, die jedoch nur bedingt auf Zeitreihen mit ausgeprägtem stochastischem Verhalten angewandt werden können. Die Darstellung als Streudiagramm mit Zeitversatz für eine der Zeitreihen kann außerdem dazu verwendet werden, um zeitlich verschobene Zusammenhänge zwischen zwei Zeitreihen sichtbar zu machen.

### 3.6.7 Zeitreihen-Histogramme mit Verlaufsdarstellung

Eine weitere kompakte Darstellungsmöglichkeit der Charakteristik einer Zeitreihe wird in *ViAT* durch Histogramme geboten. Durch sie können die Auftrittshäufigkeiten im Wertebereich dargestellt werden. Dazu wird der Wertebereich in eine einstellbare Anzahl gleich großer Teilbereiche (sog. *Bins*) aufgeteilt und die Zahl der in die einzelnen Bereiche fallenden Werte ermittelt. Die Zählungen der Werte innerhalb eines jeden Bereichs werden als Balken gleicher Breite dargestellt, deren Fläche die Anzahl kodiert. Histogramme haben dadurch den Vorteil, dass sich hier der Platzbedarf nicht mit zunehmender Zeitreihenlänge vergrößert, sondern konstant bleibt. Histogramme eignen sich besonders gut zur Beurteilung der Modalität von Zeitreihen und können auch für eine Ähnlichkeitsbeurteilung zweier Zeitreihen herangezogen werden, die unabhängig von zeitlichen Verschiebungen und Stauchungen bzw. Dehnungen ist.

Da gewöhnliche Histogramme dadurch den zeitlichen Bezug verlieren, wurde ein *Zeit-Histogramm* entwickelt, welches das Fortschreiten der Zeit durch einen linearen Farbverlauf darstellt (siehe Abbildung 3.23). Man kann sich die Erzeugung eines solchen Zeit-Histogramms so verbildlichen, dass für jeden Teilbereich des Wertebereichs ein Zylinder aufgestellt wird, in welchen dann farbige Sandkörner rieseln und Säulen bilden. Die Wahl des Zylinders wird dabei vom jeweiligen Wert bestimmt und die Farbe des Sandkorns ändert sich mit jedem Zeitschritt, so dass nachträglich ersichtlich ist, zu welcher Zeit ein Sandkorn hinzugefügt wurde. Des Weiteren wurden Steuerelemente integriert, welche es erlauben, eine Zeitspanne für die Eingangsdaten auszuwählen und diese interaktiv zu verschieben. Das ermöglicht eine schnelle detaillierte Untersuchung einzelner Zeitbereiche und außerdem – durch allmähliche Verschiebung des Zeitbereichs – eine animierte Darstellung der zeitlichen Entwicklung des Histogramms eines gewählten Zeitbereichs, was aus der Bildsequenz in Abbildung 3.24 ersichtlich wird. Die neuen Darstellungsmöglichkeiten bieten eine effiziente abstrakte Übersicht über die Charakteristik beliebig langer Zeitreihen unter Erhalt des Zeitbezugs. Multimodale Zeitreihen (wie in Abbildung 3.23) werden so sofort erkannt und die Darstellung ermöglicht eine Einschätzung, zu welcher Zeit die Zeitreihe sich in welchem Teil des Wertebereichs bewegt. Sie ermöglicht somit auch die Identifikation von Zuständen, in denen sich das beobachtete System zu bestimmten Zeiten aufhält. Abbildung 3.23

zeigt ein Beispiel einer multimodalen Werteverteilung in der Zeitreihe der neunten Spannungs-Oberschwingung einer EDR-Merkmaldatei über einen Tag. Die farbliche Kodierung der Zeit durch einen linearen Farbverlauf von rot (00:00 Uhr bei Messbeginn) nach blau (00:00 Uhr am Ende des Messtages) lässt erkennen, dass die Zeitreihe sich von einem Bereich eher niedriger Werte zu Beginn des Tages hin zu höheren Werten entwickelt. Am Ende des Tages werden die Werte wieder geringer.

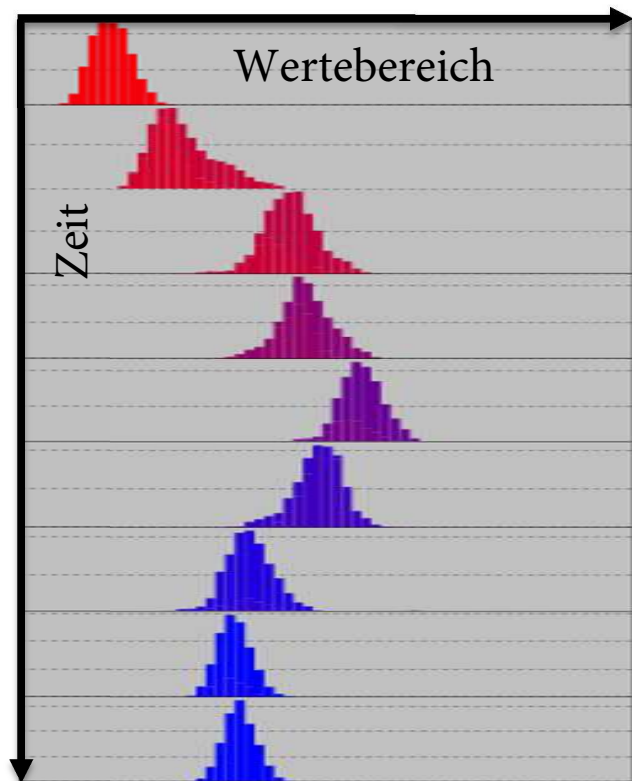


**Abbildung 3.23:** Zeit-Histogramm über 24 Stunden der neunten Oberschwingung (OW9) der Spannung einer EDR-Merkmaldatei mit Wertebereichsaufteilung in 64 Bins und zeitlicher Farbkodierung durch 24 Farbstufen.

Die Verteilung zeigt deutlich zwei Modi, von denen die linke (rote) Häufung nahezu ausschließlich durch sehr frühe Messwerte hervorgerufen wird. Noch genauer kann die Entwicklung untersucht werden, indem der Zeitbereichsregler (grün in Abbildung 3.23) auf ein fixes, kleineres Zeitintervall eingestellt wird, so dass sich das Histogramm für diesen Bereich in eine unimodale Gaußverteilung verwandelt. Nun kann mit dem Positionsregler unterhalb des Zeitbereichsreglers der Ausschnitt verschoben werden, so dass die Entwicklung des Wertebereichs als animiertes Histogramm beobachtet werden kann. Die Bildsequenz in Abbildung 3.24 zeigt das Ergebnis dieses Verschiebevorgangs von links (Messbeginn) nach rechts (Ende des Messzeitraums).

Es ist außerdem optional möglich, für die Eingangszeitreihe eine Z-Normalisierung

(siehe Anhang A.1.2.4) vor Erstellung des Histogramms durchzuführen. Hierdurch werden Histogramme für Zeitreihen mit unterschiedlichen Wertebereichen besser vergleichbar. Außerdem kann die Legende, welche die farbliche Zuordnung zu Zeitbereichen darstellt, ausgeblendet werden, um mehr Platz für das Histogramm zur Verfügung zu stellen.



**Abbildung 3.24:** Effekt der Zeitbereichsverschiebung von Beginn (oben, rot) nach Ende (unterstes Bild, blau) der neunten Oberschwingung auf das Bereichs-Histogramm, als Bildsequenz.

### 3.7 Neue SAX-basierte Analysewerkzeuge für ViAT

Die aus Zeitreihendaten abgeleiteten Metadaten (siehe Abs. 3.4) enthalten neben diversen Statistiken für die einzelnen Zeitreihen auch eine symbolische Darstellung in je drei unterschiedlichen Detailgraden. Eine solche SAX-Repräsentation kann außerdem auch direkt aus den Originaldaten gebildet werden, z. B. auch für Teilsequenzen mithilfe eines Fensterverschiebungsalgorithmus [153].

Die SAX-Darstellung einer Zeitreihe als Zeichenkette von Symbolen erlaubt den Einsatz spezieller Methoden des Data-Mining, v. a. ähnlichkeitsbasierter unscharfer Suchen. Für die Durchführung

und Visualisierung der SAX-Codierung selbst und der darauf basierenden neuen Methoden wurden interaktive Softwarewerkzeuge entwickelt und in *ViAT* integriert, die in den folgenden Unterabschnitten behandelt werden, wie z.B. ein Codierungseditor, ein ähnlichkeitsbasiertes Clustering, ein Werkzeug zur Analyse der Strukturcharakteristik, ein Sequenzeditor für unscharfe Suchen nach bekannten und unbekanntem Mustern, Sequenz-Histogramme und Werkzeuge zum Auffinden seltener und häufiger Sequenzmuster. Sie erlauben eine interaktive Parametrisierung und eine grafische Darstellung des Codierungsablaufs und unterstützen somit die Analyse und verbessern die Nachvollziehbarkeit der Methoden. Durch eine sofortige visuelle Rückmeldung der Auswirkung von Parameteränderungen wird eine schnelle effektive Arbeitsweise ermöglicht, durch welche die abstrakten komplexen Methoden für den Anwender durchschaubar und nutzbar gemacht werden.

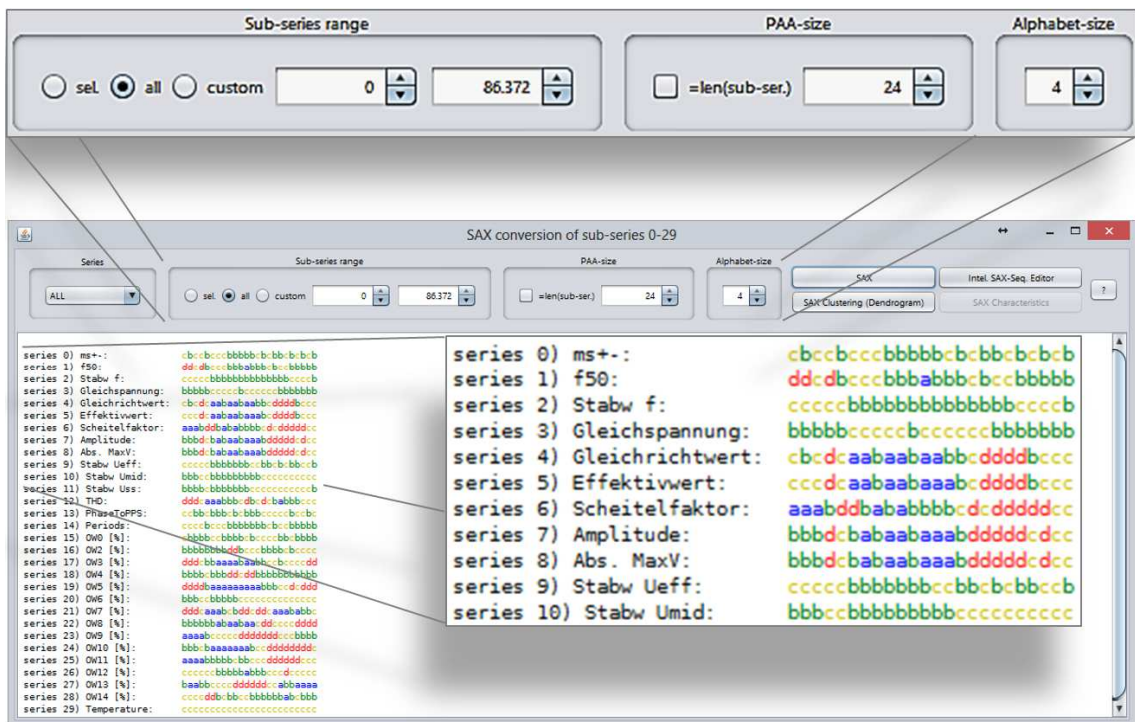
### 3.7.1 Interaktiv parametrisierbare SAX-Codierung in ViAT

Zunächst wurde eine Visualisierung der SAX-Codierung selbst in *ViAT* integriert, die über das Kontextmenü gestartet werden kann und in Abbildung 3.25 zu sehen ist. Hierin können alle für die Codierung relevanten Parameter eingestellt werden. Anschließend kann durch einen Klick auf die Schaltfläche „SAX“ die Codierung angestoßen werden, die gemäß der Beschreibung in Abs. 3.4.3 durchgeführt wird. Die Resultate werden direkt in einem Textfeld unterhalb der Kontrollelemente angezeigt, das speziell für die formatierte Darstellung von SAX-Sequenzen entwickelt wurde.

Die einzustellenden SAX-Parameter sind (von links nach rechts in Abbildung 3.25):

- Zu codierende Zeitreihe (oder „ALL“ für alle)
- Zeitbereich (Selektion, vollständig oder Bereichswahl)
- SAX-PAA-Größe (Anzahl zeitlicher Aggregate)
- SAX-Alphabetgröße (Anzahl symbolischer Aggregate).

Die Parameterzahl ist damit – vor allem im Vergleich mit anderen dimensionsreduzierenden Zeitreihenrepräsentationen – recht überschaubar. Die Visualisierung der Auswirkungen von Parameteränderungen erlaubt es Benutzern, sich sehr schnell experimentell mit der SAX-Codierung vertraut zu machen.



**Abbildung 3.25:** Dialog zur SAX-Codierung, mit Vergrößerungsansicht der Kontrollelemente zur Parametereinstellung (oben) und der farbcodierten Textanzeige des Resultats (rechts).

Für das in Abbildung 3.25 gezeigte Codierungsergebnis wurden alle Zeitreihen über den gesamten Zeitbereich mit einer PAA-Größe von 24 und einer Alphabetgröße von vier codiert. Da die Eingangszeitreihen wiederum aus einem Zeitreihendatensatz für einen Tag stammen, bewirkt die Wahl einer PAA-Größe von 24, dass jedes Symbol den Mittelwert über eine Stunde codiert – hier also 3600 Werte (86400 Messwerten pro Tag, geteilt durch 24). Als Symbole wurden die Buchstaben des lateinischen Alphabets gewählt, wobei der Buchstabe *a* niedrige Werte – und der Buchstabe *d* hohe Werte repräsentiert.

Das Textfeld, in welchem das Ergebnis präsentiert wird, wurde speziell für die Darstellung von SAX-Sequenzen neu entwickelt und erlaubt die automatische farbliche Kodierung der einzelnen Zeichen (*a* blau, *b* grün, *c* gelb und *d* rot). In Kombination mit der gestapelten bündigen Listung der einzelnen SAX-Sequenzen ist es damit möglich, die Struktur der einzelnen Zeitreihen schnell visuell zu erfassen. Zeitliche Korrelationen zwischen den Zeitreihen werden so bereits in dieser Ansicht sichtbar. Falls die aktuell in *ViAT* geladenen Daten Bezeichnungen der einzelnen Zeitreihen enthalten, werden die Namen ebenfalls aufgeführt.

Der SAX-Codierungs-Dialog stellt sowohl ein mächtiges Analysewerkzeug als auch einen Experimentierkasten dar, mit dem Anwender die Codierung anschaulich erproben können. Die Resultate können flexibel kopiert und weiterverwendet werden, da es sich lediglich um Textdaten

handelt. Es ist jedoch auch möglich, direkt aus der Darstellung heraus weiterführende SAX-basierte Analysewerkzeuge zu starten, welche die eingestellten Codierungs-Parameter und die Daten übernehmen. Beispielsweise kann ein SAX-basiertes *Clustering* durchgeführt und als Dendrogramm [153] angezeigt werden, mit einem *SAX-Sequenzeditor* nach unbekanntem oder bekannten Mustern gesucht werden oder – sofern eine einzelne Zeitreihe codiert wurde – die strukturelle SAX-Charakteristik analysiert werden. Diese Werkzeuge werden in den Folgeabschnitten vorgestellt.

### 3.7.2 SAX-basiertes Clustering mit Dendrogramm-Darstellung

Zur Bewertung der Ähnlichkeit von Zeitreihen wurde eine SAX-basierte Clustering-Methode konzipiert und implementiert, deren Resultate in einem Dendrogramm dargestellt werden können. Hierzu wurde u. a. ein geeignetes Ähnlichkeitsmaß bzw. Distanzmaß benötigt, über welches ein Ähnlichkeitswert für Zeitreihenpaare ermittelt werden kann. Anhand der Ähnlichkeitswerte kann dann ein hierarchisches Clustering durchgeführt und als Dendrogramm dargestellt werden.

Die in Abbildung 3.25 gezeigte SAX-Repräsentation von Zeitreihen bildet die Basis der Ähnlichkeitsbewertung, die auch in diversen anderen Modulen benötigt wird. Das ermöglicht beispielsweise über das Clustering hinaus die Realisierung schneller unscharfer Ähnlichkeitssuchen aufgrund vorberechneter Metadaten, welche SAX-Repräsentationen der einzelnen Zeitreihen in verschiedenen Detailgraden enthalten. Es wurden zwei unterschiedliche Distanzmaße implementiert:

- Levenshtein-Distanz [173]
- SAX-MINDIST [181]

Das SAX-MINDIST-Maß zeigt sich jedoch wesentlich besser geeignet für Zeitreihendaten, da es nicht – wie die Levenshtein-Distanz – nur die Anzahl an Editieroperationen berücksichtigt, die nötig sind, um eine Zeitreihe in eine andere zu überführen, sondern auch die tatsächlichen numerischen Abstände. MINDIST wurde außerdem so entworfen, dass damit berechnete Zeitreihendistanzen stets den „tatsächlichen“ Euklidischen Abstand zwischen den ursprünglichen Zeitreihen unterschreiten. Somit ist sichergestellt, dass etwa bei Ähnlichkeitssuchen keine falschen Negativresultate auftreten, falsche Positive können jedoch aufgrund der Vergrößerung durch SAX vorkommen. In manchen Anwendungsbereichen – etwa bei der Suche in Texten – mag jedoch eine Edit-Distanz wie die Levenshtein-Distanz ebenfalls zu guten Ergebnissen führen [201].

Die SAX-Parameterwahl kann das Ergebnis der Distanzberechnung stark beeinflussen. Wird beispielsweise ein verhältnismäßig hoher Wert für die PAA-Größe gewählt, der sich nicht allzu sehr von der Länge der Ursprungszeitreihe unterscheidet, so fließen sämtliche lokalen Strukturdetails

in die Ähnlichkeitsbewertung ein. Mit kleinen PAA-Werten, wie dem in Abbildung 3.25 zu sehenden Wert von 24 werden stattdessen sehr viele Messwerte zu einem Mittelwert zusammengefasst und somit wird eher die Ähnlichkeit des groben Kurvenverlaufs bewertet. Die Parameterwahl ist stark anwendungsabhängig und sollte bei Bedarf auf die für die jeweilige Anwendung entscheidenden Merkmale zur Ähnlichkeitsbewertung abgestimmt werden. Ähnlich verhält es sich mit der Wahl der Alphabetgröße. Eine geringe Alphabetgröße (zwei bis vier) reicht bereits für die meisten ähnlichkeitsbasierten Aufgaben aus und kompensiert außerdem automatisch durch Rauschen und kleine Werteschwankungen verursachte Distanzen, während größere Alphabete benötigt werden, wenn die Ähnlichkeit von Zeitreihen bewertet werden sollen, die sich nur geringfügig in der Werteausprägung unterscheiden.

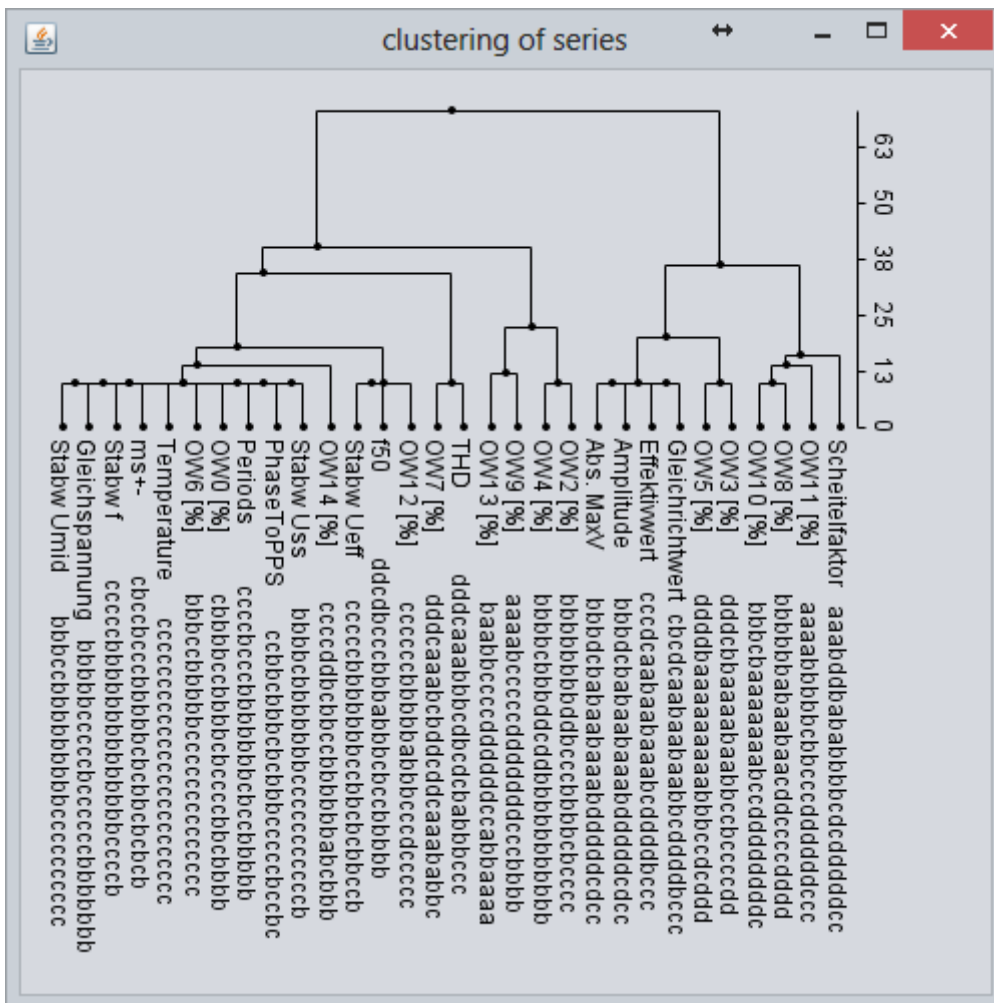
```
public double[][] buildProximityMatrixSeries() {
    double[][] res = new double[saxWords.size()][saxWords.size()];
    for (int i = 0; i < saxWords.size(); i++) {
        for (int j=i; j<saxWords.size(); j++) {
            if (i == j) {
                res[i][j] = 1.;
            } else {
                double similarity =
                    saxWords.get(i).similarity(saxWords.get(j));
                res[i][j] = similarity;
                res[j][i] = similarity;
            }
        }
    }
    return res;
}
```

**Code 3.7:** Java-Code zur Berechnung der Proximitätsmatrix.

Für ein Clustering mehrerer Zeitreihen wird zunächst eine Proximitätsmatrix [283] berechnet, welche die Abstände zwischen allen Zeitreihenpaaren der auszuwertenden Zeitreihendatenbank beinhaltet (siehe Code 3.7). Eine Proximitätsmatrix enthält im Gegensatz zu einer Distanzmatrix Ähnlichkeitswerte, die mit zunehmender Ähnlichkeit ansteigen. Eine Normalisierung dieser Matrix durch Skalierung auf das Intervall [0,1] ermöglicht die Konvertierung in eine Distanzmatrix und vice versa. Hierzu werden alle Ähnlichkeitswerte durch die maximal auftretende Ähnlichkeit dividiert und anschließend invertiert.

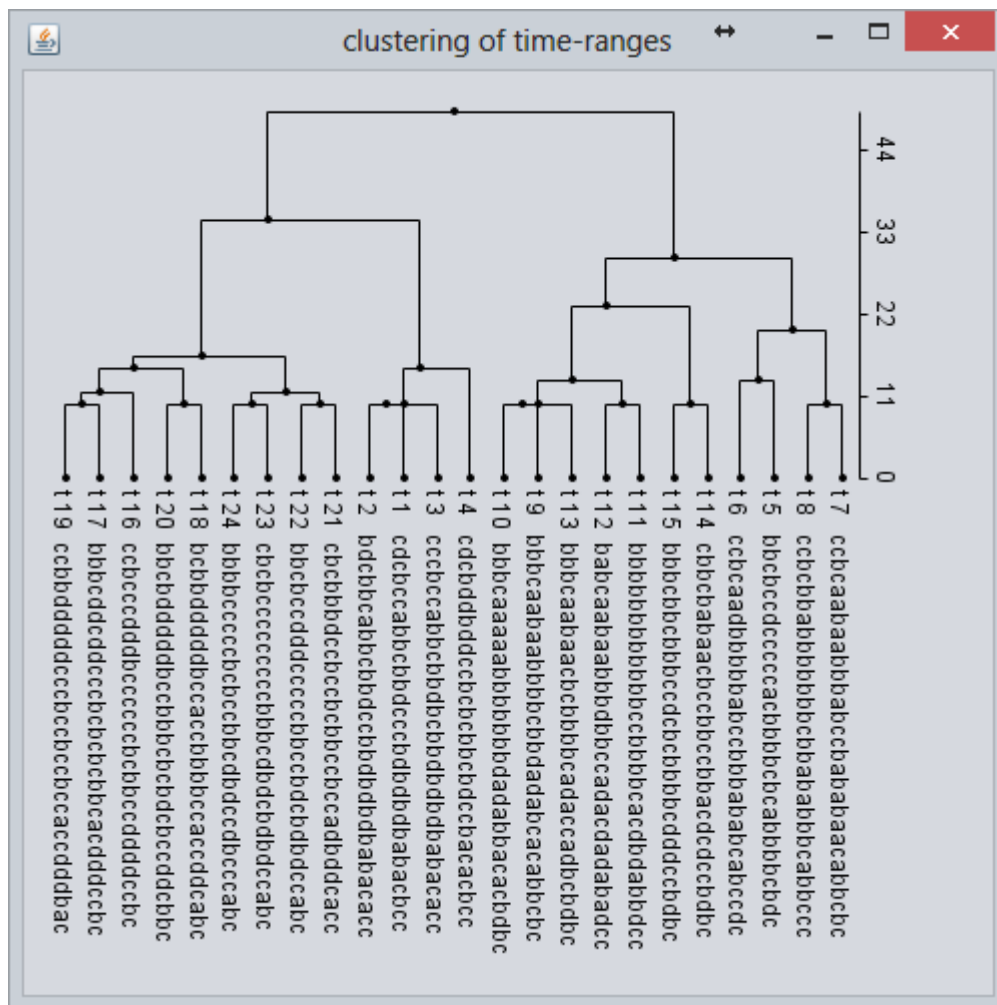
Soll die Ähnlichkeit zweier multivariater Zeitreihen gleichen Typs ermittelt werden, z. B. für zwei EDR-Merkmalssatze, so können die paarweise berechneten Ähnlichkeiten einfach addiert werden, wobei wiederum anwendungsabhängig einzelne Zeitreihen unterschiedlich gewichtet werden können.

Für die Darstellung der Ähnlichkeit wurde auf ein agglomeratives hierarchisches Clustering [283] zurückgegriffen, also auf ein Verfahren, welches zunächst eigene Cluster für jedes Objekt zuweist, welche dann sukzessive zusammenfasst werden, so dass eine hierarchische Struktur entsteht. Hierzu wurde teilweise Code der quelloffenen Clustering-Implementierung von Lars Behnke [171] verwendet. Abbildung 3.26 zeigt ein Beispiel eines generierten Dendrogramms mit hierarchischer Gruppierung der Zeitreihen aus EDR-Merkmalen für einen Tag. Hierfür wurden die Zeitreihen bei einer Alphabetgröße von vier und einer PAA-Größe von 24 SAX-codiert.



**Abbildung 3.26:** Dendrogramm des SAX-basierten hierarchischen Clustering nach Ähnlichkeit zwischen den einzelnen Merkmalszeitreihen aus einem EDR-Merkmalensatz für einen Messtag.





**Abbildung 3.27:** Dendrogramm des SAX-basierten „vertikalen“ Clusterings von Zeitbereichen der Merkmalszeitreihen aus einem EDR-Merkmalssatz für einen Messtag.

Die resultierenden SAX-Sequenzen sind im unteren Teil des Dendrogramms dargestellt. Die Darstellung als Dendrogramm ermöglicht dabei die schnelle Identifikation ähnlicher Clustergruppen. Zusätzlich zum strukturellen Clustering von Zeitreihen wurde ein Clusteringverfahren entwickelt, welches ähnliche Zeitbereiche in multivariaten Zeitreihen gruppiert (siehe Abbildung 3.27).

Wird das Clustering gestartet, wird der Benutzer deshalb zunächst in einem Dialog gefragt, welche Art des Clustering durchgeführt werden soll (siehe Abbildung 3.28). Die Funktionsweise des eigentlichen SAX-basierten Clusterings unterscheidet sich dabei zwischen den beiden Verfahren kaum. Lediglich die Bildung der Eingangssequenzen ist unterschiedlich, was in Abbildung 3.29 verdeutlicht wird.

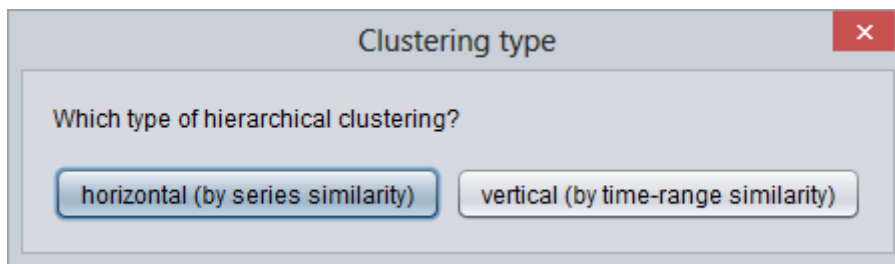


Abbildung 3.28: Dialog zur Auswahl des Clustering-Typs.

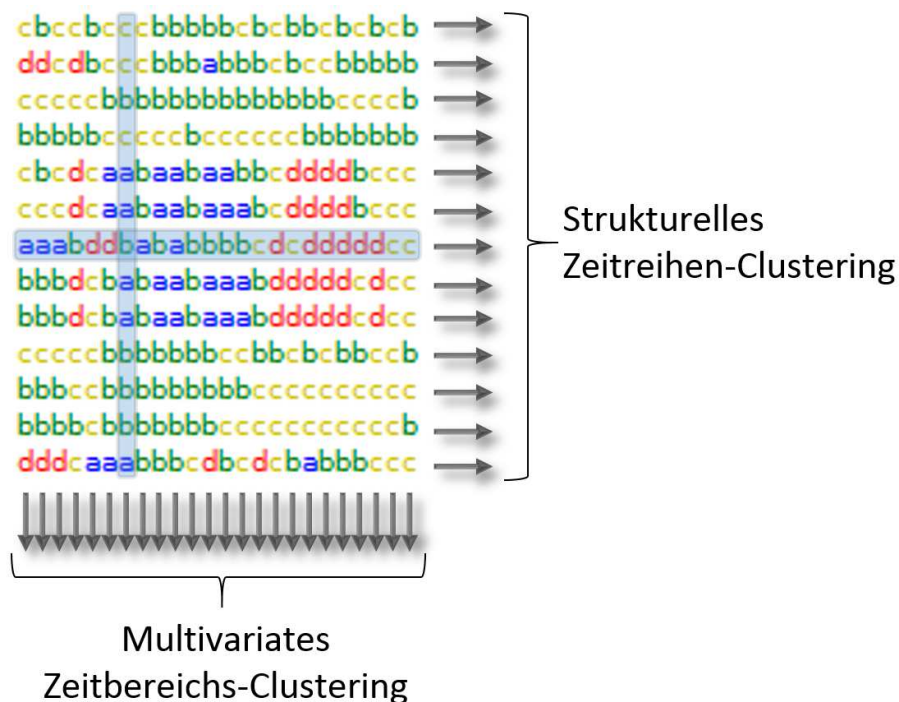


Abbildung 3.29: Erläuterung der SAX-Sequenzbildung als Eingangsdaten für ein hierarchisches Clustering, mit markierter Zeile für Zeitreihen und markierter Spalte für multivariate Zeitbereiche.

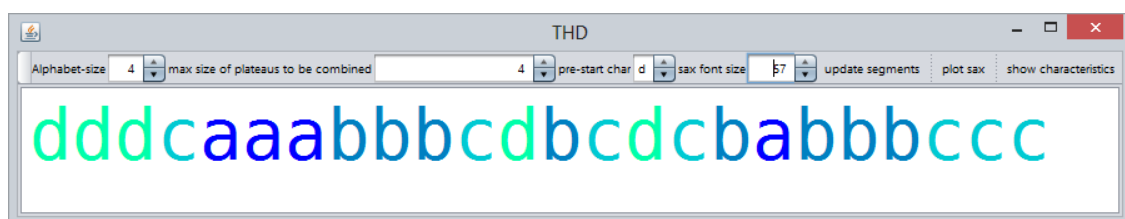
Bei einem *horizontalen Clustering* werden die Sequenzen gruppiert, welche die Zeitreihen repräsentieren und in Abbildung 3.29 als gestapelte Zeilen dargestellt sind. Da alle Sequenzen jedoch gleich lang sind, können ebenso die Sequenzen verglichen werden, welche sich aus Spalten ablesen lassen und einen Zeitbereich als Querschnitt über alle Zeitreihen repräsentieren, weshalb dieses Clustering-Verfahren im Folgenden *vertikales Clustering* genannt wird. Das in Abbildung 3.28 gezeigte Dendrogramm eines vertikalen Clustering zeigt erwartungsgemäß hauptsächlich benachbarte Bereiche („t1“ bis „t23“ stehen für einstündige multivariate Aggregate) gemeinsam gruppiert. Das triviale Ergebnis rührt aus der Tatsache, dass es sich um EDR-Daten eines einzigen

Tages handelt. Da sich die Kurvenverläufe einzelner Tage jedoch ähnlich sind, ist bei einem vertikalen Clustering der Stundenaggregate mehrerer Tage zu erwarten, dass zu ähnlichen Uhrzeiten auch ähnliche Daten vorliegen, so dass z. B. die Zeitbereich zwischen 12:00 Uhr und 13:00 Uhr mehrerer Wochentage gemeinsam eingruppiert würden. Somit eignet sich die Methode auch, um periodisches Verhalten in multivariaten Daten zu detektieren und Saisonalitätseigenschaften zu ermitteln.

### 3.7.3 SAX-Strukturcharakteristik-Editor MetaSAX




Die farbliche Kodierung der SAX-Symbole wie in Abbildung 3.25 ermöglicht bereits eine gute visuelle Erkennung der Struktur von SAX-Sequenzen. Um die Strukturen weitergehend analysierbar zu machen, wurde ein parametrisierbares Strukturanalyseverfahren in Kombination mit einem speziellen Text-Editor (siehe Abbildung 3.30) entwickelt, welches die strukturelle Analyse visualisiert. Das neue Verfahren mit dem Namen *MetaSAX* ermöglicht eine schnelle Überprüfung der Auswirkung von Parameteränderungen, so dass geeignete Parameter auf einfache Weise durch Erprobung ermittelt werden können. Der in Abbildung 3.30 gezeigte Editor weist eine Farbkodierung der Symbole auf, die von der Farbkodierung im Dialog zu SAX-Codierung aus Abs. 3.7 abweicht.

Zwar werden die Symbole hier ebenfalls so gefärbt, dass ihre numerische Ordnung sich in der Farbe widerspiegelt. Hier wurde jedoch ein Farbverlauf von dunkelblau nach grün gewählt, da während der Strukturanalysen auch der Hintergrund der Symbole unterschiedlich eingefärbt werden können soll, ohne die Lesbarkeit zu beeinträchtigen. Die entwickelte Strukturanalyse stellt eine Form der semantischen bzw. formbasierten Zeitreihensegmentierung dar. Dies wird so realisiert, dass die SAX-Sequenz, welche eine Zeitreihe repräsentiert, in Abschnitte aufgeteilt wird, in denen keine strukturellen Veränderungen detektierbar sind.



**Abbildung 3.30:** Text-Editor zur Visualisierung SAX-basierter Strukturanalysen, vor der Segmentierung der SAX-Sequenz.

Hierzu werden die folgenden drei grundlegenden Formelemente zur Beschreibung der strukturellen Charakteristik definiert:

- *Plateau:*  Ein Abschnitt der Sequenz ohne Veränderung, also ohne signifikanten Trend in der Ursprungszeitreihe, wie z. B. die Sequenz „bbbbbb“. Es kann außerdem je nach gewählter SAX-Alphabetgröße zwischen hohen, tiefen und zentralen Plateaus unterschieden werden.
- *Negativtrend:*  Ein Abschnitt der Sequenz mit kontinuierlicher Abnahme der numerischen Werte der Symbole, wie z. B. „dcb“.
- *Positivtrend:*  Ein Sequenzabschnitt mit kontinuierlicher Wertezunahme, wie z. B. „abcd“.

Der Editor besitzt im oberen Bereich eine Kontrollleiste zur Einstellung diverser Parameter:

- *Alphabetgröße:* Sie wird benötigt, um die beiden Symbole mit niedrigstem und höchstem numerischen Wert ermitteln zu können und damit den Wertebereich zu definieren. Ist die Information aus den Daten verfügbar bzw. wird der Editor aus dem SAX-Kodierungsdialog heraus gestartet, wird sie automatisch übernommen.
- *Maximalgröße kombinierbarer Plateaus:* Kleine Plateaus können in manchen Fällen als Teil eines längeren Trends aufgefasst werden. Daher kann über den Parameter die maximale Sequenzlänge der Plateaus festgelegt werden, die in einem Postprocessing zu Trends kombiniert werden können.
- *Startsymbol:* Da es sich bei den untersuchten Sequenzen meist um Teilsequenzen längerer Zeitreihen handelt, kann hierüber ein Symbol als letztes Symbol der vorangehenden Sequenz festgelegt werden. Das ermöglicht beispielsweise Strukturanalysen über Dateigrenzen hinweg. Standardmäßig wird das erste Symbol der Sequenz verwendet.
- *Textgröße:* Hierüber kann die Textgröße der Sequenzdarstellung angepasst werden, so dass Sequenzen unterschiedlicher Länge gut dargestellt werden können.

Die Kontrollleiste enthält außerdem folgende Schaltflächen:

- *Update der Segmente:* Die Segmente werden aufgrund aktuell gewählter Parameter neu berechnet.
- *Plot SAX:* Die SAX-Sequenz wird rücktransformiert in eine Zeitreihe und in der Kurvendarstellung angezeigt, um die ermittelte Strukturcharakteristik überprüfen zu können.
- *Anzeige der Charakteristik:* Die Strukturcharakteristik wird in eine Textdatei geschrieben, welche mit dem Standardeditor des Systems geöffnet wird oder alternativ in eine Datenbank mit Metadaten geschrieben werden kann.

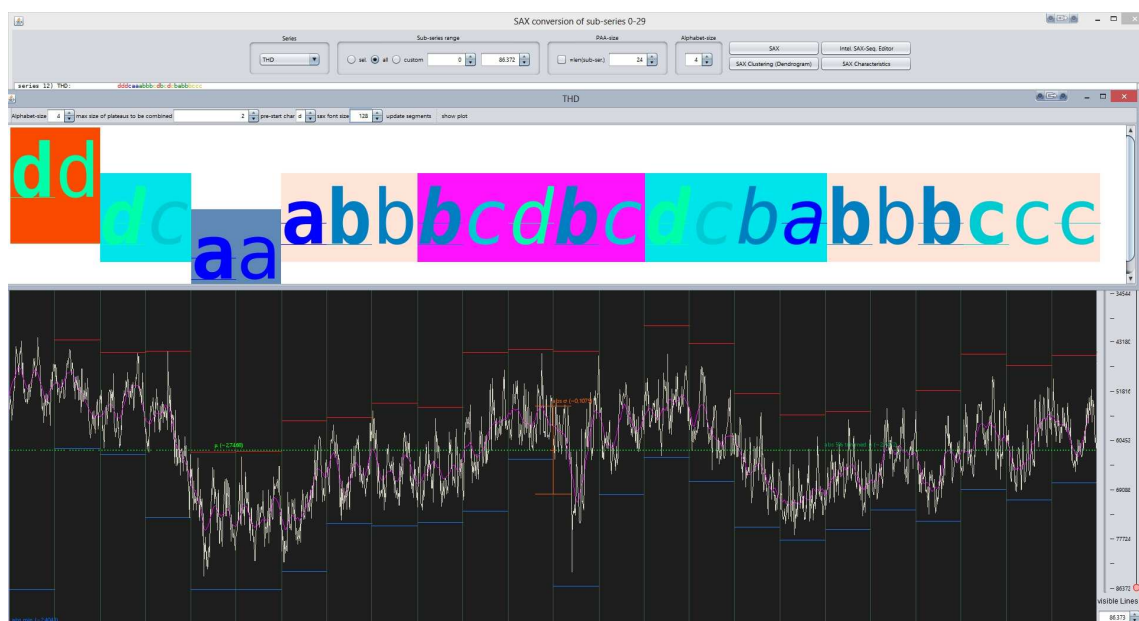
Abbildung 3.31 zeigt ein Beispiel eines Resultates der Strukturanalyse für eine Alphabetgröße von vier und eine Maximalgröße kombinierbarer Plateaus von zwei. Die Textgröße wurde hierfür so eingestellt, dass sich jeder Buchstabe genau über dem einstündigen Abschnitt der Originalzeitreihe befindet, welchen er codiert (siehe Abbildung 3.32). Die Darstellung verwendet über die

Farbcodierung der Symbole hinaus eine farbliche Codierung der ermittelten Segmente, welche den Segmenttyp verdeutlicht, sowie eine Hoch- bzw. Tiefstellung des Textsegmentes, falls es sich um hohe bzw. tiefe Plateaus handelt. Da die Segmente um ein Symbol überlappen können, wird das Startsymbol eines jeden Segmentes außerdem unterstrichen, das Endsymbol durchgestrichen und beide in fetter Schrift dargestellt. Die Textformatierung ist aufgrund der Textgröße und der Bildverkleinerung in Abbildung 3.32 nicht leicht zu erkennen, weshalb Abbildung 3.31 die Zeichensequenz nochmals in kleinerer Schriftgröße zeigt.



dd d d c a a a b b b c d b c d c b a b b b c c c

**Abbildung 3.31:** Textuelle Sequenzdarstellung mit Hervorhebung der Segmente durch Farbkodierung, Hoch- bzw. Tiefstellung und Unter- bzw. Durchstreichung des Textes.



**Abbildung 3.32:** Darstellung der ermittelten Strukturcharakteristik aus einer SAX-Sequenz (oben) und gegenübergestellte Ansicht der Originalkurve (unten).

Die ermittelte Strukturcharakteristik kann in einem einfachen XML-Format als weitere Metadaten exportiert werden bzw. in eine entsprechende Datenbank geschrieben werden. Über die entsprechende Schaltfläche zur Anzeige der Charakteristik wird sie im Standardeditor des Betriebssystems zur Ansicht geöffnet (siehe Code 3.8). Die XML-Beschreibung der Strukturcharakteristik

rakteristik enthält zahlreiche weitere Eigenschaften der Symbolsequenz (z. B. Häufigkeiten einzelner Symbole, Positionen lokaler Maxima und Minima, Zustandsübergänge, enthaltene Standardmuster etc.), die automatisch während der Segmentierung ermittelt werden.

```
<?xml version="1.0" ?>
<data.mining.saxMining.features.SaxWordCharacteristics>
  <preStartChar>d</preStartChar>
  <monotonicallyIncreasing>false</monotonicallyIncreasing>
  <monotonicallyDecreasing>false</monotonicallyDecreasing>
  <nonMonotonicallyIncreasing>false</nonMonotonicallyIncreasing>
  <nonMonotonicallyDecreasing>false</nonMonotonicallyDecreasing>
  <containsOnlyOneChar>false</containsOnlyOneChar>
  <maxIdenticalCharsInARow>4</maxIdenticalCharsInARow>
  <saxCharCnts>
    <int>4</int>
    <int>8</int>
    <int>7</int>
    <int>5</int>
  </saxCharCnts>
  <saxCharFreq>
    <double>0.1666666666666666</double>
    <double>0.3333333333333333</double>
    <double>0.2916666666666667</double>
    <double>0.20833333333333334</double>
  </saxCharFreq>
  <includingCylinder>false</includingCylinder>
  <includingNegativeCylinder>false</includingNegativeCylinder>
  <includingBell>false</includingBell>
  <includingFunnel>false</includingFunnel>
  <nDifferentOccurringChars>4</nDifferentOccurringChars>
  <alphabetSize>4</alphabetSize>
  <localMaxPositions>
    <int>11</int>
    <int>14</int>
  </localMaxPositions>
  <localMinPositions>
    <int>12</int>
    <int>17</int>
  </localMinPositions>
  <nCharsRising>7</nCharsRising>
  <nCharsFalling>6</nCharsFalling>
  <stateTransitions>
    <data.mining.saxMining.features.SaxWordCharacteristics_-StateTrans>
      <pos>-1</pos>
      <type>0</type>
      <outer-class reference="../../../../" />
    </data.mining.saxMining.features.SaxWordCharacteristics_-StateTrans>
    <data.mining.saxMining.features.SaxWordCharacteristics_-StateTrans>
      <pos>2</pos>
      <type>1</type>
      <outer-class reference="../../../../" />
    </data.mining.saxMining.features.SaxWordCharacteristics_-StateTrans>
  </stateTransitions>
</data.mining.saxMining.features.SaxWordCharacteristics>
```

**Code 3.8:** Detailliertes Resultat der Strukturanalyse als XML.

Neben den eingestellten Parametern zur Strukturanalyse wird unter anderem folgendes festgehalten:

- Monotonität
  - monoton
    - steigend
    - fallend
  - nichtmonoton
    - steigend
    - fallend
- ob die Sequenz eine Wiederholung eines einzigen Symbols darstellt
- wie viele identische Symbole maximal in Reihung auftreten
- Anzahl unterschiedlicher vorkommender Symbole
- Gesamtzahl der einzelnen Symbole des Alphabets
- relative Häufigkeit der Symbole in der Sequenz
- ob bestimmte typische Kurvenformen enthalten sind wie
  - Zylinder
  - inverser Zylinder
  - Glocke
  - Trichter
- Position von Extremwerten
  - lokale Maxima
  - lokale Minima
- Anzahl der Zeichen, die einen Anstieg bedeuten
- Anzahl der Zeichen, die einen Abfall bedeuten.

Des Weiteren werden neben den Segmenten selbst auch Zustandsübergänge erfasst. Sie besitzen eine Position und einen Übergangs-Typ, welcher Anfangs- und Endzustand als eine Ganzzahl codiert. Mögliche Übergangstypen sind in Tabelle 3.4 aufgeführt. Dabei steht jeweils ein Buchstabe für einen Zustand (fallend, steigend oder keine Änderung). Übergänge ohne Zustandsänderung werden in einem Postprocessingschritt zusammengefasst, sofern eine Zustandsänderung folgt. Diese Notation eignet sich hervorragend zur Detektion sogenannter *Änderungspunkte* (engl. *Change points*) und damit für eine Änderungserkennung, die prinzipiell ebenso online (für einen laufenden Datenstrom) wie offline möglich ist. Sie zeigt sich außerdem als sehr gut skalierbar. Die Information kann auch zur Erkennung von Anomalitäten beitragen.

**Tabelle 3.4:** Identifizierte Zustandsübergänge bei der SAX-basierten Strukturanalyse.

Name	Wert	Beschreibung
<i>UNDEF</i>	-1	Undefiniert
<i>NN</i>	0	Keine Änderung (bzw. temporärer Übergang)
<i>NF</i>	1	Von <i>NN</i> zu „fallend“
<i>NR</i>	2	Von <i>NN</i> zu „steigend“ (R steht für engl. <i>Rising</i> )
<i>RN</i>	3	Von „steigend“ zu <i>NN</i>
<i>FN</i>	4	Von „fallend“ zu <i>NN</i>
<i>FR</i>	5	Von „fallend“ zu „steigend“
<i>RF</i>	6	Von „steigend“ zu „fallend“
<i>RR</i>	7	Von „steigend“ zu „steigend“ (bzw. temporärer Übergang)
<i>FF</i>	8	Von „fallend“ zu „fallend“ (bzw. temporärer Übergang)

Außerdem werden die einzelnen ermittelten Segmente gespeichert (der entsprechende Abschnitt ist in Code 3.8 nicht zu sehen, siehe Code 3.9). Für jedes Segment werden Start- und Endposition, sowie Anfangs- und Endsymbol gespeichert, um die Angaben flexibel für spätere Suchen nutzen zu können. Die XML-Datei enthält für Tags, welche Java-Objekte repräsentieren, außerdem die vollständige Klassenqualifikation, so dass eine direkte Instanziierung von Java-Objekten aus dem XML-Code möglich ist<sup>30</sup>.

```
<segments>
  <data.mining.saxMining.features.SaxWordSegment>
    <start>0</start>
    <end>2</end>
    <type>2</type>
    <startChar>d</startChar>
    <endChar>d</endChar>
  </data.mining.saxMining.features.SaxWordSegment>
  <data.mining.saxMining.features.SaxWordSegment>
    <start>2</start>
    <end>4</end>
    <type>-3</type>
    <startChar>d</startChar>
    <endChar>a</endChar>
  </data.mining.saxMining.features.SaxWordSegment>
```

**Code 3.9:** XML-Darstellung der Sequenz-Segmente als Resultat der Strukturanalyse.

<sup>30</sup> Das Konzept wurde für alle erzeugten Metadaten im XML-Format konsequent umgesetzt, um die Daten möglichst flexibel zugreifbar zu machen.



### 3.7.4 SAX-Sequenzeditor zur interaktiven Suche unbekannter Muster

Oftmals ist es während einer Zeitreihenanalyse notwendig, Muster in Zeitreihen zu erkennen. Ist das Muster bekannt, kann es als SAX-codierte Sequenz in den vorberechneten Metadaten (siehe Abs. 3.4.2) gesucht werden, die SAX-Codierungen aller Zeitreihen in unterschiedlichen Detailstufen beinhalten<sup>31</sup>. Häufig ist jedoch nicht a priori bekannt, nach welchen Mustern überhaupt gesucht werden soll.

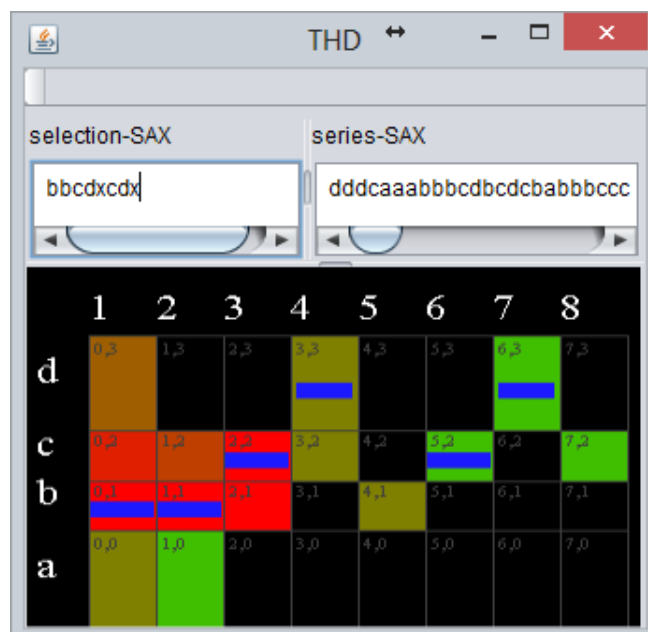
Deshalb wurde eine Methode entwickelt<sup>32</sup>, mit deren Hilfe eine interaktive Mustersuche ermöglicht wird, bei welcher Nutzer in einem Editor schrittweise Muster spezifizieren können und nach jedem Schritt die relative Auftrittshäufigkeit in einer verständlichen visuellen Form rückgemeldet bekommen. So ist es möglich, unbekannte Muster in Zeitreihen aufzuspüren oder teilweise bekannte Muster zu vervollständigen, bzw. den Kontext zu ermitteln, in welchem sie auftreten. Das Verfahren basiert auf der Berechnung und farbcodierten Darstellung der Vorkommenswahrscheinlichkeiten von Mustern, die zum aktuell selektierten Kurvenmuster passen. Es wurde in Form eines interaktiven Sequenzeditors umgesetzt, der in Abbildung 3.33 zu sehen ist.

**Spezifikation des Suchmusters:** Das Suchmuster wird in einer Tabelle im unteren Bereich mithilfe der Maus festgelegt, indem durch Klicken einzelne Zellen selektiert werden. Jede Zeile entspricht dabei einem SAX-Symbol und die Spalten stehen für die Symbolposition innerhalb des Suchmusters. Die Zeilenhöhen sind proportional zur Aufteilung des Wertebereichs während der SAX-Codierung. Sie entsprechen jeweils den Größen der Intervalle, in welche die Zeitbereichsmittelwerte der Ursprungszeitreihe bei der SAX-Codierung fielen. Selektierte Zellen werden durch eine vertikal zentrierte blaue Linie markiert. Die vertikale Zentrierung soll verdeutlichen, dass es sich bei den Symbolen um codierte Mittelwerte über den durch eine Zelle markierten Bereich handelt. In jeder Spalte kann nur eine Zelle gleichzeitig markiert werden, so dass die Selektion eine Kurvenform ergibt, was die Musterselektion leicht verständlich und intuitiv bedienbar macht. Eine Selektion wird durch einen erneuten Klick aufgehoben. Das Suchmuster muss nicht vollständig durch Selektion definiert werden, sondern kann auch Lücken enthalten. Aus der Selektion wird eine SAX-Sequenz abgeleitet, welche an jeder undefinierten Stelle das Symbol  $x$  enthält, analog zu einer *Wildcard*-Suche [252]. Die Suchsequenz wird in einem Textfeld links oben angezeigt.

---

<sup>31</sup> Es ist hierzu notwendig, die gesuchte Sequenz mit denselben SAX-Parametern und Statistiken zu codieren, die während der Erzeugung der Metadaten verwendet wurden. Daher sind die Parameter in den Metadaten enthalten.

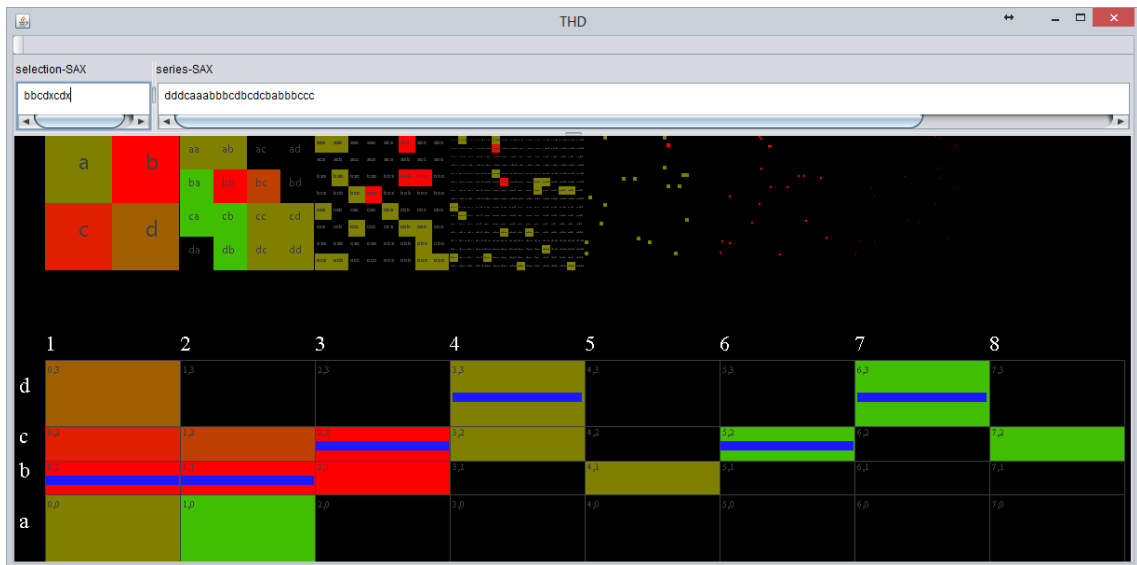
<sup>32</sup> Eine erste Beschreibung des SAX-Editors zur Suche unbekannter Muster wurde vom Autor bereits in [194] publiziert.



**Abbildung 3.33:** Musterselektionstabelle des SAX-Sequenzeditors (unten, blaue Linien markieren die Selektion) und Textfeldern für Suchmuster und Gesamtsequenz (oben), hier mit Suchsequenzlänge acht und Alphabetgröße vier.

Bei jeder Änderung der Selektion werden die Auftrittshäufigkeiten der möglichen Sequenzen neu berechnet und sofort durch Einfärbung der Zellen visuell rückgemeldet. Für Spalten, in denen keine Zelle selektiert ist, werden dabei alle Zellen der Spalte als Symbole möglicher Sequenzvarianten betrachtet, so dass jede nichtselektierte Spalte zusätzliche Varianten zulässt, die ebenfalls berücksichtigt werden. Abbildung 3.34 zeigt eine Ansicht des Editors, in welcher in sechs der acht Spalten eine Zelle selektiert wurde. Spalte fünf und acht sind unselektiert. Die visuelle Rückmeldung zeigt für diese Spalten jeweils eine einzige verbleibende Option durch grüne Zellfärbung, für welche die Suchsequenz in der durchsuchten SAX-Sequenz enthalten ist.

**Rückmeldung der Auftrittshäufigkeiten:** Bei einer Änderung der Selektion werden die Auftrittshäufigkeiten aller zur aktuellen Selektion passenden Suchsequenzen neu berechnet und die Zellen



**Abbildung 3.34:** SAX-Sequenzeditor mit Suchsequenzlänge acht und Alphabetgröße vier, hier mit eingeblendeten Teilsequenzhistogrammen (oben).

entsprechend farblich markiert. Beispielsweise sind in Abbildung 3.34 alle Zellen in Spalte fünf bis auf Zeile *b* schwarz gefärbt. Die schwarze Zelldarstellung bedeutet hier, dass keine der möglichen Teilsequenzen der Länge vier, beginnend an Position eins, an fünfter Stelle ein entsprechendes Folgesymbol aufweist. Die Selektion in den Spalten eins bis vier definiert die Suchsequenz *bbcd*. Sie taucht in der durchsuchten Sequenz *dddcaaabbbcbcdcbabbccc* ausschließlich gefolgt von einem *b* auf (*bbcdb*), und zwar genau einmal, an Position neun (siehe Abbildung 3.35). Die Sequenzen *bbcd*, *bbcd*, *bbcd* sind nicht enthalten.

*bbcdb*  
*dddcaaab**bbcdb**cdcbabbccc*  
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

**Abbildung 3.35:** Teilsequenz *bbcdb* der Suchsequenz (oben) und durchsuchte Sequenz (unten, mit rot markierter Fundstelle an Position 9) entsprechend Abbildung 3.34. Die Auftrittshäufigkeit der Teilsequenz *bbcdb* der Suchsequenz bestimmt die Färbung der Zelle in Zeile *b*, Spalte 5 im Editor.

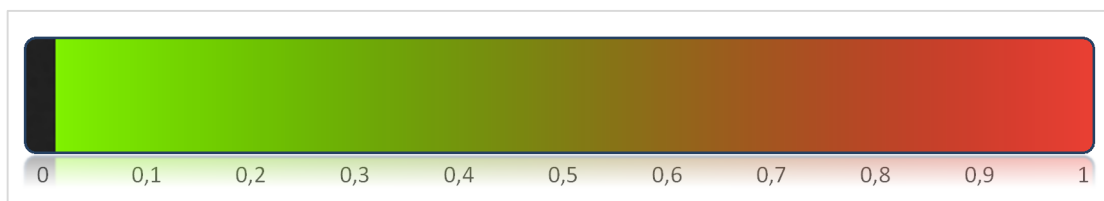
Es wird also jede Zelle in einer Farbe dargestellt, welche die Auftrittshäufigkeit derjenigen Sequenzen codiert, die das zur Zelle gehörige Symbol als Suffix beinhalten. Sind das mehrere Sequenzen, so werden deren Auftrittshäufigkeiten für die Farbcodierung addiert. Die ermittelte

Auftrittshäufigkeit einer Sequenz wird zuvor normalisiert, indem sie durch die Anzahl der Vorkommen der häufigsten Sequenz gleicher Länge geteilt wird (siehe Code 3.10).

```
double p = 0.0;
for (String s : possibilities) {
    int cnt = SAXStrings.countSubStrOccurrences(s, sax);
    double norm = cnt / (double) aProbs.get(col).getnMaxOccurrence();
    p += norm;
}
p /= possibilities.size();
```

**Code 3.10:** Java-Code zur Normalisierung der Auftrittshäufigkeit von Teilsequenzen.

Die Zuordnung der ermittelten normalisierten Auftrittshäufigkeit zu einer Farbe ergibt sich aus Abbildung 3.36. Kommt die entsprechende Sequenz überhaupt nicht vor, ist die normalisierte Häufigkeit also exakt Null, wird schwarz als Zellfarbe verwendet, ansonsten ein Wert auf der Farbskala, die einen linearen Grün-Rot-Gradienten im RGB-Farbraum abbildet.



**Abbildung 3.36:** Farbcodierung der normalisierten Auftrittshäufigkeiten im SAX-Sequenzeditor.

Das hat den Vorteil, dass visuell sofort ersichtlich ist, ob eine Sequenz vorkommt oder nicht. Der Gradient erlaubt hingegen die Einordnung, ob es sich um eine häufige (ein *Motif*) oder selten vorkommende Sequenz (ein *Discord*) handelt. Da die Häufigkeiten mit den Vorkommen der häufigsten Sequenz normalisiert werden, ist die häufigste Sequenz stets in vollem rot dargestellt. Es ist jedoch wichtig zu verstehen, dass es sich hier nicht um relative Häufigkeiten handelt, sondern um ein datenadaptives Verfahren. Gibt es z. B. neben der ermittelten häufigsten Sequenz weitere mit gleicher Häufigkeit, so werden die entsprechenden Zellen alle in vollem rot gefärbt, da die Summe der normalisierten Häufigkeiten nicht wie bei relativen Häufigkeiten Eins ergibt.

Oberhalb jeder Spalte des Sequenzeditors befindet sich eine zusätzliche Visualisierung, bei welcher es sich um farbcodierte Histogramme handelt, sog. *Intelligente Icons* [144]. Diese können wie in Abbildung 3.34 gezeigt, optional im oberen Bereich des Editors angezeigt werden. Sie stellen in einer Tabelle für alle möglichen SAX-Sequenzen (spaltenabhängiger Länge) deren Auftrittshäufigkeiten farblich dar. Sie unterstützen den Nutzer bei Mustersuchen, indem sie einen guten Über-

blick über die Verteilung der Sequenzen bieten. So kann beispielsweise unabhängig von der aktuellen Selektion im Editor abgelesen werden, wie viele Sequenzen der entsprechenden Länge überhaupt in den Daten vorhanden sind. Die SAX-Sequenz-Histogramme und deren neue Modifikationen werden nachfolgend genauer beschrieben.

### 3.7.5 Modifizierte SAX-Sequenz-Histogramme

Wie bereits in Abs. 3.6.7 dargestellt, sind Histogramme wichtige Werkzeuge bei der Zeitreihenanalyse. Das Konzept der Histogramme lässt sich auch auf SAX-Symbolsequenzen übertragen, woraus sich die *Intelligenten Icons* [144] ableiten. Für das Visualisierungsmodul *ViAT* wurden diese implementiert und im Funktionsumfang erweitert.

Die Icons wurden so umgesetzt, dass sie flexibel in verschiedenste Oberflächen integriert und nicht nur als Dateiicons verwendet werden können (z. B. wie in Abbildung 3.34 und Abbildung 3.37). Abweichend von der ursprünglichen Beschreibung der Intelligenten Icons in [144] wurde die Farbcodierung in Analogie zu dem in Abs. 3.7.4 beschriebenen Sequenzeditor umgesetzt (siehe Abbildung 3.36). Die Zellen nicht vorkommender Sequenzen werden daher schwarz gefärbt, wodurch es gegenüber des ursprünglichen Konzepts visuell leichter ersichtlich wird, falls Sequenzen überhaupt nicht vorkommen, als wenn sie denselben Farbton erhalten wie äußerst seltene Sequenzen.

Den Intelligenten Icons wurden einige optionale visuelle Informationen hinzugefügt. So können etwa die Symbolsequenzen der einzelnen Felder zur Veranschaulichung eingeblenet werden (wie in den ersten vier Icons von links in Abbildung 3.34 und in Abbildung 3.37), was hauptsächlich für kürzere Sequenzen sinnvoll erscheint, da die Textdarstellung ansonsten zu klein ist, um noch lesbar zu sein. Als Schrift wurde hierzu *Segoe UI* verwendet, sofern auf dem System verfügbar, da diese auch in kleinster Schriftgröße gut lesbar ist. Des Weiteren kann optional eine Miniatur-Kurvenansicht der Sequenzen in den Feldern angezeigt werden (gelb in Abbildung 3.37), was sich v. a. bei längeren Sequenzen eignet, für welche keine Textdarstellung der Sequenz möglich ist.

Die Icon-Darstellungen vieler Zeitreihen können auch analog zu den Metadaten datenparallel vorberechnet (siehe Abs. 3.4) und als Bilder gespeichert werden, was z. B. eine Verwendung als Dateiicons ohne erneuten Rechenaufwand ermöglicht. Eine Herausforderung stellt die Erzeugung von Dateiicons multivariater Zeitreihen dar. Hier kann z. B. eine Aneinanderreihung der einzelnen Icons für jede Zeitreihe dargestellt werden, was sich jedoch als schlecht skalierbar für eine große Anzahl an Zeitreihen erweist. Alternativ kann der Anwender aufgefordert werden, eine

der Zeitreihen auszuwählen, anhand derer sich die Datensätze im Sinne der konkreten Anwendung gut unterscheiden lassen, so dass nur Icons für die entsprechende Zeitreihe angezeigt werden.



**Abbildung 3.37:** Erweitertes Intelligentes Icon mit textueller und Kurvendarstellung der Sequenzen, hier mit Alphabetgröße vier und Teilsequenzlänge drei.

### 3.7.6 Detektion mit interaktiver Visualisierung von Discords und Motifs

Die automatische Detektion seltener und häufiger Muster in Zeitreihen kann Anwendern bei der Zeitreihenanalyse helfen, indem sie etwa auf automatisch erkannte ungewöhnliche Abschnitte oder wiederkehrende Muster hingewiesen werden. Das ist besonders bei der Exploration sehr großer Zeitreihenbestände sinnvoll oder, falls die Muster in der visuellen Darstellung nicht leicht identifiziert werden können, beispielsweise bei sehr gleichförmigen großen Zeitreihen mit geringen Variationen. Daher wurde eine Detektion von Discords und Motifs für das *FraScaTi*-System

implementiert. Sie wurde so umgesetzt, dass sowohl Detektion als auch Anzeige direkt im Visualisierungsmodul *ViAT* möglich sind.

Für die Implementierung der Detektion wurde teilweise Quellcode der quelloffenen Bibliothek *jMotif* [292] verwendet, welche eine Java-Umsetzung der SAX-basierten Detektion von Motifs gemäß [180] und von Discords gemäß des *HOT SAX*-Verfahrens von Keogh u. a. [143] enthält. Eine Implementierung eines neu entwickelten Verfahrens mit besseren Skalierungseigenschaften wird im nachfolgenden Abs. vorgestellt. Die Detektion wurde in die in Abs. 3.5 vorgestellte einheitliche Schnittstelle für Sequenzdaten integriert, so dass sie prinzipiell unabhängig von konkreten Zeitreihenformaten anwendbar ist. Eine Discord/Motif-Suche in der aktuell in *ViAT* fokussierten Zeitreihe kann über das Kontextmenü gestartet werden, das in der Kurven- und tabellarischen Farbdarstellung verfügbar ist (siehe Abbildung 3.38).

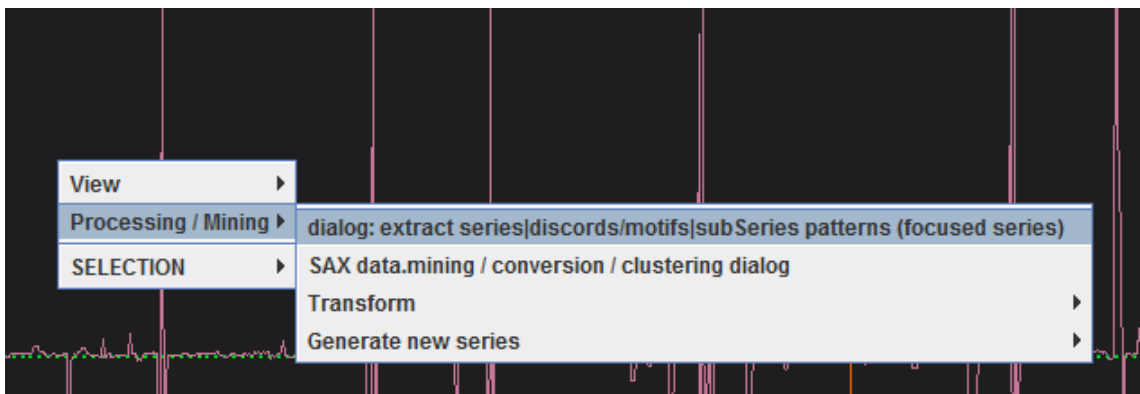


Abbildung 3.38: Suche nach Discords und Motifs über das *ViAT*-Kontextmenü.

Über einen Dialog kann die Mustersuche geeignet parametrisiert werden. Hier kann die Fenstergröße des Fensterverschiebungsalgorithmus, die SAX-Alphabetgröße und die jeweilige Anzahl zu ermittelnder Discords bzw. Motifs festgelegt werden (siehe Abbildung 3.39).

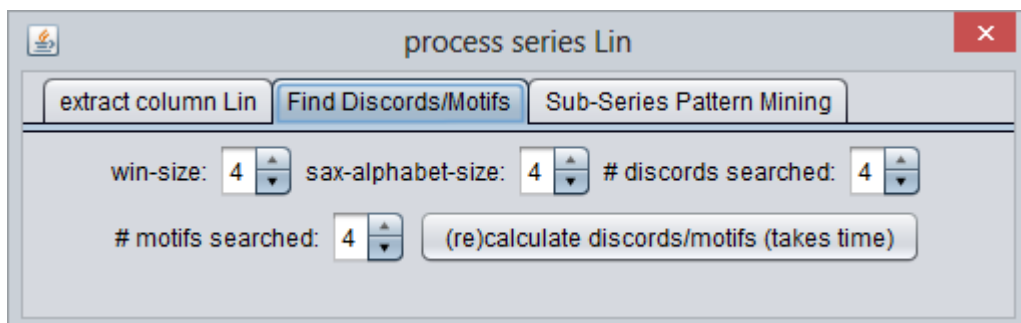


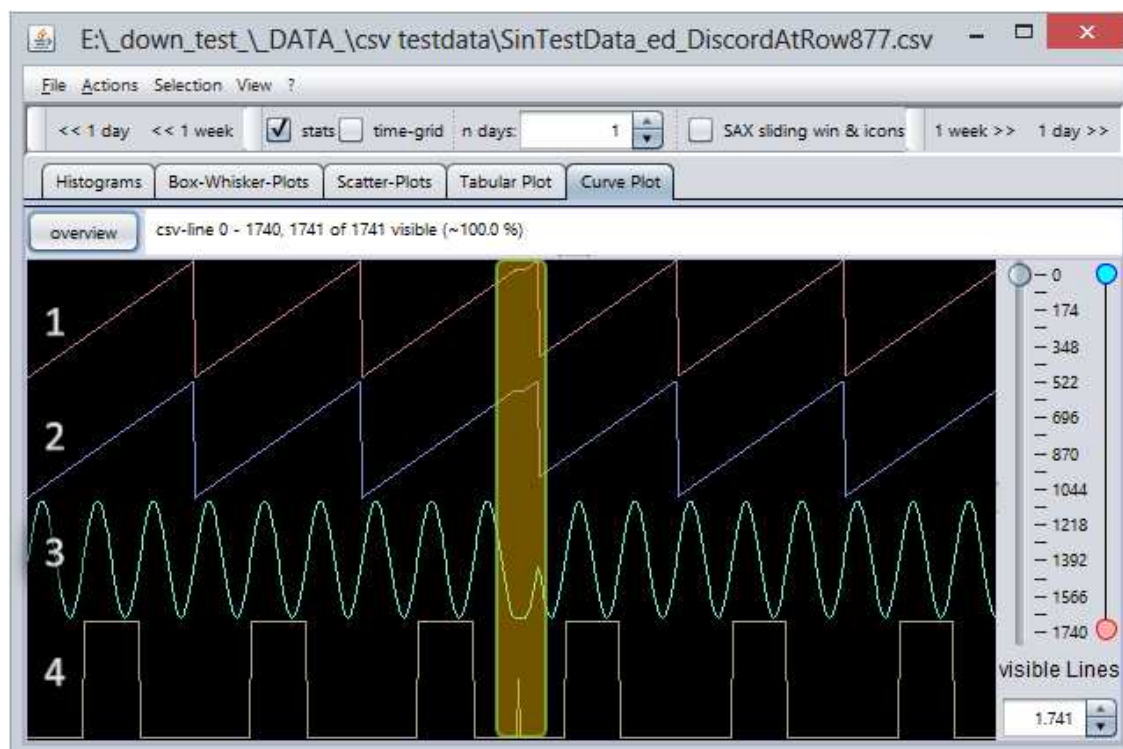
Abbildung 3.39: Dialog zur Einstellung der Suchparameter für eine Detektion von Discords bzw. Motifs.

Zur Illustration wurden diverse Zeitreihendaten generiert, welche gleichförmige Wiederholungen eines Musters (Motifs) darstellen und an einer bekannten Position ein besonderes Muster enthalten (Discord), das nur einmalig vorkommt. Die Testdaten wurden durch Wiederholung unterschiedlicher Schwingungsmuster und für unterschiedliche, willkürliche Wertebereiche gebildet, um die Unabhängigkeit der Detektion vom Wertebereich zu verdeutlichen (siehe Tabelle 3.5).

**Tabelle 3.5:** Eigenschaften der Testdaten zur Detektion von Discords und Motifs.

<i>Index</i>	<i>Grundschiwingung</i>	<i>Wertebereich</i>	<i>Position</i>	<i>hinzugefügte Daten</i>	<i>Länge</i>
1	Sägezahn	[0, 299]	877		1741
2	Sägezahn	[0, 2.9]	877		1741
3	Sinus	[0, 1]	877		1741
4	Rechteck	[0, 1]	877		1741

Abbildung 3.40 zeigt die Kurvendarstellung der Testdaten in ViAT mit einer Markierung des Zeitbereichs, welcher die Discords enthält.

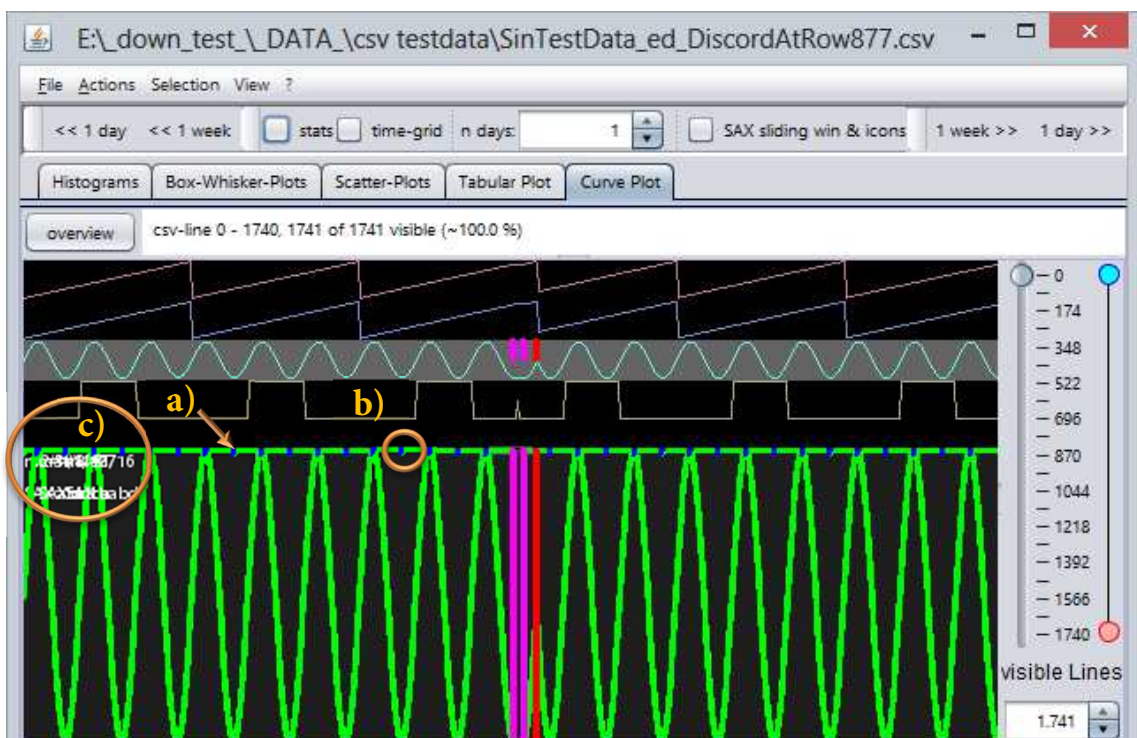


**Abbildung 3.40:** Darstellung der Testdaten zur Detektion von Discords und Motifs in ViAT mit je einem Discord im markierten Bereich.



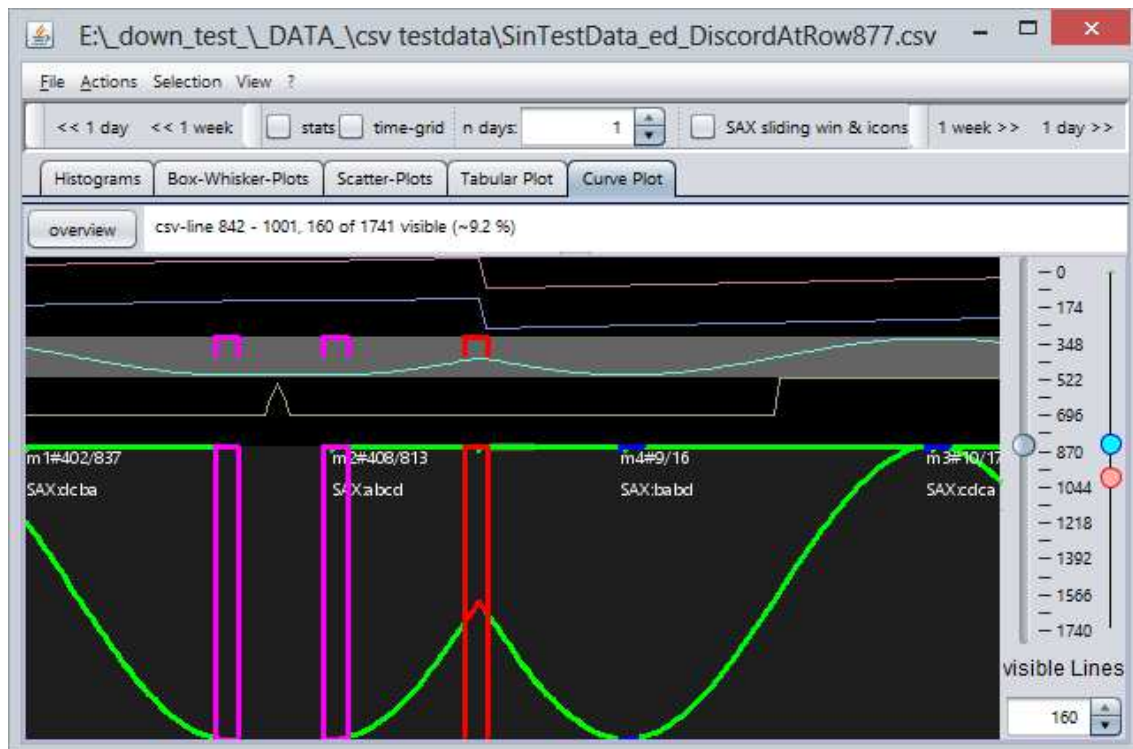
Für die einzelnen Discords wurden unterschiedliche Muster gewählt. In die beiden Sägezahnkurven (1, 2) wurde jeweils ein horizontaler Bereich ohne Werteveränderung eingefügt, in der sinusförmigen Kurve (3) wurde eine Teilsequenz der Grundschwingung wiederholt und in der Rechteck-Kurve (4) wurden zwei in der Grundschwingung vorkommende Teilsequenzen konkateniert. Hervorzuheben ist dabei, dass es sich in keinem der Fälle um Ausreißer bezüglich des Wertebereichs handelt.

Nach Festlegung der Suchparameter (siehe Abbildung 3.39) kann der Fensterverschiebungsalgorithmus über die entsprechende Schaltfläche gestartet werden. Die Zeitreihe wird dabei schrittweise durchlaufen, indem das festgelegte Fenster über der Zeitreihe vom Ende zum Anfang verschoben und ausgewertet wird. Für jede Fensterposition wird entsprechend der Parameter eine (umgekehrte) SAX-Sequenz generiert. Tritt eine solche erstmalig auf, so wird sie zusammen mit der Fensterposition als neuer Eintrag in eine Liste aufgenommen, andernfalls wird ein Zähler für eben diese Sequenz erhöht. Schlussendlich ergibt sich eine Sequenz-Liste, aus der sich durch Sortierung die häufigsten und seltensten Muster entnehmen lassen. Das Suchresultat wird nach Abarbeitung sofort in *ViAT* dargestellt. Das Ergebnis eines Suchlaufs mit den in Abbildung 3.39 gewählten Parametern ist in Abbildung 3.41 zu sehen (fette farbige Hervorhebungen in unterer Kurve). Abbildung 3.42 zeigt eine Vergrößerung der detektierten Discords (pink und rot).



**Abbildung 3.41:** *ViAT*-Darstellung detektierter Discords (rot, pink) und Motifs (grün, blau) in den sinusförmigen Testdaten (mit zusätzlicher Markierung von Elementen in orange: a) Musterübersicht über der Kurve, b) seltenes, blaues Muster, c) überlagertes textuelles Muster in weiß, lesbar in Detailansicht s. Abb. 3.42).

Die in Abbildung 3.41 a) markierte Linie oberhalb der durchsuchten unteren Kurve bietet eine Übersicht über gefundene Muster mit farblicher Codierung der Auftrittshäufigkeit. Gleichzeitig werden die entsprechenden Positionen direkt in der Kurve markiert (hier grün und blau) und textuell dargestellt (Abbildung 3.41 c) und weißer Text in Abbildung 3.42).



**Abbildung 3.42:** Detaillierte Darstellung der Region mit detektierten Discords.

**Darstellung der Motifs:** Analog zur Farbberechnung der SAX-Sequenz-Histogramme (siehe Abs. 3.7.5) wird die Auftrittshäufigkeit jeder Motif-Sequenz durch Division durch die maximale Auftrittshäufigkeit skaliert. Die sich ergebenden Werte werden auf einen linearen Farbgradienten von blau (selten) nach grün (häufig) abgebildet. Ein Großteil der Kurve wurde in Abbildung 3.41 grün eingefärbt, da die zwei häufigsten gefundenen Motifs eine Sinusperiode fast gänzlich abdecken. Nur die „Gipfel“ und „Täler“ weisen eine blaue Färbung auf, da diese Motifs für die gewählte Fenstergröße jeweils nur einmalig in jeder Sinusperiode vorkommen. Da nur das erste Auftreten eines jeden Motifs bei direkter Wiederholung im sichtbaren Bereich der Kurve beschriftet wird und sich Motifs im Beispiel in jeder Sinusperiode mehrfach wiederholen, finden sich ausschließlich Beschriftungen am Beginn der Kurve (weißer Text in Abbildung 3.42 und 3.41). So kann visuell sofort erfasst werden, an welcher Stelle neue Motifs auftreten. Je nach Zoomlevel ist der Text durch Überlagerung unleserlich (, eine Vergrößerung des Bereichs wie in Abbildung 3.42 enthüllt jedoch die Detailangaben einzelner Motifs. Eine horizontale Linie entlang des oberen Randes der Kurvendetaildarstellung, welche abschnittsweise dieselbe Farbe wie das entspre-

chende Motif erhält (siehe Markierung *a*) in Abbildung 3.41), dient dazu, die Motifs visuell leichter erfassen zu können, da sie sich häufig gegenseitig überlappen. Die Startposition eines Motifs, der sich vom zuletzt gefundenen Motif unterscheidet, wird außerdem markiert durch eine Verdickung der Linie (siehe Markierung *b*) in Abbildung 3.41). Die vier ermittelten Motifs (*m1* – *m4*) werden zusammen mit der jeweiligen SAX-Sequenz und der Anzahl der Fundstellen in Tabelle 3.6 aufgeführt.

**Tabelle 3.6:** Detektierte Motifs in den sinusartigen Testdaten.

<i>Bezeichnung</i>	<i>SAX-Sequenz</i>	<i>Auftrittshäufigkeit</i>
<i>m1</i>	abcd	837
<i>m2</i>	dcba	813
<i>m3</i>	cdca	17
<i>m4</i>	babd	16

Es zeigt sich klar, dass *m1* (ein positiver, bzw. in der Ursprungszeitreihe aufgrund der Richtungs-umkehrung ein negativer Trend) und *m2* am häufigsten auftreten – und zwar jeweils in den fallenden (*m1*) und ansteigenden (*m2*) Abschnitten einer Sinusperiode. Wesentlich seltener tritt *m3* auf, welches einen „Gipfel“ (z. B. bei Markierung *b*) der Sinuskurve repräsentiert und somit entsprechend der Anzahl der „Gipfel“ in Abbildung 3.41 17-mal vorkommt. Ähnlich verhält es sich mit *m4*, welches einem „Tal“ der Periode entspricht und nur 16-mal auftritt, da die Kurve mit einem „Gipfel“ beginnt und das „Tal“ am Ende der Kurve nicht vollständig ist, so dass die entsprechende Sequenz dort nicht gebildet werden kann.

**Darstellung der Discords:** Die detektierten Discords haben einen hohen semantischen Wert und werden deshalb durch Umrahmung besonders hervorgehoben. Die Rahmenfarbe codiert, wie bereits erwähnt, die Distanz zum nächsten Nachbar, welche unter Verwendung des MINDIST-Distanzmaßes [181] berechnet wird. Sie wird zur Skalierung durch die maximale Distanz zwischen allen gefundenen Discords geteilt. Die resultierende Farbe sollte stets einen Rotton beinhalten, um die Auffälligkeit der Discords zu erhöhen und sie von seltenen Motifs unterscheiden zu können. Deshalb wurde hier kein einfacher Gradient von blau nach rot gewählt, sondern die Distanz codiert lediglich den Blauanteil der RGB-Farbe, während der Rotanteil stets maximal gewählt wird. So ergibt sich eine rote Färbung bei maximalen Distanzen und eine pinke Färbung bei geringeren Distanzen. In der Detaildarstellung der Kurve werden die Discord-Bereiche komplett eingerahmt, während in der gestapelten Kurvenübersicht der Rahmen nur die halbe Kurvenhöhe einnimmt, um die Kurve nicht zu stark zu überdecken. Die in der sinusförmigen Kurve detektierten Discords werden in Tabelle 3.7 aufgeführt.

**Tabelle 3.7:** Detektierte Discords in den sinusartigen Testdaten.

<i>Bezeichnung</i>	<i>SAX-Sequenz</i>	<i>Distanz zu nächstem Nachbar</i>	<i>Fundstelle</i>
<i>d1</i>	abdb	0.06211257	914
<i>d2</i>	dbbb	0.00009866	873
<i>d3</i>	bbbd	0.00009866	891

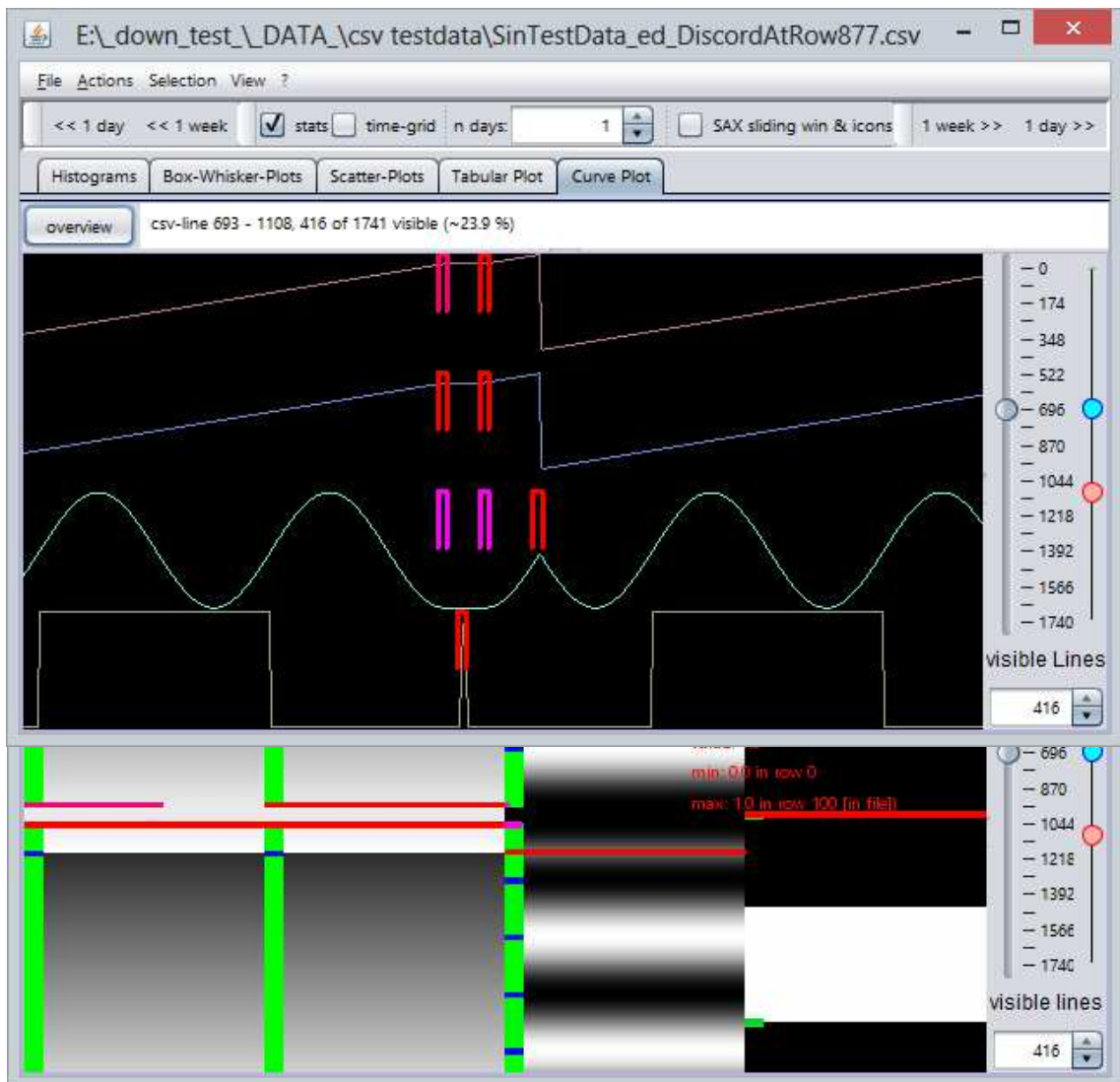
**Darstellung von Discords und Motifs in der tabellarischen Farbfelddarstellung:** Die Ergebnisse einer Discord/Motif-Suche werden analog auch in der tabellarischen Farbfeldansicht dargestellt, wobei die Distanzen der Discords zu ihren nächsten Nachbarn hier zusätzlich durch die Breite der Markierung codiert werden. Discords mit maximaler Distanz zu nächsten Nachbarn werden somit über die Gesamtbreite der Spalte gezeichnet, welche die entsprechende Zeitreihe repräsentiert, Discords mit geringerer Distanz entsprechend kleiner. Die Darstellung der Motifs erfolgt auf konstanter Breite. Zur Illustration wurde in allen Zeitreihen der Testdaten nach Motifs und Discords gesucht. Die Resultate werden in Abbildung 3.43 sowohl für die Kurven- als auch für die tabellarische Farbfelddarstellung präsentiert. Die Fundstellen der Discords weichen dabei geringfügig von der Position der zusätzlich in den Testdaten eingefügten Bereiche ab, da letztere teilweise auch an weiteren Stellen vorkommen und somit nur abschnittsweise Discords darstellen.

### 3.7.7 Algorithmus zur Schnell-Detektion potentieller Discords und Motifs

Die automatische Suche von Discords bzw. Motifs gemäß [180],[143] in größeren Datensätzen kann für manche Suchparameter längere Zeit in Anspruch nehmen. Außerdem ist die Parametrisierung der Suche nicht sonderlich flexibel, da hier eine Baumstruktur namens *SAX-Trie* (siehe [143]) verwendet wird, bei welcher Alphabetgröße und PAA-Größe bei der Bildung der Teilsequenzen stets identisch sind. Das verhindert einige potentiell nützliche Analyseanwendungen, vor allem in einer interaktiven Explorationsumgebung.

Daher wurde eine alternative Methode zur Detektion von Discords und Motifs entwickelt, die eine entsprechend flexible Parametrisierung erlaubt und außerdem eine geringere Komplexität aufweist und somit Suchen in großen Datensätzen bei akzeptablen Laufzeiten ermöglicht. Die neue Methode basiert wie die *HOTSAX*-Methode [180] auf der Extraktion und Sammlung aller Teilsequenzen einer bestimmten Länge  $l$ . Dabei können jedoch beliebig wählbare Alphabet-Größen  $a$  und PAA-Größen  $p$  verwendet werden, so dass bei der Diskretisierung der Zeitreihen die zeitliche Kardinalität unabhängig von der Kardinalität des Wertebereichs, also der Anzahl von

Symbolen, gewählt werden kann. Die Zeitreihe wird auch hier mit einem Fensterverschiebungs-  
algorithmus durchlaufen, indem ein Fenster der Länge  $l$  vom Ende bis zum Beginn der Zeitreihe  
verschoben wird und für jede Position eine SAX-Sequenz abgeleitet wird.



**Abbildung 3.43:** Darstellung der in allen Test-Zeitreihen detektierten Discords in der Kurvendarstellung (oben) und in der tabellarischen Farbfeldarstellung (unten, zusammen mit den Motifs).

Die dabei vorkommenden Teilsequenzen werden als neue Elemente einer Liste aufgenommen, falls diese dort noch nicht vorhanden sind und die Position der neuen Fundstelle außerdem mindestens um einen festzulegenden konstanten Wert abweicht. Niedrige Werte von Eins oder zwei bis hin zur Fensterlänge (wie in [180] zur Vermeidung von „*trivial matches*“ bei der Discordsuche) erscheinen für die Mindestabweichung sinnvoll. Der Ausschluss nahe beieinanderliegender

Fundstellen derselben Sequenz dient dazu, zu vermeiden, dass durch zu geringe Fensterverschiebung dieselbe SAX-Sequenz gebildet wird. Die Positionen aller Fundstellen werden in entsprechenden parallelen Listen festgehalten. Die Liste der Sequenzen wird anschließend mit einem Quicksort-Algorithmus [115] nach aufsteigender Auftrittshäufigkeit sortiert. Hierfür wurde die Implementierung des Quicksort-Algorithmus so modifiziert, dass sie eine Sortierung paralleler Listen erlaubt, so dass eine sequenzielle Sortierung mehrerer Listen vermieden werden kann. Potentielle Discords sind nach Sortierung im unteren, potentielle Motifs im oberen Teil der Liste zu finden. Dieser vergleichsweise einfache Algorithmus verzichtet im Vergleich zu [143] auf die aufwendige Ähnlichkeitsbewertung der einzelnen Sequenzen, so dass bei der Discord-Detektion keine Distanzen zum jeweils nächsten Nachbar genutzt werden. Für eine interaktive Exploration ist es bedeutsamer, schnell potentielle Discords zu identifizieren, als durch aufwendige Berechnungen die Discords mit maximalem Abstand zu allen weiteren Discords zu finden. Außerdem ergeben sich bei der Evaluation des Verfahrens zahlreiche weitere Vorteile (siehe Abs. 4.3). Daher wurde sowohl das Originalverfahren *HOTSAX* als auch der neue Algorithmus *SortList* in *ViAT* integriert, so dass im jeweiligen Anwendungsfall entschieden werden kann, welche Methode eingesetzt werden soll.

### 3.8 Angepasste Volltextsuche für Metadaten

Um die Metadaten durchsuchen zu können, die datenparallel auf dem Hadoop-Cluster für große Mengen an Zeitreihendaten vorberechnet werden können, wurde eine indexbasierte Suche entwickelt. Hierzu wurde eine Indizierung für die XML-Metadaten-Dateien in HDFS entwickelt, die mit der quelloffenen Programmbibliothek *Apache Lucene* [9], [95] zusammenarbeitet und so Volltextsuchen ermöglicht. Da Lucene selbst sehr gut mit großen Datenmengen skaliert und mit *Apache Solr* [11] auch eine verteilte Indizierung möglich ist, sollte das Verfahren gut mit einem weiteren Zuwachs an Daten zurechtkommen, sofern eine ausreichende Anzahl an Datenknoten bereitsteht. Lucene kann u. a. die Formate XML und CSV indizieren, die sehr häufig zur Speicherung von Zeitreihendaten verwendet werden (z. B. auch für die EDR-Daten, siehe Abschn. 5.1.2 und 5.1.3).

Um eine Indizierung von HDFS-Dateien zu ermöglichen, wird ein sog. *RAM-Ordner* erzeugt, welcher einen *Index-Ordner* in HDFS im lokalen Speicher eines Client-Rechners spiegelt. Damit ist es möglich, auch auf entfernten Clients, die einen Zugang zum Hadoop-Cluster besitzen, lokal einen Index der HDFS-Metadaten-Dateien zu erzeugen und im lokalen Dateisystem zu speichern und dort zu durchsuchen, sowie diesen in HDFS zu speichern und datenparallel zu durchsuchen (mittels *Apache Solr* oder auf Basis von MapReduce). Soll nun eine XML-Metadaten-Datei indiziert werden, die aus einer EDR-Merkmaldatei gebildet wurde, wird ein Index für jedes Feld eines jeden Merkmals (siehe Abs. 3.4.2) erzeugt. Code 3.11 zeigt einen Ausschnitt der Konsolenausgabe

während der Indexerstellung. Das Datenvolumen wird dabei für textbasierte Dateien auf etwa 20 % bis 30 % der Ausgangsdaten reduziert [8]. Prinzipiell kann der Index auch auf einzelne Felder begrenzt werden, je nachdem wonach später gesucht werden soll.

```
Indexing file: /user/felix.bach/MetaData/meta.xml
to index dir: /user/felix.bach/indexRootEdrFeatureMetaData

doc: adding new Field: PhaseToPPS.name=PhaseToPPS
doc: adding new Field: PhaseToPPS.min=30.0
doc: adding new Field: PhaseToPPS.minPos=0
doc: adding new Field: PhaseToPPS.max=51.0
doc: adding new Field: PhaseToPPS.maxPos=0
doc: adding new Field: PhaseToPPS.mean=49.99362069165132
doc: adding new Field: PhaseToPPS.trimmedMean=0.0
doc: adding new Field: PhaseToPPS.stdDev=0.026113613220742796
doc: adding new Field: PhaseToPPS.med=50.0
doc: adding new Field: PhaseToPPS.paaSizeFull=0
doc: adding new Field: PhaseToPPS.paaSizeMedium=0
doc: adding new Field: PhaseToPPS.paaSizeSmall=0
doc: adding new Field: PhaseToPPS.alphabetSizeFull=0
doc: adding new Field: PhaseToPPS.alphabetSizeSmall=0
doc: adding new Field: PhaseToPPS.saxSmall=4c1b3c7b1c1b2c5b
```

**Code 3.11:** Ausschnitt der Konsolenausgabe bei Durchführung einer Indizierung einer Metadatenfile für EDR-Merkmalen.

## 3.9 Zusammenfassung

Zur Umsetzung des in Kapitel 2 vorgestellten Konzepts wurden in diesem Kapitel zunächst die Voraussetzungen für eine skalierbare verteilte Verarbeitung von Zeitreihendaten untersucht. Anschließend wurde ein Prototyp der datenparallelen Verarbeitung von Zeitreihendaten in Hadoop entwickelt, evaluiert und als tauglich für skalierbare Auswertungen bewertet. Neue Verfahren zur Interaktion mit Hadoop auf verschiedenen Abstraktionsebenen, wie z.B. ein auf Skalierbarkeit mit großen Daten und Dateimengen optimierter HDFS-Dateiexplorer mit Möglichkeiten, Daten direkt zu visualisieren und zu analysieren, wurden vorgestellt und als Java-Software erfolgreich implementiert und erprobt. Dadurch wird eine effektive Organisation und Auswertung speziell für große, teilweise unstrukturierte Daten in Hadoop über eine effiziente und benutzerfreundliche Schnittstelle ermöglicht.

Eine Hadoop-basierte datenparallele und daher sehr effiziente Erzeugung von Zeitreihen-Metadaten wurde vorgestellt. Die extrahierten Metadaten enthalten relevante statistische Merkmale

der Zeitreihen und eine dimensionsreduzierte symbolische Repräsentation (SAX) und erlauben zahlreiche neue explorative Auswerteverfahren wie unscharfe inhaltsbasierte Suchen, die Extraktion abstrakter struktureller Merkmale, ein interaktives Teilsequenz-Mining, welches die Suche nach zuvor unbekanntem häufigen bzw. seltenen Mustern erlaubt, eine automatische Extraktion von Discords und Motifs und effiziente Volltext-Suchen, deren Ergebnisse sich auf die Originaldaten übertragen lassen.

Für einen einheitlichen Zugriff auf Zeitreihen in verschiedenen Datenformaten wurde eine universelle Schnittstelle entwickelt. Diese wird in den einzelnen vorgestellten Verfahren zur skalierbaren explorativen Analyse großer Zeitreihendaten genutzt, die unterschiedliche neue Sichten auf Zeitreihendaten bieten.

Die interaktiven Explorationsverfahren wie eine multivariate Kurvendarstellung, eine tabellarische Visualisierung, eine Benutzerkooperation durch Zeitreihen-Kommentare, Box-Whisker- und Scatter-Plots für Zeitreihen sowie neue Zeitreihenhistogramme mit farblicher Verlaufsdarstellung, wurden als Software design, implementiert und in ein Visualisierungsmodul namens *ViAT* integriert, in welchem eine flexible interaktive explorative Analyse sowohl von lokalen Zeitreihendaten als auch von Zeitreihendaten direkt auf dem Hadoop-Cluster möglich ist. Des Weiteren wurden neue und modifizierte interaktive SAX-basierte Analysewerkzeuge eingeführt, wie eine parametrisierbare textbasierte SAX-Codierung, ein Sequenz-Clustering mit Dendrogrammdarstellung, ein neuer Struktureditor zur Extraktion abstrakter Struktureigenschaften, ein SAX-Sequenz-Editor, der innovative interaktive Mustersuchen ermöglicht, modifizierte Sequenzhistogramme, eine interaktive Detektion von Discords und Motifs sowie ein modifizierter Detektionsalgorithmus hierzu.

Die hier vorgestellten Verfahren werden in Kapitel 4 hinsichtlich ihrer Funktions- und Leistungsfähigkeit untersucht.



## 4 Evaluation der Verfahren im Verbund

Der Kapitel 4 befasst sich mit der experimentellen Evaluation der in den vorigen Abschnitten vorgestellten Methoden und Softwaremodulen. Zunächst wird die Integration der einzelnen Module zu einem skalierbaren Gesamtsystem präsentiert. Anhand einiger Experimente werden typische Abläufe bei der Exploration großer Zeitreihendaten anhand von Testdaten bzw. mit unterschiedlichen Parametern evaluiert und bewertet und dabei die Funktions- und Leistungsfähigkeit einzelner Komponenten untersucht. Die Ergebnisse werden dargestellt und diskutiert.

### 4.1 Zusammenspiel der Komponenten als skalierbares Gesamtsystem

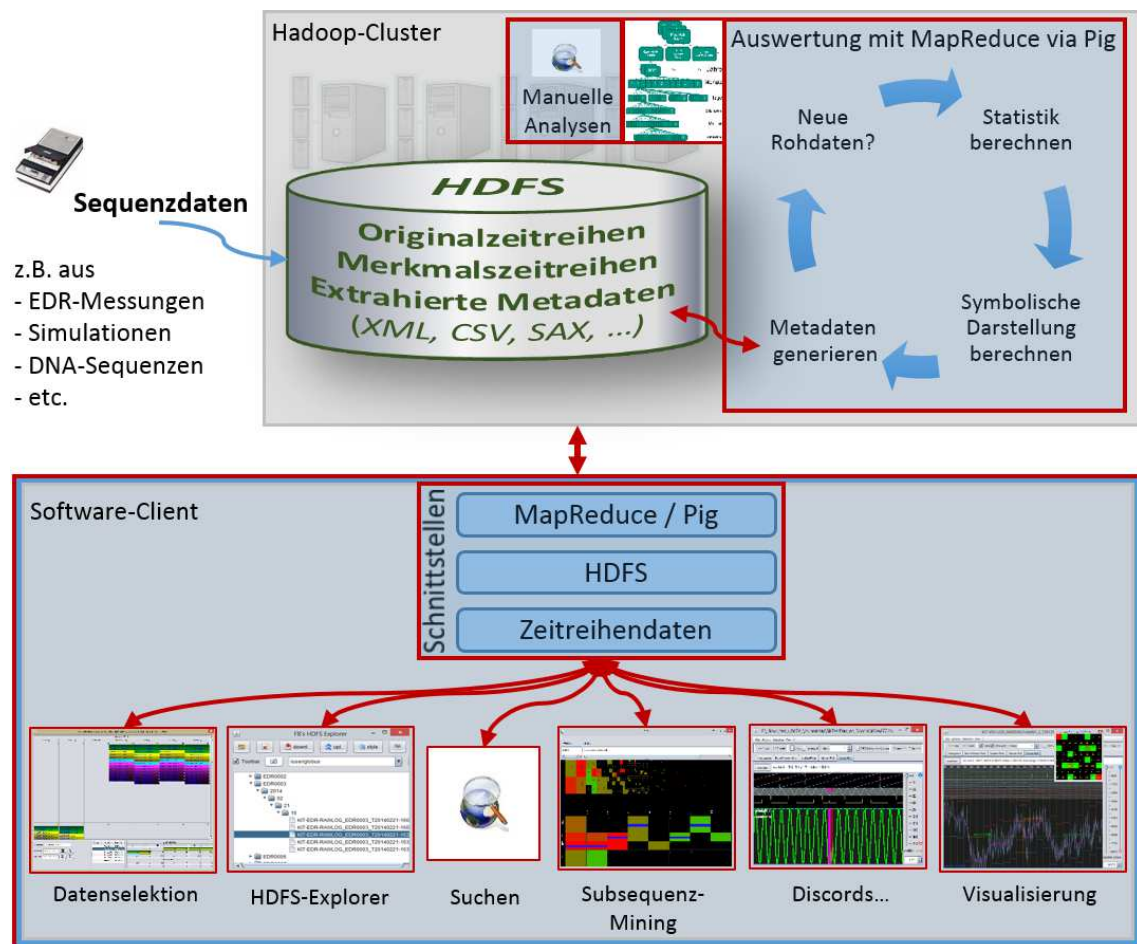
Die in Kapitel 3 vorgestellten Module und Komponenten wurden als Java-Software design und umgesetzt und bilden als Verbund das Gesamtsystem *FraScaTi*, das Anwendern die Exploration und Analyse großer multivariater Zeitreihendaten ermöglichen und gut mit der Datenmenge skalieren soll. Eine schematische Darstellung der Systemumsetzung mit rot markierten Eigenanteilen ist in Abbildung 4.1 zu sehen. Die einzelnen Module wurden auf ein effektives und effizientes Zusammenspiel hin optimiert, was u. a. durch Nutzung gemeinsamer Datenaustauschformate und -Schnittstellen sowie durch Synchronisationsmechanismen der Darstellungsmodule untereinander erreicht wurde. Resultate abstrakter Analysen, Nutzerkommentare, Bereichsselektionen, fehlende Werte und Zeitstempelfehler werden in unterschiedlichen Ansichten, z. B. in der Kurven- und der tabellarischen Farbfelddarstellung, synchronisiert dargestellt, was einen Wechsel zwischen den Ansichten während der Durchführung von Analysen erlaubt.

Nahezu alle Module können einzeln oder im Verbund gestartet werden und über die Zeitreihenschnittstelle sowohl auf lokale als auch auf verteilte in HDFS vorliegende Zeitreihendaten unterschiedlicher Formate zugreifen. Es ist außerdem möglich, aus einem Modul heraus weitere Module aufzurufen. Beispielsweise kann das Visualisierungsmodul aus dem HDFS-Dateiexplorer und aus dem Datenselektionswerkzeug heraus aufgerufen werden. Es ist jedoch ebenso möglich, im Visualisierungsmodul einen HDFS-Dateiexplorer zu instanziiieren, um darzustellende Daten auszuwählen oder Zeitreihensegmente, die in der Visualisierung selektiert wurden, datenparallel auszuwerten etc.

Die datenparallele Auswertung und Erzeugung von Metadaten (Kasten rechts oben in Abbildung 4.1) kann aus dem Datenselektionswerkzeug heraus manuell für ausgewählte Dateien angestoßen

Abs. 4.1 – Zusammenspiel der Komponenten als skalierbares Gesamtsystem

werden oder in regelmäßigen Abständen automatisch durchgeführt werden. Im ersten Fall können beispielsweise die Metadaten für bestimmte Datensätze erneut berechnet werden, falls beispielsweise neue Merkmale oder Statistiken aufgenommen werden sollen oder Fehler in den Metadaten gefunden wurden. Die automatische Auswertung leitet Metadaten von allen Datensätzen in einem bestimmten Ordner ab, der neue, noch nicht analysierte Daten enthält. Die berechneten Metadaten, v. a. die SAX-Repräsentationen der Zeitreihen bilden die Grundlage für nachfolgende Analysen, z. B. der Strukturanalyse, des Subsequenz-Mining und der Suchindexerstellung zur Ermöglichung von Volltextsuchen in der symbolischen Darstellung und weiteren Merkmalen, welche z.B. in der Client-Anwendung (unterer Kasten in Abbildung 4.1) durchführbar sind.



**Abbildung 4.1:** Schematische Darstellung des fertigen Gesamtsystems für skalierbare Zeitreihenanalysen (eigene Arbeiten rot markiert).

## 4.2 Subsequenz-Mining mit SAX-Sequenzeditor

Der in Abs. 3.7.4 eingeführte *SAX-Sequenzeditor* zur Suche bislang unbekannter Muster („*Subsequenz-Mining*“ in Abbildung 4.1) stellt eine neuartige interaktive Methode des Zeitreihen-Mining dar, die es ermöglicht, Zeitreihenmuster in einem Editor teilweise zu definieren und eine sofortige Rückmeldung aller verbleibenden möglichen Muster und ihrer relativen Aufttrittshäufigkeiten zu erhalten. Somit kann sowohl das Auftreten bekannter Muster schnell überprüft werden, als auch auf einfachem Wege neue häufige und seltene Muster sowie das Nichtauftreten von Mustern identifiziert werden. Die Suche wird auf symbolischen SAX-Sequenzen durchgeführt. Das einzige ähnliche Verfahren, das dem Autor aus der Literatur bekannt ist, wurde von Jessica Lin unter dem Namen *VizTree* [179] veröffentlicht (s. Abs. A.2.4). Im Folgenden werden die beiden interaktiven Verfahren gegenübergestellt und jeweilige Stärken und Schwächen identifiziert.

Beide Ansätze haben gemeinsam, dass die Aufttrittshäufigkeit relativ kurzer Teilsequenzen mithilfe von *Suffix-Bäumen* [320] bestimmt und visuell dargestellt wird, um Nutzern visuell einen Eindruck von der Zusammensetzung der Zeitreihe zu vermitteln. In *VizTree* wird zur Darstellung eine Baumstruktur genutzt, während der *SAX-Sequenzeditor* eine spezielle tabellarische Ansicht der diskretisierten Zeitreihe besitzt. Die horizontale Baumstruktur in *VizTree*, mit Codierung der Aufttrittshäufigkeiten über die Strichstärke der Zweige, hat den Vorteil, dass damit die Struktur der gesamten Zeitreihe auf einen Blick erfasst werden kann, während die tabellarische Darstellung im *SAX-Sequenzeditor* zuerst Interaktion erfordert, nämlich die Selektion mindestens eines Teilsymbols, bevor die farbliche Codierung etwas über die enthaltenen Teilsequenzen verrät. Solange keine Zelle selektiert wurde, werden nur die Aufttrittshäufigkeiten der einzelnen Symbole und der monotonen Aneinanderreihung derselben dargestellt. Dies wurde im *SAX-Sequenzeditor* so kompensiert, dass über jeder Spalte ein SAX-Sequenz-Histogramm (siehe Abs. 3.7.5) angezeigt wird, das die statistische Verteilung der SAX-Sequenzen darstellt, welche der Sequenzlänge der jeweiligen Spalte entsprechen.

Umgekehrt ist jedoch in *VizTree* Interaktion erforderlich, um die Kurvendarstellung der Zeitreihen, die zu einem ausgewählten Ast des Baumes zugehörig sind, in einem separaten Fenster darzustellen. Eine solche Kurvendarstellung ist im *SAX-Sequenzeditor* bereits integriert, und zwar in einer Weise, welche den Zusammenhang zwischen Kurve und SAX-Sequenz visuell verdeutlicht. Hierzu werden die einzelnen Editor-Zellen in einer Größe dargestellt, welche den SAX-Umbruchpunkten für die gewählten Parameter entspricht. Außerdem werden selektierte Zellen mit einer horizontalen blauen Linie markiert, was verdeutlichen soll, dass es sich bei einer ausgewählten Zelle um aggregierte Mittelwerte aus Abschnitten der Originalzeitreihe handelt, welche innerhalb der Zellgrenzen liegen.

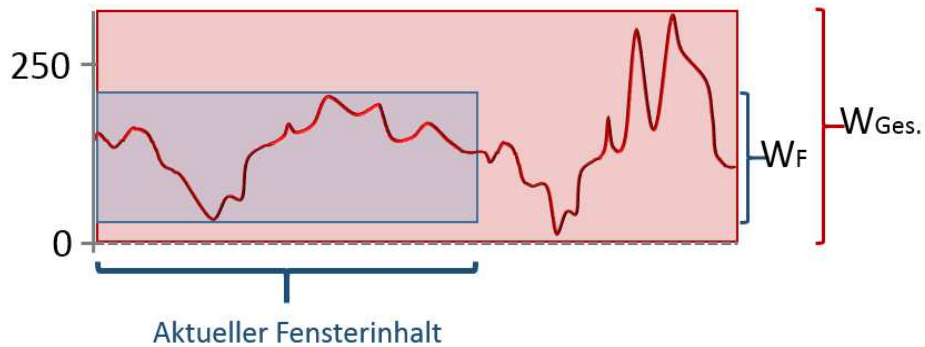
Ein klarer Nachteil der Baumdarstellung in *VizTree* ist jedoch die mit der Teilsequenzlänge und der Alphabetgröße zunehmende Anzahl der Äste. Zwar ist der benötigte Darstellungsraum für beide Ansätze unabhängig von der Länge der untersuchten Zeitreihe. Bei Verwendung eines typischen Alphabets mit vier Symbolen und einer Teilsequenzlänge von vier Zeichen müssen aber bereits  $4^4 = 256$  Äste dargestellt werden, was einen minimalen Anzeigebereich von 512 Pixeln erfordert, wenn davon ausgegangen wird, dass einer Linie mit einem Pixel Breite jeweils eine Lücke von ebenfalls einem Pixel Breite folgt – was bei schrägen Linien jedoch nicht ausreicht. Bereits bei einer Erhöhung der Sequenzlänge auf fünf ergeben sich 1024 Äste und damit mindestens 2048 benötigte Pixel, was ein aktuell typisches Display bereits vollständig ausfüllt. Die kompaktere Anzeige im *Sax-Sequenzeditor* benötigt hierzu vergleichsweise wenig Platz – nämlich eine Tabelle mit vier Reihen und fünf Spalten. Die Anzahl benötigter Reihen und Spalten ist dabei proportional zu Alphabetgröße (Reihen) und Teilsequenzlänge (Spalten) und so besser skalierbar.

Somit ist es mit dem SAX-Sequenzeditor möglich, nach wesentlich längeren Teilsequenzen und unter Verwendung wesentlich größerer Alphabete zu suchen, was völlig neue Anwendungsmöglichkeiten eröffnet, da die definierbaren Muster so komplexer sein können. Der bereits massiv skalierbare Charakter von *VizTree* wird somit nochmals erweitert.

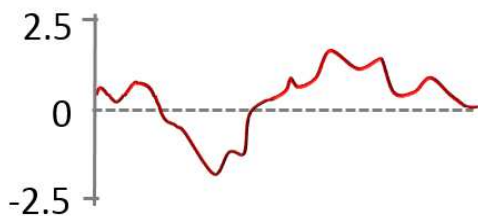
Außerdem wurde bei der Entwicklung des *SAX-Sequenzeditors* darauf geachtet, dass Musterselektionen, die nicht zu den Daten passen, visuell eindeutig und leicht erkennbar als nicht vorhanden markiert werden. Durch die Selektionsmöglichkeiten hat der Nutzer außerdem die Möglichkeit, den Suchraum auf einfache Weise effektiv einzuschränken, wodurch zielgerichtetere Suchen ermöglicht werden. Z. B. kann nach allen Sequenzen gesucht werden, die mit einem bestimmten Verlauf beginnen oder enden, ohne weitere Vorgaben zu spezifizieren. Der Editor stellt dabei eine benutzerfreundliche grafische Oberfläche dar, die eine effektive Interaktion mit sofortiger visueller Rückkopplung erlaubt. Da in symbolischen Sequenzen gesucht wird, kann jedoch von erfahrenen Benutzern prinzipiell auch die direkte Wildcard-basierte Suche [252] verwendet werden.

Ein weiterer Unterschied zwischen beiden Ansätzen ist die Tatsache, dass der SAX-Sequenzeditor auf unterschiedlich gebildeten SAX-Sequenzen operieren kann und nicht auf einer Online-Codierung der untersuchten Zeitreihe zu SAX-Sequenzen beruht. So müssen nicht zwangsweise die Originalzeitreihen ausgewertet werden, sondern es können vorberechnete SAX-Sequenzen langer Zeitreihen ausgewertet werden, welche beispielsweise zusammen mit den Codierungsparametern aus den vorberechneten Metadaten (siehe Abs. 3.4) entnommen werden können. Diese Art der Auswertung unterscheidet sich jedoch grundsätzlich von dem in *VizTree* verwendeten Ansatz, Teilsequenzen durch einen Fensterverschiebungsalgorithmus zu berechnen, der in Abbildung 4.2 dargestellt ist, wodurch sich je nach gewähltem Codierungsmodus des *SAX-Sequenzeditors* auch die Resultate unterscheiden. Das liegt v. a. daran, dass während der Teilsequenzextraktion in *VizTree* für jeden Fensterinhalt eine *Z-Normalisierung* durchgeführt wird. Diese bewirkt bei einem

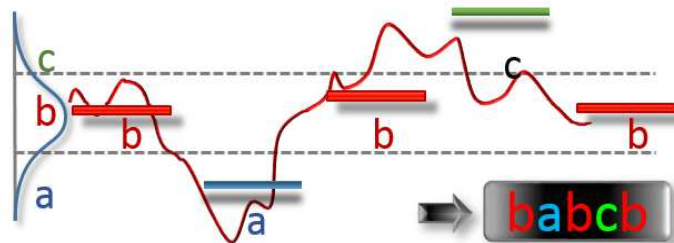
Fensterverschiebungsalgorithmus, dass der Zusammenhang zwischen dem Wertebereich innerhalb des Fensters  $W_F$  und dem Wertebereich der Gesamtzeitreihe  $W_{Ges.}$  verlorengeht, was in Abbildung 4.2 verdeutlicht wird.



Z-normalisierter Fensterinhalt:



SAX-codierter Fensterinhalt:



**Abbildung 4.2:** Ableitung von SAX-Teilsequenzen mit einem Fensterverschiebungsalgorithmus.

Das hat sowohl Vor- als auch Nachteile. Die abgeleiteten SAX-Sequenzen stellen hierdurch eher von der restlichen Zeitreihe isolierte lokale Muster dar, so dass nach „Formen an sich“, unabhängig vom globalen Verlauf und Wertebereich gesucht werden kann. Wird hingegen die gesamte Zeitreihe SAX-codiert und die resultierende SAX-Sequenz mit einem Fensterverschiebungsalgorithmus ausgewertet, bleibt der Zusammenhang erhalten, was sich jedoch zu Lasten der abstrakten Formerkennung auswirkt, da ein Suchmuster sich in diesem Fall auf den globalen Wertebereich  $W_{Ges.}$  bezieht. Welche Form der Teilsequenzableitung gewählt werden sollte, hängt von der Gewichtung bei der Bedeutung des Wertebereichs resp. der abstrakten Form für die Suche ab und

ist anwendungsabhängig. Der *SAX-Sequenzeditor* kann prinzipiell in beiden Modi betrieben werden, je nachdem, ob in bereits codierten SAX-Sequenzen aus vorberechneten Metadaten gesucht werden soll oder mittels eines Fensterverschiebungsalgorithmus Originalzeitreihen durchsucht werden sollen, für die keine Metadaten vorliegen. Das lässt sich auch auf die Komponenten für SAX-Sequenz-Histogramme und die Detektion von Discords und Motifs übertragen.

Zusammenfassend kann gesagt werden, dass eine Kombination der Darstellungen des *SAX-Sequenzeditors* und von *VizTree* sich für solche Anwendungen bewähren kann, in denen hauptsächlich nach kurzen Sequenzen gesucht werden soll. Für die Suche nach längeren und komplexeren Teilsequenzen eignet sich der Ansatz des *SAX-Sequenzeditors* jedoch aufgrund seiner kompakteren Darstellung weitaus besser.

### 4.3 Detektion von Discords und Motifs

Die Detektion von Discords und Motifs – insbesondere die Entdeckung von Discords, also ungewöhnlichen Teilsequenzen einer Zeitreihe – ist von hohem Wert für die Zeitreihenanalyse in vielen Anwendungsgebieten, was in [143] eindrucksvoll illustriert wird. Die in Abs. 3.7.6 beschriebene Detektion und Visualisierung von Discords und Motifs folgt dem *HOTSAX*-Ansatz [143] zur Detektion von Discords und dem in [180] beschriebenen Vorgehen zur Detektion von Motifs. Der *HOTSAX*-Ansatz nutzt zur Berechnung eine als *SAX-Trie* bezeichnete Baumstruktur. Die für den Ansatz benötigte Grundfunktionalität war bereits in der Softwarebibliothek *jMotif* [292] implementiert und wurde zunächst unverändert in *ViAT* integriert und um Möglichkeiten zur Suche direkt aus der Visualisierung und zur visuellen Darstellung der Ergebnisse erweitert. Durch die Verwendung der *SAX-Trie-Struktur* ist der *HOTSAX*-Ansatz jedoch nur eingeschränkt parametrisierbar und weist ohne weitere Optimierung eine hohe Laufzeitkomplexität von  $O(n^2)$  auf [143], wobei  $n$  hier für die Länge der durchsuchten Zeitreihe steht. Auch wenn optimierte Implementierungen unter Einbeziehung von Heuristiken möglich sind, welche die Anzahl der Distanzberechnungen um den Faktor 2,902 verringern [143], eignet sich der Ansatz leider nicht zur schnellen Auswertung größerer Datensätze, was sich in *ViAT* durch lange Wartezeiten (einige Minuten) bemerkbar macht. Die Software *ViAT* wurde jedoch speziell auf Interaktivität und Skalierbarkeit mit großen Datensätzen und langen Zeitreihen ausgelegt. Somit sollten auch in großen Zeitreihendaten schnell Such-Resultate vorliegen und visuell angezeigt werden.

Es wurde daher ein alternatives Verfahren zur Detektion von Discords und Motifs entwickelt und in Java umgesetzt (siehe Abs. 3.7.7), das im Folgenden mit dem *HOTSAX*-Ansatz verglichen wird.

**Vergleich der Ansätze *HOTSAX* und *SortList*:** Das neue Verfahren erhält den Namen *SortList* und sollte wesentlich schneller sein als der Ansatz auf Basis des *SAX-Trie*, da die aufwendige Be-

rechnung der Distanzen zwischen den Teilsequenzen mit dem Ähnlichkeitsmaß *MINDIST* vermieden wird, auf welche gemäß [143] ca. 99 % der Laufzeit des ursprünglichen Algorithmus zurückgeführt werden kann. Die abgeleiteten SAX-Teilsequenzen werden daher lediglich aufgrund der Anzahl an Fundstellen der einzelnen Sequenzen als Discords (selten) und Motifs (häufig) bewertet, was zu abweichenden Ergebnissen führen kann. Das neue Verfahren sammelt die Teilsequenzen in einem Fensterverschiebungsalgorithmus zusammen mit den jeweiligen Fundstellen in einer Liste, welche anschließend mit dem *Quicksort*-Algorithmus nach Auftrittshäufigkeit in aufsteigender Reihenfolge sortiert werden. Der Fensterverschiebungsalgorithmus kann sehr effizient unter Verwendung einer *doppelseitigen Queue*-Datenstruktur mit einer Laufzeitkomplexität von  $O(n)$  umgesetzt werden. Die Komplexität des Quicksortalgorithmus ist fallabhängig. Die sich ergebenden Komplexitäten können aus Tabelle 4.1 entnommen werden. Im Durchschnitt sollte sich eine Gesamtlaufzeitkomplexität von  $O(n \log n)$  ergeben.

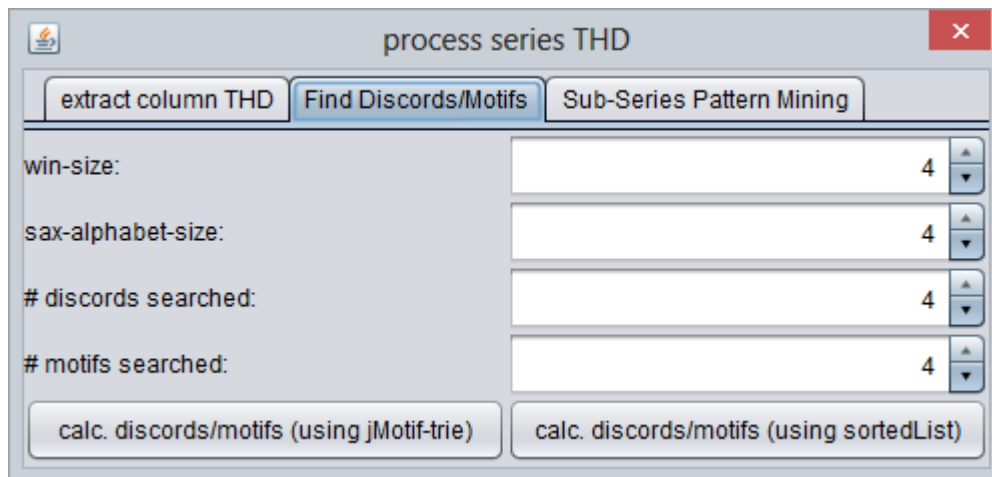
**Tabelle 4.1:** Fallabhängige Komplexität des Quicksort-Algorithmus [62].

<i>Fall</i>	<i>Laufzeitkomplexität</i>	<i>Kapazitive Komplexität</i>
<i>Schlechtester Fall</i>	$O(n^2)$	$O(n)$
<i>Beste Fall</i>	bei Zweifachzerlegung: $O(n \log n)$ bei Dreifachzerlegung: $O(n)$	$O(\log n)$
<i>Durchschnitt</i>	$O(n \log n)$	$O(n)$

Ein weiterer Vorteil des neuen Verfahrens ist die Tatsache, dass die Anzahl gesuchter Discords und Motifs hier nicht spezifiziert werden muss, da die Liste ohnehin vollständig sortiert wird und damit eine definierte Reihenfolge, sowohl für Discords als auch für Motifs, vorliegt. Allerdings stellt sich die Frage, welche Anteile der sortierten Liste als Discords und Motifs bewertet werden sollten. Einträge aus der Listenmitte sollten z. B. weder als Discord noch als Motif behandelt werden. Da insbesondere die genaue Anzahl gesuchter Discords daten- und anwendungsabhängig ist (der *HOTSAX*-Algorithmus beschränkt sich auf die Suche eines einzigen Discords), wurden deshalb die Parameter zur Spezifikation der Anzahl gesuchter Discords und Motifs aus *HOTSAX* beibehalten, um die Anzahl anzuzeigender Resultate festzulegen (siehe Abbildung 4.3).

**Visueller Vergleich der Suchresultate und Anpassung von *SortList*:** Beide Ansätze (*HOTSAX*, *SortList*) wurden implementiert und in *ViAT* integriert, so dass die Resultate einheitlich visualisiert und hinsichtlich der Güte der Detektionsergebnisse bei bekannter Laufzeit der Verfahren untersucht werden können. Hierzu wurden dieselben Testdaten verwendet, welche bereits in Abs.

3.7.6 zur Illustration der Funktionsweise der Detektion von Discords und Motifs verwendet wurden (Im Folgenden *T1* genannt). In jenen Testdaten wurden mehrere Suchdurchläufe mit verschiedenen Suchparametern durchgeführt und die Ergebnisse gegenübergestellt. In Abbildung 4.4 sind die Suchresultate dargestellt, die sich unter Verwendung der Parameter aus Abbildung 4.3 ergeben.



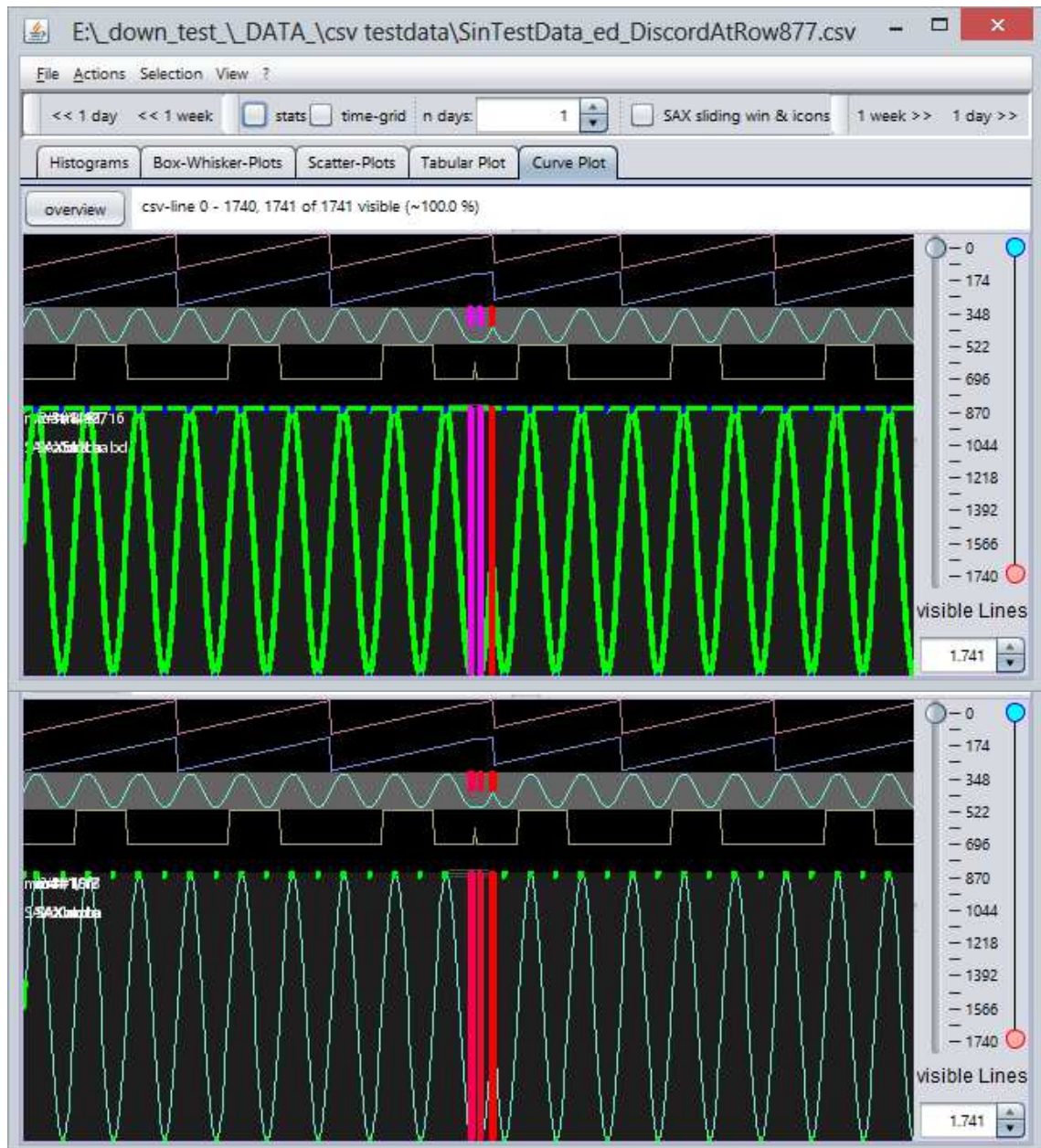
**Abbildung 4.3:** Dialog zur Eingabe der gemeinsamen Suchparameter für die Detektionsvarianten *HOTSAX* und *SortList*.

Die Darstellung des *HOTSAX*-Resultats zeigt Discords an Positionen an, die visuell mit denen im *SortList*-Suchergebnis übereinstimmen, jedoch mit leicht unterschiedlicher Färbung. Das liegt daran, dass im *SortList*-Ansatz nicht die *MINDIST*-Distanzen zum jeweils nächsten Nachbarn ermittelt werden, weshalb die Farbgebung alleine auf der jeweiligen Position der Teilsequenzen in der sortierten Liste beruht, die durch die Gesamtanzahl möglicher Fensterpositionen geteilt wird. Was die dargestellten Motifs betrifft, so wird in der *HOTSAX*-Variante nahezu die gesamte Kurve mit der Darstellung von Motifs überlagert, während in der *SortList*-Variante nur einige wenige Motifs dargestellt werden, nämlich gemäß der Suchparameter die vier häufigsten.

Die unterschiedlichen Suchergebnisse resultieren aus der unterschiedlichen Ableitung und Sammlung der SAX-Teilsequenzen und ihrer Beurteilung als Discord bzw. Motif. Während bei der *SortList*-Suche die *PAA-Größe* für die SAX-Codierung des Fensterinhalts frei gewählt werden kann, muss sie aufgrund der verwendeten *SAX-Trie-Struktur* stets bei der *HOTSAX*-Suche mit der Alphabetgröße übereinstimmen, die bei diesem Suchlauf jedoch identisch mit der Fenster- und Alphabetgröße gewählt wurde, um vergleichbare Ergebnisse zu erhalten. Die geringe Zahl dargestellter Motifs kommt dadurch zustande, dass bei der Sammlung der Teilsequenzen im gewählten Modus jeweils nur Sequenzen aufgenommen werden, die sich von der zuletzt eingeord-



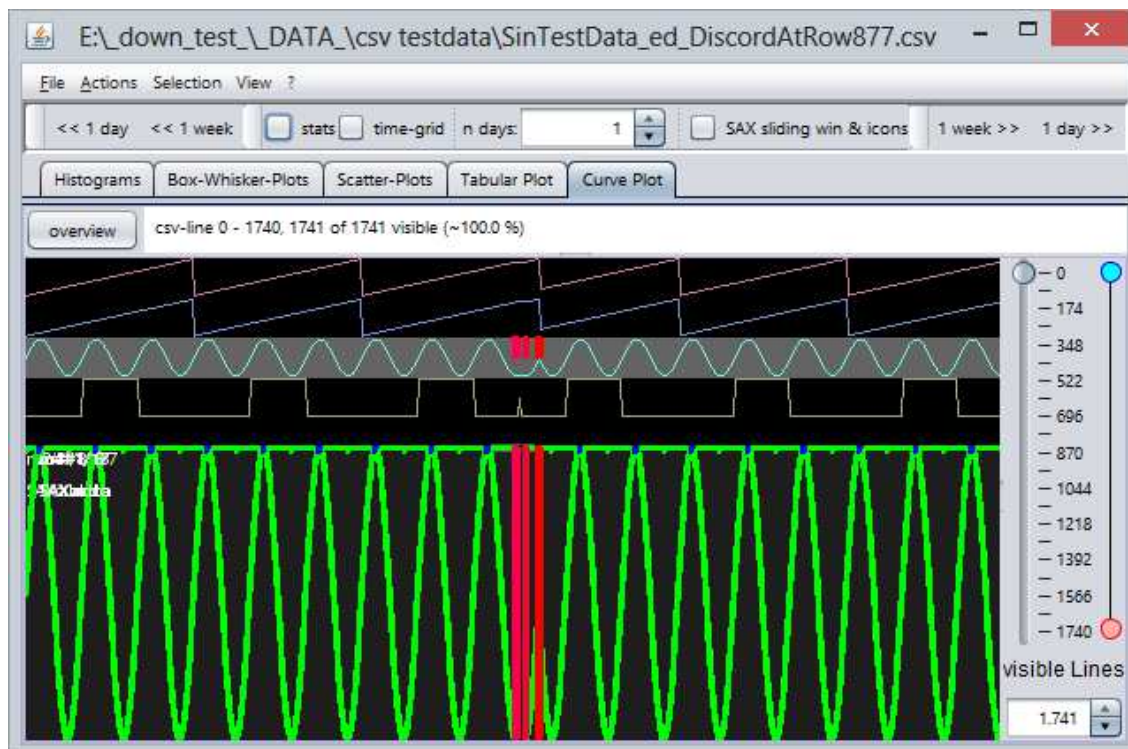
neten unterscheiden. Das führt dazu, dass nur das erste Auftreten einer sich wiederholenden Sequenz markiert wird. Um die beiden Verfahren besser vergleichen zu können, wird ein neuer Suchlauf mit gleichen Parametern gestartet, für welchen die Option deaktiviert wird.



**Abbildung 4.4:** Suchergebnis unter Verwendung der Parameter aus Abbildung 4.3 mit der *HOTSAX*-Implementierung (oben) und der *SortList*-Implementierung (unten).

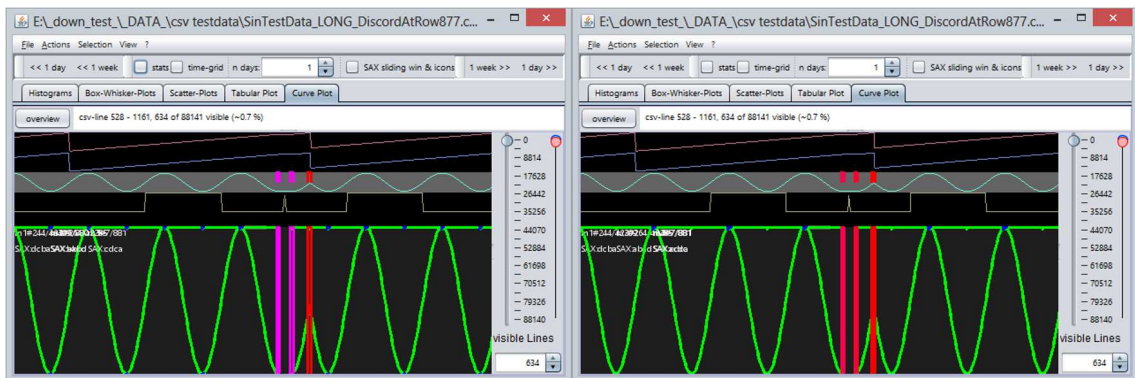
Wie aus der massiven Grünfärbung in Abbildung 4.5 hervorgeht, resultiert die Änderung in mehr angezeigten Motifs, da nun auch identische aufeinanderfolgende Sequenzen in die Liste aufge-

nommen werden. Da die Fenstergröße beim Suchdurchlauf wesentlich kleiner als die Periodendauer der Sinusschwingung gewählt wurde, ergibt sich jeweils beim Durchlaufen der Flanken einer Periode eine Abfolge des jeweils selben Motifs. Bei einer ansteigenden Flanke ist das die SAX-Sequenz *abcd* und bei fallender Flanke *dcba*. Die beiden Sequenzen kommen am häufigsten vor, gefolgt von den Sequenzen, welche an den lokalen Maxima (*cdca*, *dcba*) und Minima (*babd*, *abcd*) gebildet werden. Die Positionen der Discords werden durch das Zulassen der Kollektion wiederholter Sequenzen nur geringfügig verändert.



**Abbildung 4.5:** Suchergebnis unter Beibehaltung der Parameter, mit der Implementierung *SortList*, ohne Ausschluss aufeinanderfolgender identischer Teilsequenzen.

**Laufzeitvergleich:** Da die Laufzeiten der Suchläufe in den Testdaten *T1* mit 1741 Zeitschritten für beide Detektionsvarianten nur einige Millisekunden betragen, wurde der Testdatensatz auf 88 141 Zeitschritte vergrößert, um die Laufzeitunterschiede eindeutiger erfassen zu können (im Folgenden *T1\_lang* genannt). Dabei wurden keine zusätzlichen Discords in die Sinusschwingung eingefügt. Wie in Abbildung 4.6 ersichtlich, stimmen die Suchresultate mit denen vorangegangener Suchläufe in *T1* überein. Für beide Detektionsvarianten wurden anschließend (abwechselnd) drei Suchläufe auf der Sinusschwingung durchgeführt, um zusätzlich zu den Laufzeiten auch deren Varianzen beurteilen zu können. Die gemessenen Laufzeiten werden in Tabelle 4.2 präsentiert. Es zeigt sich, dass die Detektion von Discords und Motifs mit dem neuen *SortList*-Ansatz etwa um den Faktor 1200 schneller abläuft als mit *HOTSAX*.



**Abbildung 4.6:** Resultate der Suchläufe in den erweiterten Testdaten  $T1\_lang$  (Länge: 88 141 Samples), mit *HOTSAX* (links) und *SortList* (rechts).

**Tabelle 4.2:** Vergleich der Laufzeit der Discord/Motif-Detektion mit dem *HOTSAX*- und *SortList*- Ansatz.

<i>Lauf</i>	<i>Laufzeit HOTSAX</i>	<i>Laufzeit SortList</i>
1	0 min 52 s 901 ms	56 ms
2	1 min 13 s 213 ms	57 ms
3	1 min 03 s 188 ms	46 ms
<i>Durchschnitt</i>	1 min 03 s 101 ms	53 ms

**Vergleich der Detektionsergebnisse:** Betrachten wir nun die jeweiligen Ergebnisse der Detektion genauer. Gemäß der Suchparameter wurde nach den „besten“ vier Discords und vier Motifs gesucht. Die Ergebnisse werden in Code 4.1 als zusammengeführte Konsolenausgabe beider Suchläufe aufgeführt. Bei der Betrachtung fallen einige Unterschiede auf. Die drei „besten“ gefunden Motifs stimmen für beide Varianten überein. Als viertes Motif wird jedoch mit dem *SortList*-Ansatz die Sequenz *acdc* mit 881 Vorkommen detektiert und mit dem *HOTSAX*-Ansatz die Sequenz *babd* mit 880 Vorkommen. Der Unterschied ist auf die unterschiedliche Sequenz-Bewertung der Ansätze zurückzuführen. Während der *SortList*-Algorithmus die häufigsten Muster als Motifs wertet, ohne die tatsächlichen MINDIST-Distanzen zu berechnen, wird eine Sequenz mit 881 Vorkommen vor einer Sequenz mit nur 880 Vorkommen eingruppiert. Beide Sequenzen stammen aus Bereichen lokaler Extrema. *acdc* enthält ein lokales Maximum – es repräsentiert einen „Gipfel“ der Sinusschwingung – und *babd* enthält entsprechend ein lokales Minimum.

**Results SortList:**

**Motifs**, as <sax>, <frequency>, last to first:

**acdc**, 881  
**cdca**, 881  
**abcd**, 42285  
**dcba**, 42309

**Discords**, as <sax>, <offset>, <distance>, last to first:

**cccc**, at 874, 0.00007942180922881423  
**bbbd**, at 891, 0.00009076778197578769  
**abdb**, at 914, 0.00010211375472276115  
**bdba**, at 915, 0.00011345972746973462

**Results HOTSAX:**

**Motifs**, as a list <sax>, <frequency>, from last to first:

**babd**, 880  
**cdca**, 881  
**abcd**, 42285  
**dcba**, 42309

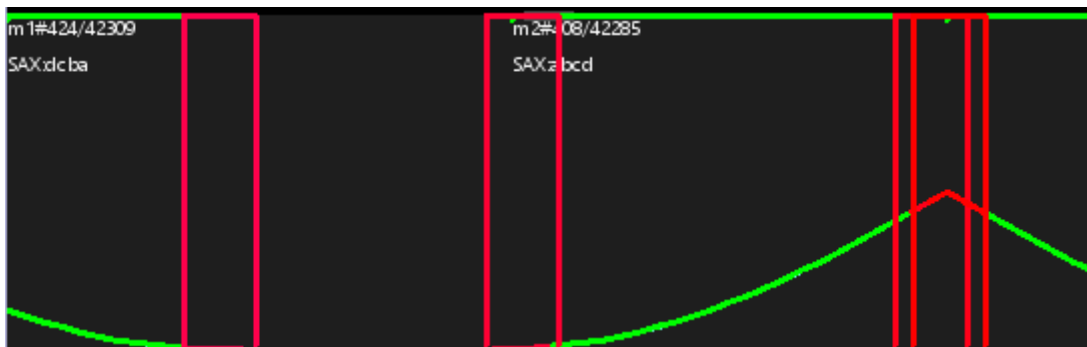
**Discords**, as <sax>, <offset>, <distance> from last to first:

**dbbb**, at 873, 0.000986636  
**bbbd**, at 891, 0.000986636  
**abdb**, at 914, 0.062112574464780225

**Code 4.1:** Konsolenausgabe der Discord/Motif-Detektion in den Testdaten *T1\_lang*, mit dem SortList-Ansatz (oben, blau) und dem HOTSAX-Ansatz (unten, grün), zusammengeführt und formatiert.

Bei Betrachtung der detektierten Discords fällt als erstes auf, dass, trotz der Vorgabe von vier zu detektierenden Discords, mit dem *HOTSAX*-Ansatz lediglich drei Sequenzen als Discords klassifiziert wurden. Der Grund hierfür kann wiederum in der Bewertung der Sequenzen aufgrund der Distanzen zu den „nächsten Nachbarn“ (im Sinne der geringsten Distanz) gesucht werden.

Diese Eigenschaft des *HOTSAX*-Ansatzes ist als Vorteil zu werten, da nur tatsächlich signifikante Abweichungen vom Normalverhalten der Zeitreihe als Discord gewertet werden. Wie sich jedoch zeigt (siehe Abbildung 4.7), überlappen die beiden letzten durch *SortList* detektierten Discords (*bdba* an Position 915 und *abdb* an Position 914). Das ist der wahre Grund dafür, dass nur drei Discords durch *HOTSAX* detektiert werden, da hier keine überlappenden Discords zugelassen werden (siehe Definition 5 in [143]). Der *SortList*-Ansatz kann problemlos um eine Zusammenfassung überlappender Discords erweitert werden, was ein sinnvollerer Vorgehen zu sein scheint als der Ausschluss überlappender Discords, da ansonsten der detektierte Discord von der Richtung abhängt, in welcher das Fenster verschoben wird.



**Abbildung 4.7:** Vergrößerter Ausschnitt des SortList-Resultats mit zwei nicht überlappenden (links) und zwei überlappenden Discords (rechts, rot).

Als „bester“ Discord wurde mit dem *HOTSAX*-Ansatz die Sequenz *abdb* detektiert, welche sich im Resultat von *SortList* auf Platz zwei befindet, zusammen mit dem überlappenden Discord *bdba*, welcher hier als bester Discord gefunden wurde. Das für die Visualisierung entscheidende Ergebnis, die Fundstelle des besten Discords, stimmt also bei einer Positionsabweichung von einer Zeiteinheit nahezu mit dem *HOTSAX*-Resultat überein. Auch der von *HOTSAX* an Position 891 detektierte Discord *bbbd* findet sich auf der nachfolgenden Position im Resultat von *SortList*. Der letzte durch *HOTSAX* detektierte Discord *dbbb* an Position 873 stimmt zwar nicht hinsichtlich der Sequenz mit dem von *SortList* detektierten letzten Discord *cccc* an Position 874 überein, jedoch überlappen sich die beiden Discords, wodurch die entscheidende Information, die Position, wiederum mit einer Abweichung von einer Zeiteinheit übereinstimmt. Insgesamt zeigt sich, dass der neu entwickelte *SortList*-Ansatz bei der Anwendung auf die Testdaten *T1* und *T1\_lang* ebenso leistungsfähig bei der Detektion häufiger und auffälliger seltener Teilsequenzen wie der *HOTSAX*-Ansatz ist, wobei der Suchlauf in *T1\_lang* mit *SortList* etwa 1200 mal schneller durchgeführt werden kann.

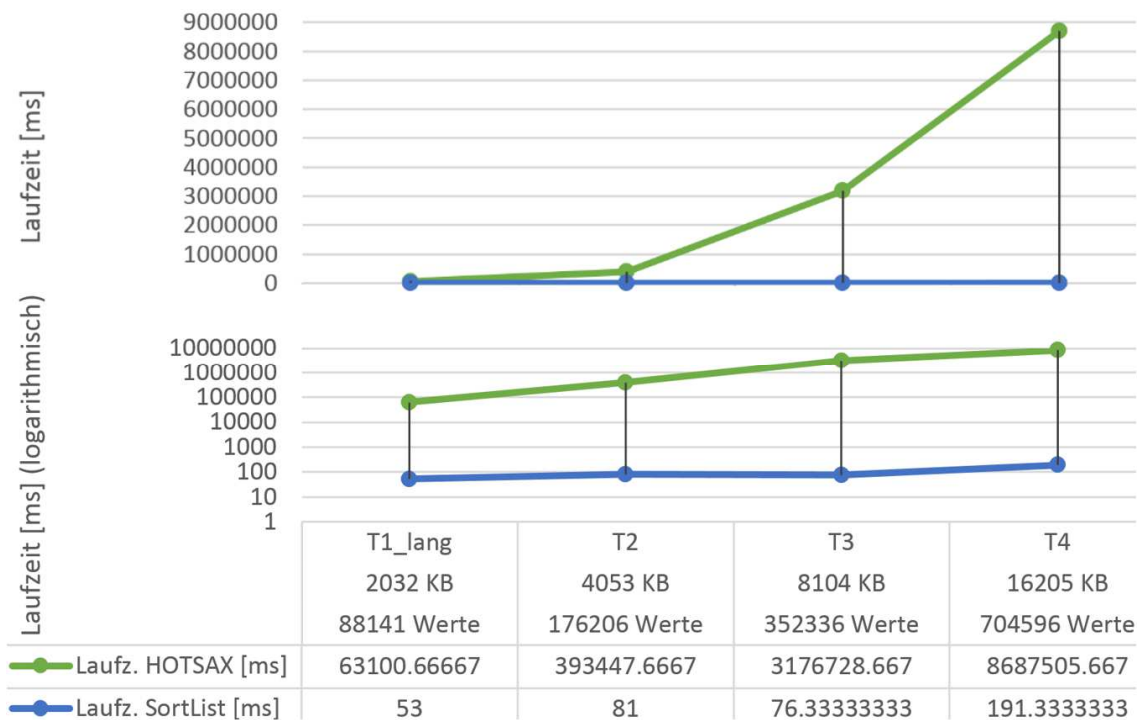
**Zusammenhang zwischen Laufzeit und Zeitreihengröße:** Um den Zusammenhang zwischen Laufzeit und Zeitreihengröße (bzw. Datenmenge) zu beleuchten, wurden weitere Sinus-Testdaten (*T2*, *T3* und *T4*) generiert, welche Konkatenationen der Einzelwerte aus *T1\_lang* darstellen, so dass auch der Bereich mit Discords erneut auftritt. Dabei wurde die Anzahl der Zeitschritte jeweils ungefähr verdoppelt. Die Testdaten wurden mit beiden Ansätzen unter Beibehaltung der zuvor verwendeten Suchparameter durchsucht. Die Ergebnisse werden in Tabelle 4.3 aufgeführt.

**Tabelle 4.3:** Gemessene datenabhängige Laufzeiten der Discord/Motif-Suche in den diversen Testdaten für den *SortList*- (blau) und *HOTSAX*-Ansatz (grün).

	<i>T1_lang</i>	<i>T2</i>	<i>T3</i>	<i>T4</i>
<i>Dateigröße</i>	2032 KB	4053 KB	8104 KB	16205 KB
<i>Anzahl Zeitschritte</i>	88141	176206	352336	704596
<i>Laufzeit 1 SortList</i>	56 ms	79 ms	109 ms	221 ms
<i>Laufzeit 2 SortList</i>	57 ms	94 ms	63 ms	164 ms
<i>Laufzeit 3 SortList</i>	46 ms	70 ms	57 ms	189ms
<i>Laufzeit 1 HOTSAX</i>	0 min 52 s 901 ms	5 min 17 s 324 ms	57 min 4 s 751 ms	2 h 25 min 45 s 955 ms
<i>Laufzeit 2 HOTSAX</i>	1 min 13 s 213 ms	8 min 05 s 229 ms	43 min 45 s 11 ms	2 h 14 min 23 s 851 ms
<i>Laufzeit 3 HOTSAX</i>	1 min 03 s 188 ms	6 min 17 s 790 ms	58 min 0 s 424 ms	2 h 34 min 12 s 711 ms

Die Ergebnisse deuten im Falle der *HOTSAX*-Methode auf einen exponentiellen Zusammenhang zwischen Zeitreihengröße bzw. Datenmenge und Laufzeit hin. *HOTSAX* ist daher als schlecht mit der Datenmenge skalierbar einzuschätzen. Für *T3* ist die Wartezeit von fast einer Stunde außerdem inakzeptabel lang für die Integration in eine interaktive Zeitreihenexplorationsumgebung. Die Laufzeiten der *SortList*-Methode sind vergleichsweise kurz und weisen einen eher linearen Zusammenhang zur Datenmenge auf (die mittlere Laufzeit der Suche in *T3* lässt sich für manche Durchgänge kaum von der in *T1\_lang* unterscheiden, weshalb die empirische Untersuchung keine verlässlichen Schlüsse zulässt), so dass das Verfahren auch für Suchen nach Discords und Motifs in sehr großen Datensätzen und zur interaktiven Zeitreihenexploration geeignet ist. Da der in *SortList* eingesetzte Quicksort-Algorithmus jedoch den Hauptspeicher benötigt, ergeben sich auch hier Grenzen bei der Größe durchsuchbarer Datensätze.

Zur Verdeutlichung wurden für Abbildung 4.8 die Ergebnisse aus je drei Durchläufen pro Datensatz gemittelt, da die Laufzeiten für denselben Datensatz in Tabelle 4.3 deutliche Varianzen aufweisen. Die exponentielle Laufzeitabhängigkeit des *HOTSAX*-Ansatzes von der Datengröße zeigen sich in Abbildung 4.8 deutlich im entsprechenden Kurvenverlauf (obere grüne Kurve). Eine ungefähre Verachtfachung der Werteanzahl (*T4* im Vergleich zu *T1\_lang*) hat für *HOTSAX* im Mittel eine ungefähre Laufzeitverlängerung um Faktor 140 zur Folge, bei *SortList* hingegen nur um Faktor ca. 3,6.

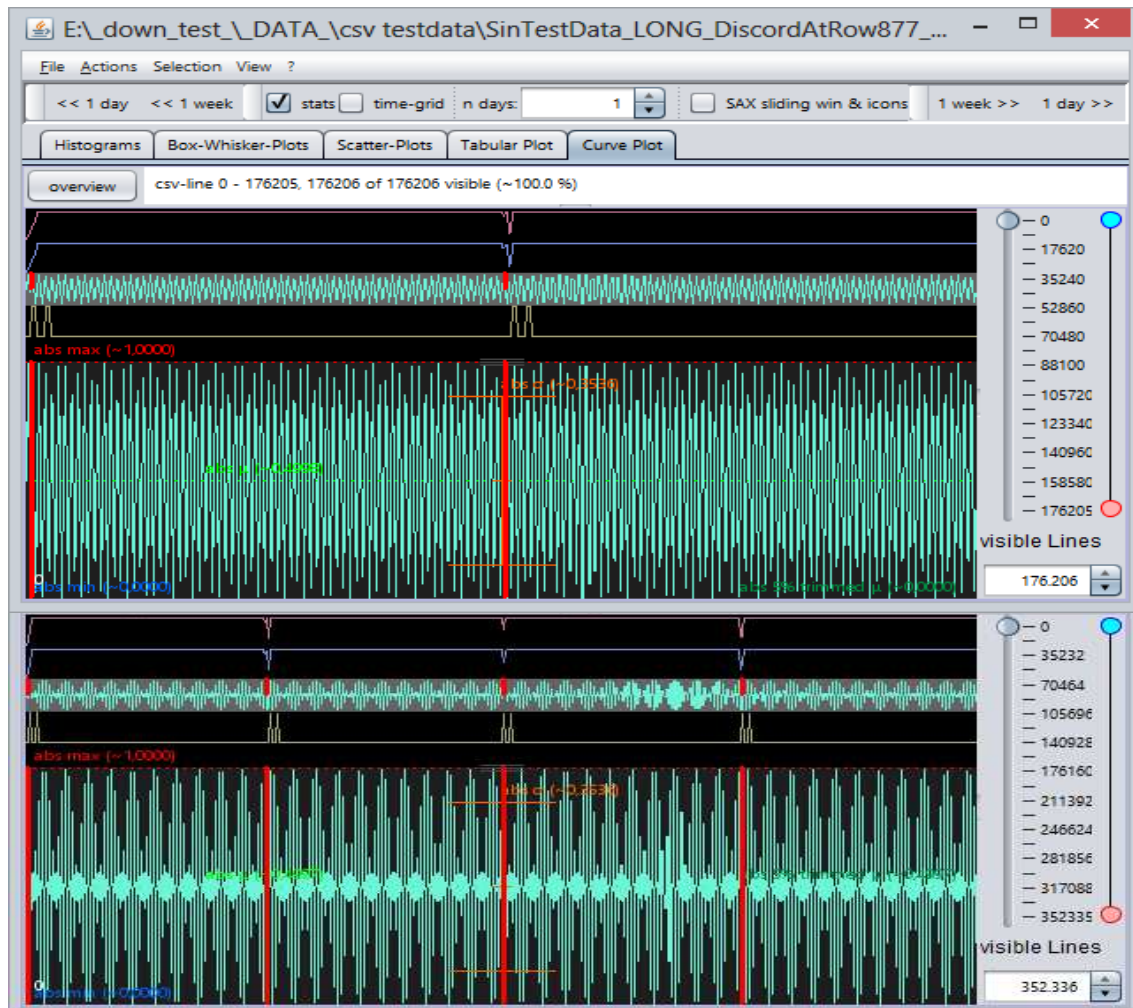


**Abbildung 4.8:** Diagramm zum Laufzeitvergleich zwischen HOTSAX (grüne Kurve) und SortList (blaue Kurve) aufgrund gemittelter Laufzeiten über drei Testdurchläufe pro Testdatensatz, mit linearer vertikaler Zeitskala (Diagramm oben) und logarithmischer Zeitskala (Diagramm unten), sowie einer tabellarischen Listung von Eigenschaften der jeweiligen Testdatensätze (unten).

**Vergleich der Detektionsergebnisse für Suchen in T2 und T3:** Über die stark unterschiedlichen Laufzeitergebnisse hinaus unterscheiden sich auch die Detektionsergebnisse der beiden Ansätze signifikant. So wurden bei der Suche mit dem *HOTSAX*-Verfahren keinerlei Discords in T2 gefunden, jedoch erwartungsgemäß zwei Discords in T2 und vier Discords in T3 mit dem *SortList*-Verfahren. Da es sich hier um ein wiederholt vorkommendes Discord-Muster handelt – die Daten aus *T1\_lang* wurden lediglich konkateniert – findet das *HOTSAX*-Verfahren keinen Discord, da die entsprechenden Sequenzen nun mehrmalig auftreten und sich somit Distanzen zum nächsten Nachbar von Null ergeben. Das folgt aus den Definitionen in [143] und ist in Fällen wie diesem eine Schwachstelle des *HOTSAX*-Verfahrens.

Da das *SortList*-Verfahren auf der Sortierung der Teilsequenzliste nach Auftrittshäufigkeit beruht, unterscheiden sich die Definitionen von Discords und Motifs von denen des *HOTSAX*-Ansatzes. Als Discords werden hier trivialerweise die seltensten nichtüberlappenden Teilsequenzen und als Motifs die häufigsten Teilsequenzen aufgefasst, was sich für die meisten Anwendungsfälle als

sinnvoll erweist. Insbesondere ist mit zunehmender Zeitreihengröße damit zu rechnen, dass die gesuchten ungewöhnlichen Muster mehrmalig in sehr ähnlicher oder gleicher Weise auftreten.



**Abbildung 4.9:** Resultate der Suchen in den Testdaten  $T_2$  (oben) und  $T_3$  (unten) mit dem SortList-Ansatz mit zwei bzw. vier detektierten Discords (die Anzeige von Motifs wurde der Übersicht halber deaktiviert). Eine entsprechende Suche mit dem HOTSAX-Ansatz detektierte keine Discords. Die Suchparameter wurden nicht verändert und entsprechen Abbildung 4.3.

Das *SortList*-Verfahren wurde hauptsächlich mit dem Ziel entwickelt, Analysten bei der visuellen Exploration zu unterstützen, indem auf alle ungewöhnlichen Muster hingewiesen wird und häufig wiederkehrende Muster leichter identifiziert werden können. Dieses Ziel erfüllt das neue Verfahren aufgrund der Evaluationsergebnisse bei einer sehr guten Skalierbarkeit mit wachsendem Datenvolumen gut.



## 4.4 Strukturanalyse MetaSAX

Der SAX-Strukturcharakteristik-Editor, welcher in Abs. 3.7.3 vorgestellt wurde, ermöglicht eine strukturelle Analyse von Zeitreihen auf Basis der SAX-Repräsentation, welche besondere Merkmale der Zeitreihe zu extrahieren vermag, mit denen eine neuartige Charakterisierung des zeitlichen Werteverlaufs möglich ist. Eine Zeitreihe kann damit auf einem sehr hohen Abstraktionslevel beschrieben werden. So können aufgrund der extrahierten Struktur-Metadaten sehr effizient Fragen zu einer Zeitreihe beantwortet werden wie:

- „Enthält die Zeitreihe...
  - ... einen zunehmenden bzw. abnehmenden Trend?“
  - ... einen positiven oder negativen Shift? (Und wie viele?)“
  - ... eine Zylinderform?“
- „Wie viele ‚nicht-kurzfristige‘<sup>33</sup> Maxima bzw. Minima enthält die Zeitreihe?“
- „An welchen Stellen ändert sich das Verhalten der Zeitreihe“

U. a. können diese Merkmale bei der Ähnlichkeitsbewertung von Zeitreihen verwendet werden, die für viele Analyseaufgaben wie z. B. der Klassifizierung, dem Clustering und der inhaltsbasierten unscharfen Suche in Zeitreihendaten (siehe Anhang A.1.3) benötigt wird.

Die verschiedenen Methoden der Ähnlichkeitsbewertung haben unterschiedliches Verhalten beim Umgang mit verrauschten Daten, Ausreißern, der Ähnlichkeitsbewertung von Zeitreihen unterschiedlicher Länge, unterschiedlicher Skalierung von Amplituden, Trends oder der zeitlichen Verzerrung von Zeitreihen. Das meistverwendete Ähnlichkeitsmaß, die punktweise Euklidische Distanz (siehe Anhang A.1.4.3), besitzt zwar eine niedrige Laufzeitkomplexität von  $O(n)$  und eignet sich daher gut zur Bewertung großer Zeitreihendaten. Sie erfordert jedoch eine intensive Vorverarbeitung der Daten (Anhang A.1.2), da sie sich nicht robust gegenüber den beschriebenen Problemen zeigt. Eine Vorverarbeitung von Zeitreihen ist jedoch meist anwendungsabhängig, da hierzu ein Modell des Zustandekommens der Daten notwendig ist, welches das Normverhalten beschreibt und somit festlegt, in welchem Umfang Rauschen gefiltert werden kann, welche Werte als Ausreißer entfernt werden dürfen etc. Ein solches Modell ist aber insbesondere für Zeitreihendaten, die als Big Data bezeichnet werden können, meist nicht gegeben. Außerdem ist die punktweise Euklidische Distanz nicht für Zeitreihen unterschiedlicher Länge definiert.

Als mächtigstes Ähnlichkeits- bzw. Distanzmaß für Zeitreihen gilt nach wie vor die DTW-Methode [28] (s. Anhang A.1.4.4), da sie in der Lage ist, selbst Zeitreihen unterschiedlicher Länge

---

<sup>33</sup> Gemeint sind hier Maxima und Minima, die auch nach Durchschnittsbildung über einige benachbarte Messwerte noch sichtbar sind. Manche Extrema – z. B. solche, die nur einen Messwert umfassen – gehen durch die zeitliche Aggregation bei der SAX-Codierung verloren.

oder Auflösung zu bewerten und sich robust zeigt gegenüber einer zeitlichen Verzerrung in Zeitreihen (sog. *Warping*). Es existieren besondere Implementierungen von DTW (z. B. UCR-DTW [243], [209]), die unter bestimmten Umständen eine Komplexität von  $O(n)$  erreichen können. Wird einer Zeitreihendatenbank jedoch eine neue Zeitreihe hinzugefügt, muss die gesamte Datenbank erneut mittels DTW bewertet werden – es existiert kein Verfahren für eine iterative Indizierung für DTW.

Die in Abs. 3.7.3 eingeführte Strukturbeschreibung *MetaSAX* weist Eigenschaften auf, die in diesem Kontext neu erscheinen und wird daher im Folgenden genauer beleuchtet und anderen Verfahren gegenübergestellt.

Bereits die SAX-Repräsentation einer Zeitreihe stellt eine Beschreibung derselben auf hohem Abstraktionsniveau dar. Eigenschaften, wie zeitliche Verzerrungen der Originalzeitreihe, bleiben jedoch bei der Codierung erhalten. Die SAX-Repräsentation einer Zeitreihe kann nun zusätzlich durch das *MetaSAX*-Verfahren ausgewertet werden. Dabei ergibt sich eine Beschreibung der Struktur, welche zwar die Reihenfolge enthaltener Grundformen erhält (*Plateau*, *Negativtrend*, *Positivtrend*, siehe Abs. 3.7.3), jedoch unabhängig vom linearen Zeitverlauf der Originalzeitreihe ist. Somit sind – auf völlig unterschiedliche Weise – Auswertungen möglich, welche ähnlich flexibel mit der Zeitreihenlänge und zeitlichen Verzerrungen umzugehen vermögen wie DTW. Insbesondere erlaubt das Verfahren Ähnlichkeitsbewertungen für Zeitreihen basierend auf abstrakten Merkmalen, welche auch bei zeitlich unterschiedlicher Skalierung, zeitlichem *Warping*, unterschiedlichen Wertebereichen und Zeitreihen unterschiedlicher Länge möglich sind. Die Genauigkeit der zeitlichen Auflösung und der Diskretisierung des Wertebereichs können dabei über die SAX-Parameter und einige *MetaSAX*-spezifische Parameter, wie z.B. Maximalzahl zu kombinierender *Plateaus*, zu verwendendes *Startzeichen* und Textgröße, sehr flexibel angepasst werden.

Eine *MetaSAX*-Beschreibung einer Zeitreihe ist dabei völlig unabhängig von der Anzahl an Datenpunkten der Originalzeitreihe. So können beispielsweise verhältnismäßig „einfache“, z. B. monotone Zeitreihen oder entsprechende Zeitreihenabschnitte äußerst kompakt beschrieben werden, wodurch sich gegenüber SAX eine starke zusätzlich Datenverdichtung bzw. Kompression ergibt. Möglicherweise lässt sich hieraus auch ein abstraktes kompressionsbasiertes Komplexitätsmaß ableiten (ähnlich wie durch die in Abs. 3.4.3 beschriebene RLE-Kompression für SAX).

## 4.5 Datenparallel auf dem Hadoop-Cluster erzeugte Zeitreihenmetadaten

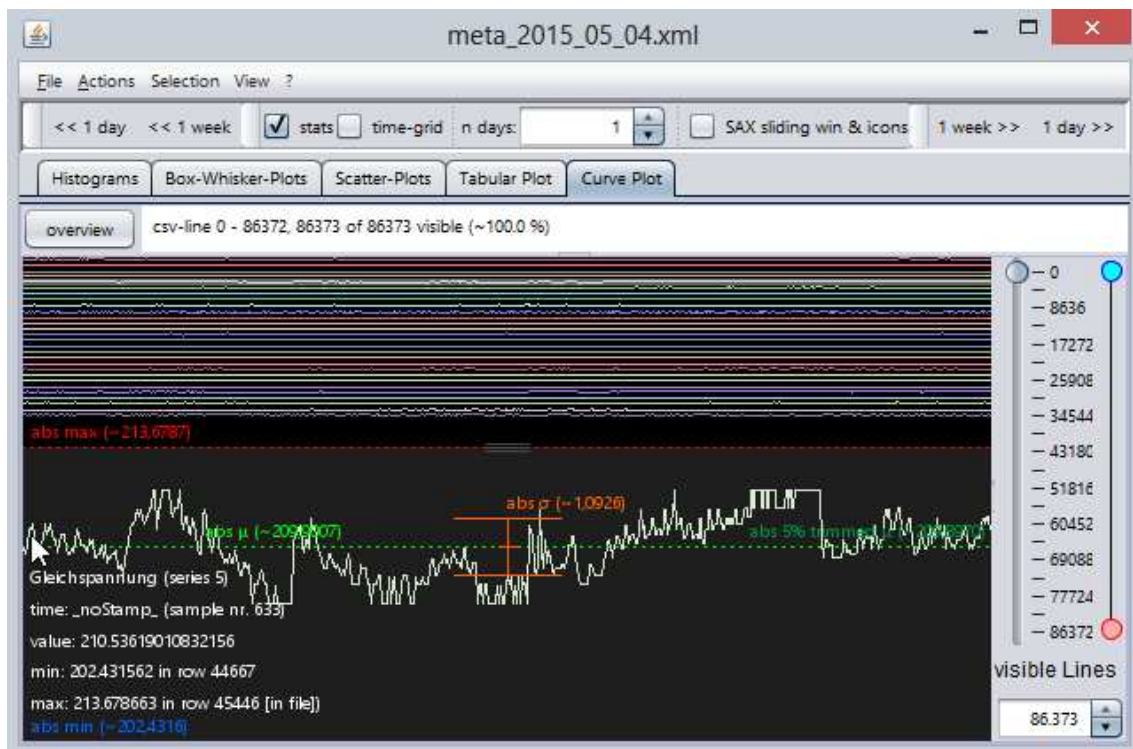
Die in Abschnitt 3.4 auf die Erzeugung der Metadaten bezogene Darstellung der datenparallelen Metadatenerzeugung in Hadoop stellt eine sehr effektive Möglichkeit dar, durch Datenreduktion sehr großer Zeitreihendaten wesentlich effizienter auf diese zugreifen zu können. Nach der Datenreduktion können die wesentlich kleineren Metadaten auch auf Standardarchitekturen weiter ausgewertet werden.

**Performanz der Erzeugung:** In Abs. 3.2 wurde die Performanz und Skalierbarkeit der datenparallelen Verarbeitung mit Apache Pig auf dem Hadoop-Cluster bereits evaluiert, die Ergebnisse sind in [18] veröffentlicht. Für die verwendete Hardware und die ausgewerteten Daten zeigte sich die MapReduce-basierte Verarbeitung bereits bei verhältnismäßig kleinen Datenmengen als deutlich performanter und besser skalierbar im Vergleich zur taskparallelen Verarbeitung auf zwei leistungsfähigen Mehrkernrechnern. Insbesondere zeigte sich bei der Zeitreihenauswertung in Hadoop eine eher linear mit der Datenmenge zunehmende Laufzeit, während bei der taskparallelen Verarbeitung die Laufzeit exponentiell mit der Datenmenge zunahm. Da es die Datenparallelität der Verarbeitung ist, auf welche der Zugewinn an Performanz zurückzuführen ist, sollte sich die Aussage prinzipiell auch auf andere Auswertungsvorgänge übertragen lassen, bei denen der Flaschenhals hauptsächlich durch Datenmanipulationen und nicht ausschließlich aufgrund der Komplexität der Rechenoperationen verursacht wird, was ab einer kritischen Datenmenge stets der Fall ist.

**Metadaten:** Die Metadaten, die für jeden einzelnen Zeitreihendatensatz gebildet werden, enthalten für jede Merkmalszeitreihe eine statistische Zusammenfassung mit den Extremwerten und den entsprechenden Positionen innerhalb des Datensatzes, welche ein schnelles Auffinden von Datensätzen erlauben, in denen einzelne Merkmale ungewöhnlich hohe oder tiefe Werte aufweisen. Über die enthaltene Position innerhalb der Datensätze ist auch der genaue Zeitpunkt bekannt. Aufgrund der geringen Größe der Metadaten können sie auf einen Client-Rechner übertragen werden und so auch lokal durchsucht werden – auch über sehr große Zeitbereiche. Da die Metadaten SAX-Codierungen der ursprünglichen Zeitreihendaten enthalten und auf letztere über die Dateinamen verweisen, können auf dieser Basis Anwendungen wie z. B. zweistufige unscharfe Inhalts-Suchen realisiert werden. Hierzu können u. a. die in Abs. 3.7 dargestellten SAX-basierten Analysewerkzeuge verwendet werden, v. a. der SAX-Sequenzeditor und das SAX-basierte Clustering.

Die enthaltenen SAX-Sequenzen können aber auch in herkömmliche Zeitreihen rücktransformiert und in *ViAT* visualisiert und ausgewertet werden. Abbildung 4.10 zeigt die Darstellung einer Metadatendatei, die für eine EDR-Merkmalsdatei gebildet wurde. In der Detaildarstellung im

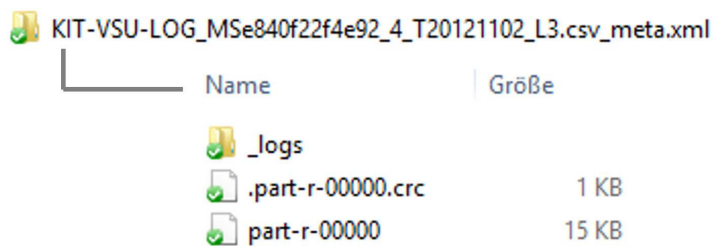
unteren Bereich ist die aus einer SAX-Sequenz rekonstruierte Kurve der Gleichspannung zu sehen. Die Kurvendarstellung leidet natürlich am Genauigkeitsverlust durch die Bereichsaufteilung in 16 Symbole, es ist jedoch für jedes Merkmal eine SAX-Sequenz mit einer hohen Alphabetgröße von 16 in den Metadaten enthalten, bei deren Codierung keine PAA-Aggregation durchgeführt wurde – die Länge der SAX-Sequenz entspricht hier also der vollen Zeitreihenlänge. Werte fern des Mittelwertes können zwar nicht genau rekonstruiert werden, die Verlaufscharakteristik bleibt jedoch bei einer Alphabetgröße von 16 und ohne zeitliche Aggregation erhalten.



**Abbildung 4.10:** ViAT-Kurvendarstellung der in den EDR-Merkmalmetadaten enthaltenen SAX-Repräsentation.

Die aufgrund der datenparallelen Verarbeitung sehr effizient erzeugbaren Metadaten ermöglichen also die Erstellung einer lokalen Kopie von datenreduzierten Messdaten über den Gesamtzeitraums einer Messung. Die Messdauer ist dabei aufgrund der parametrisierbaren Aggregation im Prinzip nicht beschränkt. Im Falle der Metadaten für EDR-Merkmale können Metadaten über den gesamten Messzeitraum von über drei Jahren problemlos lokal gespeichert werden. Die Metadaten eines Messtages mit zwei EDR-Messgeräten über alle Kanäle enthalten 24 Dateien und besitzen eine Gesamtgröße von 3,88 MB, etwa 0,75 % der Gesamtgröße von 518,4 MB Gesamtgröße der ursprünglichen EDR-Merkmalmetadaten. Eine einzelne EDR-Merkmalmetadaten-datei besitzt eine Größe von nur ca. 15 KB, abhängig von der Charakteristik der enthaltenen Zeitreihen, da die RLE-Codierung der SAX-Sequenzen monotone Abschnitte zusammenfasst und die Dateien damit kleiner werden.

Aufgrund der MapReduce-Verarbeitung liegt jeder Metadatenatz für jede Merkmalsdatei als Ordner vor, der neben der eigentlichen Metadatenfile („*part-r-00000*“ in Abbildung 4.11) auch eine CRC-Prüfsumme und Logdateien zur Erzeugung enthält. Die Metadaten über einen größeren Messzeitraum können komplett unabhängig von der verteilten Cluster-Infrastruktur in *ViAT* visualisiert und analysiert werden.



Name	Größe
_logs	
.part-r-00000.crc	1 KB
part-r-00000	15 KB

**Abbildung 4.11:** Datenstruktur eines einzelnen datenparallel erzeugten Metadatenatzes über eine EDR-Merkmalsdatei der dritten Spannungsphase.

Die datenparallele Erzeugung der Metadaten für die EDR-Merkmalsdaten eines Tages von drei EDR-Geräten (es waren in diesem Fall 39 Dateien und eine Gesamtgröße von ca. 846,3 MB) dauerte auf dem Hadoop-Cluster der LSDF etwa fünf Minuten. Eine Hochrechnung ist nicht ganz korrekt aufgrund der starken Schwankungen der Laufzeiten (siehe Abs. 3.2) und der datenparallelen Erzeugung, die höchstwahrscheinlich schneller abläuft, reicht jedoch für eine grobe Vorstellung der Obergrenze der Verarbeitungsdauer aus. Die Metadatenerzeugung für EDR-Merkmalsdaten aus Messungen über ein Jahr mit drei Messgeräten würde demnach ca. 30 Stunden und 25 Minuten dauern.

## 4.6 Visualisierung

**Skalierbarkeit:** Das Visualisierungsmodul *ViAT* wurde hinsichtlich der Skalierbarkeit mit dem Datenvolumen bzw. der Anzahl an Datenpunkten darzustellender Zeitreihendaten evaluiert. Hierzu wurden mehrere Testdatensätze unterschiedlicher Größe erzeugt, die jeweils sieben Zeitreihen enthielten. Die Testdaten wurden gemäß des in Abs. 5.1.2 dargestellten XML-Formats für EDR-Rohmessdaten auf einer lokalen Festplatte eines Client-Rechners<sup>34</sup> gespeichert. Ausgehend von einem Datensatz mit 300 005 Zeitschritten und einer Dateigröße von 5.53 MB (5799936 Bytes Festplattenspeicher) wurde die Größe bzw. die Anzahl an Datenpunkten (mit `double`-Werten)

<sup>34</sup> Es handelt sich um folgendes PC-Model: HP Z230 Workstation, Prozessor: Intel Core i7-4790 3.6GHz 4C TC, RAM: HP 32GB DDR3-1600 nECC (4x8GB) TC WS, FP: HP 1TB 7200 RPM SATA 1st HDD TC WS.

schrittweise verdoppelt<sup>35</sup>, um eine ungefähre Grenze der darstellbaren Datenmenge zu ermitteln. Alle Maßnahmen in *ViAT* zur Datenreduktion, wie z. B. das Subsampling beim Lesen der Daten, wurden zu diesem Zweck deaktiviert. Jeder Datensatz wurde in *ViAT* geladen und visualisiert, dabei wurde das Verhalten bei Interaktionen wie dem Zoomen und Navigieren in den Daten beobachtet und festgehalten. Die einzelnen Datensätze und das jeweils entsprechende Verhalten von *ViAT* werden in Tabelle 4.4 aufgeführt.

**Tabelle 4.4:** Test mit lokalen Testdatensätzen zur Evaluation des Visualisierungsmoduls *ViAT*. Quelle: Eigene Messung.

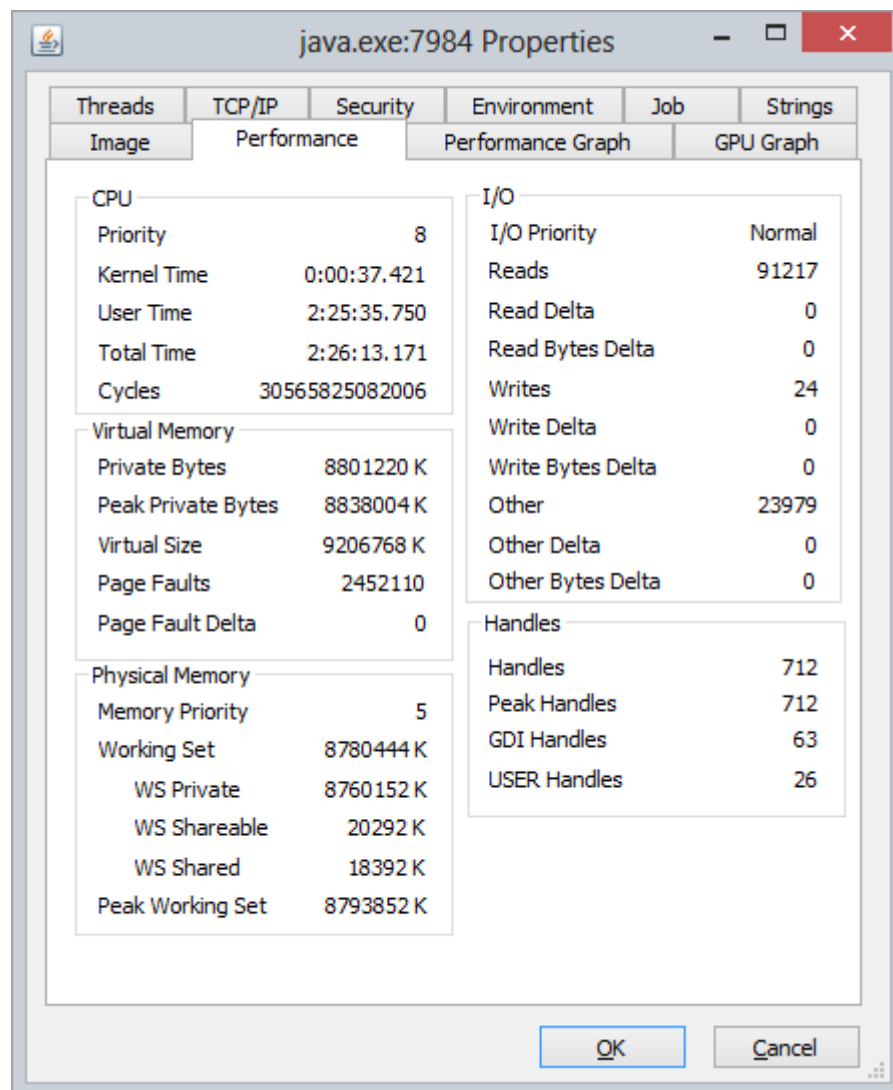
<i>ID</i>	<i>Anzahl an Zeitschritten</i>	<i>Gesamtzahl an Datenpunkten</i>	<i>Volumen in KB</i>	<i>Verhalten von ViAT</i>
<i>X_1</i>	300 005	2 100 035	5 661	Flüssig
<i>X_2</i>	600 010	4 200 070	11 322	Flüssig
<i>X_4</i>	1 200 020	8 400 140	22 643	Flüssig
<i>X_8</i>	2 400 040	16 800 280	45 285	Flüssig
<i>X_16</i>	4 800 080	33 600 560	90 569	Flüssig
<i>X_32</i>	9 600 160	67 201 120	181 137	Flüssig mit kurzen Blockaden
<i>X_64</i>	19 200 320	134 402 240	362 274	Rhythmisch verzögerte Interaktion
<i>X_128</i>	38 400 640	268 804 480	724 546	Java Exception beim Laden <sup>36</sup>

Wie aus der Tabelle hervorgeht, sind also rund 134 Millionen Datenpunkte und rund 363 MB noch ohne größere Beeinträchtigungen der Interaktion darstellbar. Schnelle Interaktionsmöglichkeiten mit sofortiger visueller Rückmeldung sind von entscheidender Bedeutung für die Effektivität der Exploration. Deshalb ist eine statische Visualisierung, mit welcher die Anzeige einer grö-

<sup>35</sup> Die vom Betriebssystem angezeigte Dateigröße nahm bei der jeweiligen Verdopplung der Anzahl an Datenpunkten geringfügig langsamer zu als die Anzahl der Datenpunkte, was auf einen volumenunabhängigen Datenanteil konstanter Größe zurückzuführen ist, welchen jede der Dateien enthält.

<sup>36</sup> Konsolenausgabe: Exception in thread "AWT-EventQueue-0" java.lang.OutOfMemoryError: GC overhead limit exceeded.

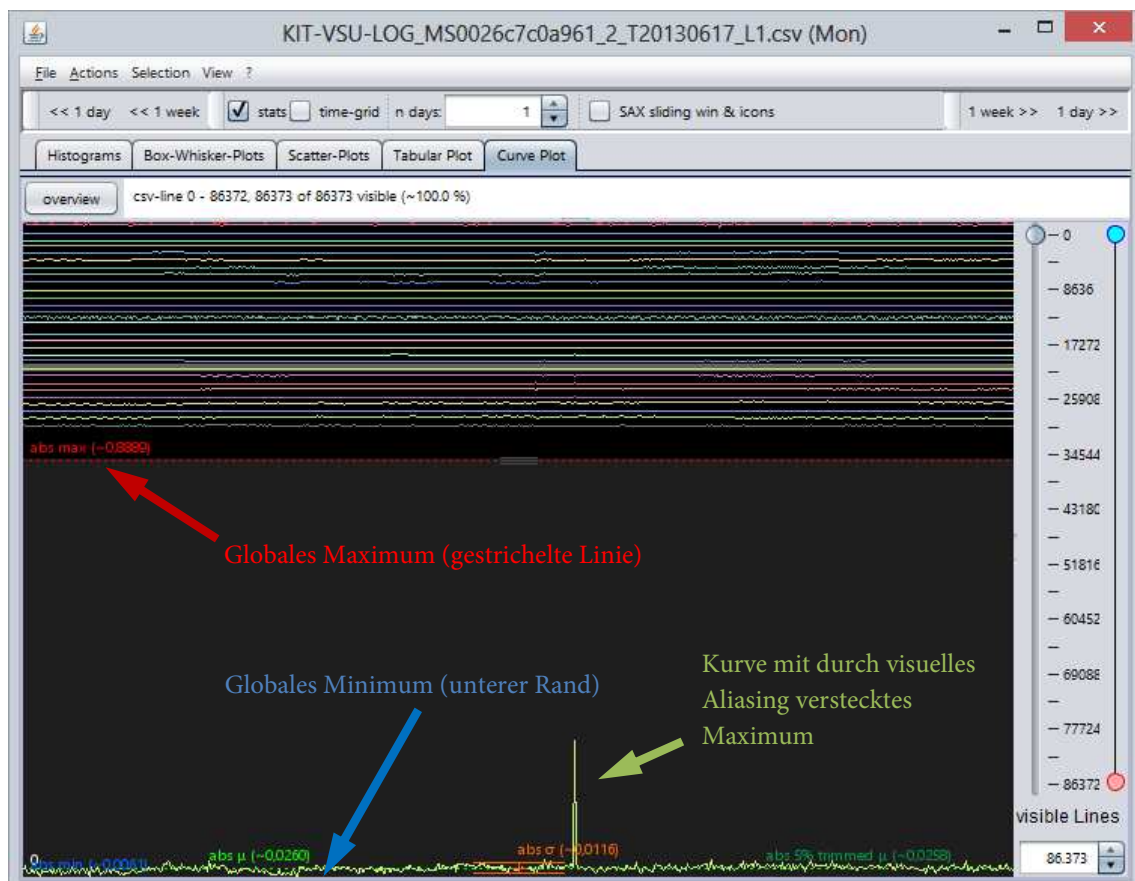
ßeren Anzahl an Datenpunkten möglich ist, keine Option. Der limitierende Faktor scheint hauptsächlich der Arbeitsspeicher zu sein, wobei der physikalische Speicher des verwendeten Rechners (32 GB) bei der Anzeige von *X\_64* nur zu etwa 27 % ausgelastet ist (siehe Anzeigebereich unten links in Abbildung 4.12). Um festzustellen, ob beim Laden der Daten direkt aus HDFS abweichendes Verhalten beobachtet werden kann, wurden die Testdatensätze *X\_32* und *X\_64* in HDFS geladen. Die Visualisierung wurde anschließend über den HDFS-Explorer gestartet. Das Interaktionsverhalten von *ViAT* wies keine merklichen Unterschiede auf, falls die Daten direkt aus HDFS geladen wurden.



**Abbildung 4.12:** Performanzindikatoren bei Anzeige von *X\_64* in *ViAT* (Bildschirmabzug aus dem Prozessmanager *ProcessExplorer* von Sysinternals).

Es bleibt zu betonen, dass in *ViAT* weitaus größere Datenmengen als *X\_64* dargestellt werden können, indem die besonderen Mechanismen genutzt werden, die beispielsweise für die Anzeige von EDR-Daten automatisch eingesetzt werden. Sollen etwa EDR-Merkmalen Daten dargestellt werden, die sich über mehrere Tage und damit über mehrere Datensätze erstrecken, so wird ein Subsampling beim Lesen der Werte angewandt, das automatisch mit der Anzahl an Datensätzen skaliert wird. Somit bleibt die Anzahl darzustellender Datenpunkte unabhängig von der Datenmenge, zu Lasten der Detailtreue. Das wird jedoch in Kauf genommen, da bei der Anzeige großer Zeitbereiche davon auszugehen ist, dass sich der Benutzer einen Überblick über Charakteristik und Verlauf der Daten verschaffen möchte. Bei Bedarf kann ein kleiner Zeitbereich, welchen der Nutzer genauer untersuchen möchte, ohne Subsampling geladen werden, um durch Zoomen kleinste Details bis hin zu den Einzelwerten erkennen zu können.

**Zeitreihen-Aliasing:** Eine Unterabtastung von Zeitreihendaten kann sich auf die Visualisierung auswirken. Besonders problematisch ist dabei, dass die Sichtbarkeit kurzer Ereignisse nicht mehr gewährleistet ist, wenn mehr Zeitschritte einer Zeitreihe dargestellt werden sollen als Pixel zur



**Abbildung 4.13:** Beispiel einer Unterabtastung bei der Kurvendarstellung in *ViAT*, bei der ohne Darstellung von Zeitbereichsstatistiken kurze Ereignisse verborgen bleiben, was dadurch ersichtlich ist, dass die Kurve nicht an das absolute Maximum (rote gepunktete Linie) anschließt.

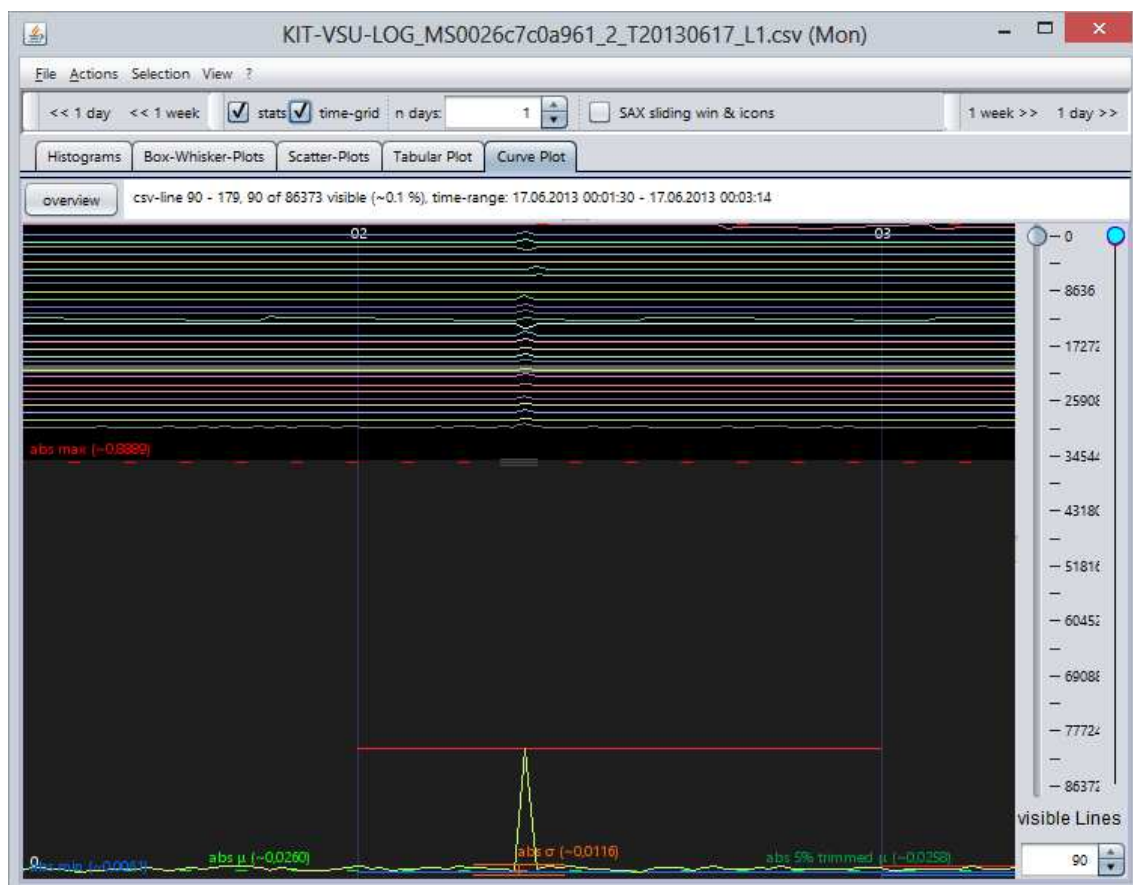


Verfügung stehen. Um diesem Problem, das zwangsläufig bei der Überblicksdarstellung großer Zeitreihendaten auftritt, zu begegnen, wurden in *ViAT* zeitliche Aggregat-Statistiken eingeführt. Im Folgenden wird evaluiert, wie sie dazu beitragen, kurze prägnante Ereignisse in Zeitreihendaten sichtbar zu machen, die in herkömmlichen Visualisierungen verborgen bleiben oder durch aufwendige Berechnungen für die jeweilige Ansicht detektiert und gesondert visualisiert werden müssen. Abbildung 4.13 zeigt ein Beispiel einer Kurvendarstellung von Messdaten über einen Zeitraum von 24 Stunden mit Unterabtastung, wobei kurze Ereignisse verborgen bleiben, die erst durch eine entsprechende Vergrößerung sichtbar werden. Um jene Ereignisse, die von enormer Bedeutung bei der Einschätzung der Kurvencharakteristik sein können, dennoch sichtbar zu machen, können in *ViAT* zusätzliche Aggregatstatistiken angezeigt werden, die auf extreme Werte innerhalb eines Zeitbereichs hinweisen. Sie werden ähnlich dargestellt wie die globalen Statistiken, z. B. das in Abbildung 4.13 gestrichelt dargestellte globale Maximum (rot) und Minimum (blau). Zur Veranschaulichung ist in Abbildung 4.14 dieselbe Kurvendarstellung zu sehen, wobei hier ein zusätzliches Zeitgitter mit Zeitbereichsstatistiken für jede Stunde eingblendet wurde. Durch die durchgezogenen roten Linien, welche das Maximum jedes Zeitbereichs markieren,



**Abbildung 4.14:** Beispiel einer Unterabtastung bei der Kurvendarstellung in *ViAT* mit verborgenden lokalen Maxima, die durch Einblendung von Zeitbereichsstatistiken sichtbar werden (z.B. bei blauem Pfeil).

wird sofort visuell ersichtlich, dass in einzelnen Bereichen hohe Werte vorkommen, die nicht in der Kurvendarstellung erkennbar sind. Z. B. wird ersichtlich, dass bereits zu Beginn der Kurve vergleichsweise hohe Werte vorkommen, die in den darauffolgenden elf Stunden nicht vorkommen (markiert durch einen blauen Pfeil). Der Nutzer hat daraufhin die Möglichkeit, entsprechende Bereiche ausreichend zu vergrößern, um die interne Dynamik sichtbar zu machen. Das Zeitgitter verhält sich dabei adaptiv. Es passt sich zusammen mit den entsprechenden Bereichsstatistiken dem jeweils sichtbaren Zeitausschnitt an, so dass ab einer bestimmten Vergrößerung automatisch ein minütliches Raster mit entsprechenden Statistiken angezeigt wird, wodurch auffällige Bereiche hierarchisch bis zu den Einzelwerten verfolgt werden können. Abbildung 4.15 zeigt eine vergrößerte Ansicht der zweiten Minute der ersten Stunde derselben Kurve, welche den klar erkennbaren Ursprung des aufgefallenen stündlichen Maximums zu Beginn der Zeitreihe enthält.



**Abbildung 4.15:** Vergrößerter Ausschnitt der zweiten Minute der ersten Stunde der Beispielkurve mit klar identifizierbarer Herkunft des zuvor aufgefallenen stündlichen Maximums.

## 4.7 Zusammenfassung

In diesem Kapitel wurden die neu entwickelten Verfahren zur explorativen Analyse großer Zeitreihendaten und ihr Zusammenspiel als Analysesystem anhand verschiedener Nutzungsszenarien experimentell evaluiert. Hierzu wurde zunächst deren Integration als interagierende Softwaremodule zu einem skalierbaren Analyse-Framework namens *FraScaTi* präsentiert. Es wurde aufgezeigt, wie die einzelnen interaktiven Visualisierungsansichten automatisch untereinander synchronisiert werden, um stets dieselben Teilbereiche der Zeitreihendaten anzuzeigen, so dass jederzeit zwischen den Darstellungsvarianten gewechselt werden kann und welche Vorteile das für eine explorative Analyse bringt. Außerdem wurde dargestellt, wie die neu entwickelten Analyseverfahren aus den Visualisierungsansichten heraus direkt auf dort selektierte Teilbereiche angewandt und die Analyseergebnisse wiederum in der Visualisierung angezeigt werden können. Anschließend wurde die Funktions- und Leistungsfähigkeit der einzelnen Verfahren für typische explorative Analyseaufgaben anhand von Testdaten nachgewiesen.

Für das Subsequenz-Mining mit dem neu entwickelten *SAX-Sequenzeditor* wurde gezeigt, dass dieses Verfahren neue, nützliche Analysemöglichkeiten auf Basis der effizient verteilt vorberechenbaren Metadaten (hier dem Teil der SAX-Repräsentation) bietet. Insbesondere ermöglicht es eine schnelle interaktive Suche nach zuvor unbekanntem Mustern im Bereich des *Subsequenz-Mining*, indem es Nutzern erlaubt, Zeitreihenmuster nur teilweise interaktiv zu definieren und eine sofortige Rückmeldung aller verbleibenden vorkommenden Muster mit Farbcodierung ihrer relativen Auftrittshäufigkeiten zu erhalten. Somit kann sowohl das Auftreten bekannter Muster schnell überprüft werden als auch auf einfachem Wege neue häufige und seltene Muster sowie das Nichtauftreten von Mustern herausgefunden und gleichzeitig visualisiert werden. Durch einen Vergleich mit dem einzigen ähnlichen Verfahren namens *VizTree* [179] wurden die Stärken und Schwächen der Anwendbarkeit herausgestellt, die v. a. darin bestehen, dass mit dem *SAX-Sequenzeditor* für vergleichsweise längere Teilsequenzen eine übersichtliche Visualisierung der Auftrittshäufigkeiten möglich ist. Des Weiteren ermöglicht die Einbettung modifizierter SAX-Sequenz-Histogramme (siehe Abs. 3.7.5) eine Übersicht über die Häufung von Sequenzen unterschiedlicher Länge auf einen Blick.

Zur *Detektion von Discords und Motifs* wurde gezeigt, dass der modifizierte Algorithmus *SortList* eine geringere Laufzeitkomplexität aufweist und dadurch im Vergleich zur Variante *HOTSAX* [143] wesentlich schneller arbeitet (ca. um Faktor 1200), da aufwendige Berechnungen vermieden werden, welche gemäß [143] ca. 99 % der Laufzeit ausmachen. Die Ergebnisse, insbesondere gefundene Discords, können sich in manchen Fällen leicht unterscheiden, die Leistungsfähigkeit in Bezug auf die Fundstellen ist jedoch vergleichbar. Mit dem eingeführten Algorithmus *SortList* ist somit erstmals eine automatische Detektion von Discords und Motifs in großen Zeitreihendaten

möglich, die ausreichend schnell Ergebnisse liefert, um in eine interaktive Visualisierung eingebettet werden zu können.

Die Strukturanalyse *MetaSAX* erlaubt eine neue, komprimierte und lesbare Strukturbeschreibung großer Zeitreihendaten auf abstraktem Niveau. Sie kann als Teil der Zeitreihenmetadaten gespeichert werden und liefert direkte semantische Aussagen über die Zeitreihe, etwa über einen möglicherweise vorhandenen Trend (zunehmend oder abnehmend), eine Verschiebung im Wertebereich (positiv oder negativ sowie Anzahl), das Auftreten typischer Teilverläufe (z. B. Zylinderform, Hügel, Tal etc.) und deren Anzahl, sowie Stellen, an denen sich der Verlauf stark ändert (sog. Changepoints). U. a. können diese Merkmale bei der Ähnlichkeitsbewertung von Zeitreihen verwendet werden, die für viele Analyseaufgaben (z. B. Klassifizierung, Clustering, inhaltsbasierte unscharfe Suche) benötigt wird.

Für die datenparallel auf dem Hadoop-Cluster erzeugten Metadaten, auf denen die neuen explorativen Analyseverfahren basieren, wurde die Performanz der verteilten Berechnung aus den Rohdaten aufgezeigt. Die hohe Kompressionsrate (ca. 0,75 % gegenüber Rohdaten) der symbolischen Codierung erlaubt eine Visualisierung und explorative Analyse auf einem lokalen Arbeitsplatz-Rechner für große Zeitbereiche (in der Größenordnung von mehreren Messjahren bei sekundlichen Messwerten). Für vergleichbare Zeitbereiche können Zeitreihen-Rohdaten aufgrund ihrer Datenmenge bisher nur verteilt gespeichert und analysiert werden.

Die Visualisierungsmöglichkeiten und die Nützlichkeit der interaktiven Bedienung zur Navigation, Selektion, Kommentierung und Fehlervisualisierung der unterschiedlichen Visualisierungen des Visualisierungsmoduls *ViAT* und ihre Skalierbarkeit mit großen Datenmengen wurden gezeigt. Die Visualisierung ist hochperformant, wobei die gleichzeitig visualisierbare Datenmenge vom verfügbaren Arbeitsspeicher determiniert wird. Durch eine Steuerungslogik können bei Bedarf jedoch jederzeit entfernte Daten nachgeladen werden. Die implementierten statistischen Lösungsansätze für das Problem des visuellen Aliasing ermöglichen eine Darstellung großer Mengen an Datenpunkten, deren Anzahl die zur Verfügung stehenden Pixel übersteigt. Dabei verhindern die eingeführten adaptiven Zeitbereichstatistiken, dass entscheidende Information ausgeblendet wird.

# 5 Anwendung des neuen Analyseframeworks zur Untersuchung umfangreicher Stromnetz-Messdaten

In diesem Kapitel wird ein Anwendungsfeld näher beschrieben, in dem das vorgestellte Framework für die skalierbare Verwaltung und Analyse großer Zeitreihendaten (*FraScaTi*) gewinnbringend angewandt wurde. Es handelt sich um den Bereich der Stromnetzanalyse, die eine kritische Voraussetzung für geplante Anpassungen der Netzinfrastruktur an zukünftige Herausforderungen darstellt. In den kommenden Jahren werden durch großflächige Stromnetzanalysen große Zeitreihendaten aus synchronisierten Strom- und Spannungsmessungen anfallen, die mit neuen, skalierbaren Systemen verwaltet und ausgewertet werden müssen. Bislang sind hierzu nur wenige Verfahren bekannt, die in skalierbaren Analysesystemen einsetzbar sind, weshalb das *FraScaTi*-System einen Beitrag dazu leisten kann, Stromdaten bisher nicht verarbeitbaren Umfangs explorierbar und analysierbar zu machen.

## 5.1 EDR-Strommessung am KIT

Im Folgenden werden die EDR-Geräte und die Datenformate erklärt, welche für die Speicherung von EDR-Stromnetz-Messdaten und abgeleiteten Merkmalsdaten zum Einsatz kommen. Die hochfrequent Strom und Spannung messenden EDR-Geräte stellen Vorarbeiten des IAI am KIT im Rahmen der Auswertung umfangreicher Stromnetz-Messdaten dar. Sie sollen einen Beitrag zu einem besseren Verständnis der komplexen transienten Dynamik im Stromnetz leisten. Die Formate und die Erzeugung der resultierenden Daten werden in den folgenden Abschnitten näher beleuchtet, da nachfolgend das vorgestellte *FraScaTi*-System für skalierbare Zeitreihenanalysen auf diese Daten angewandt wird.

### 5.1.1 Elektrischer Daten-Rekorder (EDR) und synchronisierte verteilte Strommessungen

Um Spannung und Strom für alle drei Phasen und den Neutralleiter hochfrequent auf allen Spannungsebenen synchronisiert und langfristig erfassen zu können und so auch transiente Effekte über lange Zeiträume festzuhalten, wurde am IAI des KIT ein Gerät mit dem Namen „*Elektrischer Daten Recorder*“ (EDR) entwickelt. Die Geräte bestehen aus einem DAQ Board mit 8 Kanälen,

einem GPS-Empfänger und einem Notebook mit Netzverbindung. Der Analog-Digital-Konverter dient der Überführung der kontinuierlichen Spannungs- und Stromsignale in ein quantisiertes digitales Signal, dessen Einzelwerte jeweils mit einem aktuellen Zeitstempel versehen und auf dem dedizierten Notebook gespeichert werden. Der daran angeschlossene GPS-Empfänger stellt weitere Angaben zum aktuellen Ort und der aktuellen Zeit bereit, welche zusammen mit dem GPS-PPS-Signal<sup>37</sup> ebenfalls in die Messdaten eingehen. Eine am IAI entwickelte Software sammelt die Daten lokal und überträgt diese über einen gesicherten Kanal über das Netzwerk an einen Server, von wo aus die Daten an Archivierungs- und Analysensysteme weitergeleitet werden.

Die Abtastfrequenz kann zwischen 10 kHz und 25 kHz gewählt werden und ist damit gegenüber der Netzfrequenz von 50 Hz relativ hoch. Die hier untersuchten Daten resultieren aus Messungen mit einer Messfrequenz von 12,8 kHz, so dass für eine Sinus-Periode einer 50 Hz Wechselspannung genau 256 Einzelwerte erfasst werden. Diese werden in digitale 16 Bit Gleitkommawerte überführt und mittels Base64<sup>38</sup> zu einer ASCII-Sequenz codiert, welche zusammen mit Metainformationen zur Messung in einer XML-Datei gespeichert wird. Zu den festgehaltenen Metainformationen gehören beispielsweise auch der via GPS ermittelte Messort sowie der Zeitpunkt der Messung aus unterschiedlichen Quellen, um einen zeitsynchronen Vergleich zwischen verschiedenen Messstationen zu ermöglichen. Das vom GPS-Modul bereitgestellte PPS-Signal (von engl. *Pulse per second*) ermöglicht hierbei eine exakte, hochaufgelöste ( $\pm 1\mu\text{s}$ ) Synchronisationsmöglichkeit, die durch weitere Zeitangaben ergänzt wird, etwa der Systemzeit des Notebooks und der über NTP<sup>39</sup> Protokoll ermittelten Zeit. Dies ermöglicht im Falle einer Störung einer der Zeitquellen eine Möglichkeit zur Rekonstruktion bzw. Korrektur der Zeitstempel. Als Zeitskala wird die *koordinierte Weltzeit* UTC<sup>40</sup> verwendet. Die in dieser Arbeit untersuchten Daten stammen aus Messungen mit zwei EDR-Geräten am KIT Campus Nord, welcher ein inselartiges elektrisches Verteilnetz besitzt, sowie aus Messungen mit einem weiteren EDR-Gerät am ca. 12 km entfernten

---

<sup>37</sup> PPS steht für *1 Puls pro Sekunde* und wird in GPS für eine präzise Sekundentaktung verwendet. Es stellt den Beginn und das Ende einer Sekunde als ansteigende oder abfallende Flanke dar.

<sup>38</sup> In den Vorgaben der MIME- (von engl. *Multipurpose Internet Mail Extensions*) Protokollerweiterung des SMTP Protokolls für Email wird *Base64* verwendet, um Binärdaten als Text zu kodieren und somit direkt in Texte einbetten zu können. Die Base64-Codierung in MIME stützt sich auf eine in RFC 1421 spezifizierte Version von PEM (von engl. *Privacy-enhanced Electronic Mail*) und verwendet ein Alphabet mit 64 Zeichen.

<sup>39</sup> NTP steht kurz für engl. *Network Time Protocol*, einem paketvermittelten Protokoll zur zeitlichen Synchronisation von Computern über das Netzwerk.

<sup>40</sup> UTC steht kurz für engl. *Coordinated Universal Time*, und ist die seit 1972 gültige *Weltzeit*. Die *Mittel-europäische Zeit* (MEZ) geht der UTC um eine Stunde voraus. Die UTC nutzt den Sekundentakt der internationalen Atomzeit TAI und sieht eine Verwendung sog. Schaltsekunden vor, um Schwankungen der Erdrotation zu begegnen. Hierin unterscheidet sich UTC von der sog. Universalzeit (UT), die Erdrotationsschwankungen durch Anpassung der Zeiteinheiten kompensiert.

KIT Campus Süd, nahe des städtischen Versorgungsnetzes der Stadt Karlsruhe. Nähere Details zu den EDR-Geräten und den durchgeführten Messungen finden sich z. B. in [18], [47], [192], [193], [195].

### 5.1.2 EDR-Messdaten

Die EDR-Messgeräte schreiben die Rohdaten minütlich in Form von XML-Dateien auf die Festplatte eines per USB angeschlossenen Notebooks. Am Tagesende werden die Daten auf den Hadoop-Cluster übertragen. Jede XML-Datei enthält Messwerte in 60 Binärdatenblöcken in *Base64*-Codierung (siehe Code 5.1) für alle drei Phasen von Spannung und Strom sowie den Strom des Neutralleiters.

```
...
<DAQ-SerNr>Omega OM-USB-1608FS.50</DAQ-SerNr>
<samplesPerSecond>5000</samplesPerSecond>
<evaluationVersion>2.1</evaluationVersion>
<dataRepresentation>2byte,base64</dataRepresentation>
<byteorder0x123456><![CDATA[VjQS]]></byteorder0x123456>
<channelCount>8</channelCount>
<captureType>44</captureType>
</acquisition>
<samplesSinceLastStartIndex channel="L1">1.24107
</samplesSinceLastStartIndex>
<data channel="La" samples="5000" scaleFactor="42.3053" unit="V">
  <![CDATA[TTe3MvwuYivQKK4mNCUrI8whCiDuHtYd9hx ...
```

**Code 5.1:** Beispiel für einen Daten-Block (rot hervorgehoben, gekürzt) einer EDR-Messwert-Datei.

Jeder Block umfasst also eine Messdauer von einer Sekunde. Um Messdaten über mehrere EDR-Geräte an unterschiedlichen Messorten synchronisiert zu erfassen und so vergleichbar zu machen, wird jeder Datenblock durch zwei zeitlich hoch aufgelöste PPS-Signale ( $\pm 1\mu\text{s}$ ) bestimmt.

```

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!-- KIT-Scope Raw measurements -->
<!-- Version 1.2 - 23.02.2012 -->
<!-- Karlsruhe Institute of Technology (KIT), Germany -->
<!-- Contact: heiko.maass@kit.edu -->
<!-- File created: 17.06.2013 13:21:00 -->

<KIT_SCOPE_RAWDATALOG>

  <VSU_RAW_DATA>
    <sys_datetime>2013-06-17ZT13:21:00</sys_datetime>
    <sys_datetime_ms>000</sys_datetime_ms>
    <MAC>00-26-c7-c0-a9-61</MAC>
    <unit>EDR0006</unit>
    <NTP>
      <offset>1242.32</offset>
      <deviation>0.862485</deviation>
      <reliability>4</reliability>
    </NTP>
    <GPS>
      <gps_datetime>2013-05-15ZT11:22:09</gps_datetime>
      <gps_datetime_ms>130</gps_datetime_ms>
      <timeValid>1</timeValid>
      <dateValid>15</dateValid>
      <posValid>1</posValid>
      <ppsValid>1</ppsValid>
      <satNumber>7</satNumber>
      <precision>1.1</precision>
      <qualityID>2</qualityID>
      <coordinate-notation>Dec:GG.xxxxx</coordinate-notation>
      <latitude>49.0959</latitude>
      <longitude>8.43335</longitude>
      <altitude>145.2</altitude>
    </GPS>
    <acquisition>
      <acq_datetime>2013-06-17ZT13:20:59</acq_datetime>
      <acq_datetime_ms>995,468</acq_datetime_ms>
      <sys_datetime>2013-06-17ZT15:20:59</sys_datetime>
      <sys_datetime_ms>904</sys_datetime_ms>
      <real_datetime>2013-06-17ZT13:21:01</real_datetime>
      <real_datetime_ms>145,068</real_datetime_ms>
      <realTimeUsedOffset>1.24107</realTimeUsedOffset>
      <realTimeReliability>4</realTimeReliability>
      <timesource>PPS,NTP,GPS</timesource>
      <PPSsync>1</PPSsync>
      <PPS Filter active="1" R="15010" C="0.022"> ...

```

**Code 5.2:** Beispiel für den Beginn einer EDR-Messwert-Datei im XML-Format mit Metainformationen zur Messung wie Zeit, Ort und Angaben zum Messgerät.



Daneben sind Metainformationen zur Messung enthalten wie EDR-ID, GPS-Daten zu Messzeit und -Ort, Detailangaben zur Codierung, zur Anzahl enthaltener Messwerte etc. (siehe Code 5.2). Außer den Metainformationen sind die XML-Daten nicht für den menschlichen Betrachter lesbar und müssen zunächst blockweise decodiert werden und in eine geeignete Datenstruktur geladen werden, um beispielsweise als zusammenhängende Zeitreihen visualisiert, weiterverarbeitet oder analysiert werden zu können.

Die Strom- und Spannungsmessungen werden seit Februar 2012 am Campus Nord des Karlsruher Instituts für Technologie (KIT) durchgeführt und vollständig gespeichert, um großskalige Auswertungen der detaillierten Daten zu ermöglichen. Das KIT besitzt ein weitgehend eigenständiges Stromnetz. Eine 110 kV Übertragungsleitung des örtlichen Energieversorgers und ein 2 MW Blockheizkraftwerk stellen die Stromquellen dieses inselartigen Netzes dar. Für zusätzliche Informationen über das Netz kann auf Verbrauchs-Messdaten von 534 auf dem gesamten Gelände verteilten und kontinuierlich messenden Smart Metern zurückgegriffen werden, welche jedoch keine hochfrequenten Messungen enthalten [194].

### 5.1.3 EDR-Merkmalsdaten

Auf den EDR-Geräten werden direkt Merkmale aus den Messdaten extrahiert und als separate Dateien in einem CSV-Format gespeichert (siehe Code 5.3). Jede CSV-Zeile enthält einen sekunden genauen Zeitstempel und die einzelnen Merkmale für die entsprechende Sekunde. Die Merkmale sind also zeitlich bereits stark aggregiert, meist durch Mittelung über mehrere Perioden.

```
Time;ms+;-;f50;Stabw f;Gleichspannung;Gleichrichtwert;Effektivwert;Scheitelfaktor;...  
17.06.2013 00:00:00;427;-13.793273;2.885583;0.074701;209.790688;233.118828;1.411723;...  
...
```

**Code 5.3:** Beispiel für den Beginn einer EDR-Merkmals-Datei (Zeilen gekürzt).

Eine Merkmalsdatei enthält die Merkmale eines Kanals bzw. einer Phase der EDR-Rohdaten über den Zeitraum eines Tages, also im fehlerfreien Fall 86400 Zeilen, was der Anzahl der Sekunden eines Tages entspricht, zusammen mit Bezeichnern der einzelnen Zeitreihen in der ersten Zeile. Die Merkmale beinhalten u. a. die Fundamentalfrequenz, den absoluten Phasenwinkel relativ zum Sekundenbeginn, Effektivwert (RMS), ARV (Average Rectified Value), Maximum und Minimum, die ersten dreizehn Harmonischen Schwingungen sowie die Gesamtverzerrung (kurz

THD, von engl. total harmonic distortion). Details zur Berechnung der Merkmale können aus Tabelle 5.2 entnommen werden. Mit \* markierte Werte stellen Kombinationen aus Spannung und Strom dar und werden nicht als Merkmals-Daten gespeichert. Im Folgenden werden die in den Dateien verwendeten Bezeichner für die einzelnen Merkmale verwendet, da sie ausgelesen und in ViAT verwendet werden.

**Tabelle 5.1:** Merkmalsbezeichnungen in den EDR-Merkmalsdaten des IAI und deren Bedeutung.

<i>Bezeichnung</i>	<i>Bedeutung</i>
<i>Time</i>	Zeitstempel, z. B. „17.06.2013 14:30:34“
<i>ms+-</i>	Zeitabweichung von der in „Time“ angegebenen Sekunde in Millisekunden.
<i>f50</i>	Abweichung der Frequenz von 50 Hz in Millihertz
<i>Stabw f</i>	Standardabweichung der Frequenz
<i>Gleichspannung</i>	Gleichspannung
<i>Gleichrichtwert</i>	Gleichrichtwert (ARV)
<i>Effektivwert</i>	Effektivwert (RMS)
<i>Scheitelfaktor</i>	Verhältnis von Scheitelwert zu Effektivwert
<i>Amplitude</i>	Amplitude (Scheitelwert)
<i>Abs. MaxV</i>	Absolutes Maximum der Spannung
<i>Stabw Ueff</i>	Standardabweichung des Effektivwerts der Spannung
<i>Stabw Umid</i>	Standardabweichung des Mittenwerts der Spannung (Mittel zw. Min und Max)
<i>Stabw Uss</i>	Standardabweichung der Spannung der Spitzenwerte
<i>THD</i>	Harmonische Gesamtverzerrung
<i>PhaseToPPS</i>	Phasenwinkel zum Zeitpunkt des PPS-Signals
<i>Periods</i>	Anzahl vollständig berücksichtigter Perioden (im Normalfall zw. 49 und 51)
<i>OW0 [%] bis OW14 [%]</i>	Harmonische 0 (OW steht für Oberwelle <sup>41</sup> ) bis Harmonische 14
<i>Temperature</i>	Temperatur (am Ort der Messung)

(Bezeichnung aus CSV-Datei übernommen)

<sup>41</sup> Die Bezeichnungen *Harmonische*, *Oberschwingung* und *Oberwelle* werden häufig gleichbedeutend für ganzzahlig vielfache Schwingungen der *Grundfrequenz* verwendet [128]. In der Fachliteratur wird jedoch genauer unterschieden. Die erste Harmonische stellt dort die Schwingung mit der Grundfrequenz dar (und wird *Grundschwingung* genannt), während die erste Oberschwingung die Schwingung mit verdoppelter Grundfrequenz meint, so dass die n. Harmonische der (n-1). Oberschwingung entspricht [30], [216].

Die Daten reichen wie die Rohdaten zurück bis Februar 2012 und beziehen sich auf Messungen am Campus Nord des Karlsruher Instituts für Technologie (KIT) [194].

**Tabelle 5.2:** Formeln zur Berechnung der sekundlichen EDR-Merkmale aus den hochfrequenten EDR-Messdaten (gemäß [194]).

<i>Merkmal</i>	<i>Formel</i>
<i>Frequenz</i>	$f = \frac{1}{T}$
<i>Gleichrichtwert</i> (En.: Average Rectified Value (ARV))	$X_{ARV} = \frac{1}{T} \int_{t_0}^{t_0+T}  x(t)  dt$
<i>Quadratisches Mittel</i> (En.: Root Mean Square (RMS))	$X_{RMS} = \sqrt{\frac{1}{T} \int_{t_0}^{t_0+T} x(t)^2 dt}$
<i>Versatz</i> (En.: Offset)	$X_{offset} = \frac{1}{T} \int_{t_0}^{t_0+T} x(t) dt$
<i>Maximum</i>	$X_{max} = \max_{\{t_0..t_0+T\}}(x(t))$
<i>Minimum</i>	$X_{min} = \min_{\{t_0..t_0+T\}}(x(t))$
<i>Amplitude</i>	$\hat{X} = \frac{(X_{max} - X_{min})}{2}$
<i>Scheitelfaktor</i> (En. Crest factor)	$C = \frac{X_{max}}{X_{RMS}}$
<i>Harmonische Gesamtverzerrung</i> (En.: Total harmonic distortion (THD))	$THD = \frac{\sqrt{(H_2)^2 + (H_3)^2 + (H_4)^2 + \dots + (H_n)^2}}{H_1}$
<i>Phase</i>	$\varphi = \frac{360}{T} * (t_{max} - t_0)$
<i>Harmonische 0 - 50</i>	$H_i = DFT_{8192} _{i=0..50}$

**Wirkleistung**

(En.: Active Power  $P$ )

$$P = \frac{1}{T} \int_{t_0}^{t_0+T} V(t) * I(t) dt$$

**Scheinleistung**

(En.: Apparent Power\* |S|)

$$S = V_{RMS} * I_{RMS}$$

**Gesamtblindleistung**

(En.: Reactive Power  $Q$ )

$$Q = \sqrt{S^2 - P^2}$$

5.1.4 Archivierung und Ordnerhierarchie der Messdaten

Die Messdaten aller EDR-Geräte werden zusammen mit den lokal vorberechneten Merkmalsdaten am Ende eines jeden Tages an einen Server übertragen und von dort aus im HDFS-Dateisystem des Hadoop-Clusters abgelegt. In den ersten Jahren des Messzeitraums wurden die Datenformate und die Ordnerhierarchie häufig geändert. Zunächst wurden alle Daten aller EDR-Geräte gemeinsam in einem Tagesordner abgelegt. Der Ordnername enthielt das Datum. Problematischerweise wurden in den Zeitangaben während der Zeit der Entwicklung des Messsystems manchmal führende Nullen bei Datumsangaben verwendet und manchmal nicht. Gleiches gilt für Zeitangaben, so dass es auch zu Kombinationen mit und ohne führende Nullen kam, was leider nicht dokumentiert wurde und bei Auswertungen berücksichtigt werden musste.

In einer ersten Version wurde eine flache Ordnerstruktur mit Tagesordnern für alle Dateien verwendet. Diese Struktur zeigte sich als ungünstig, da eine Auflistung eines Tagesordners aufgrund der großen Datenmenge viel Zeit in Anspruch nahm. Auf Anraten des Autors wurde eine veränderte Ablagestruktur eingeführt (siehe Abbildung 5.1), welche die Zeiteinheiten Jahr, Monat, Tag, Stunde etc. auf eine Ordnerhierarchie abbildet, so dass eine gezieltere ordnerweise Auflistung und Auswertung der Inhalte ermöglicht wird. Außerdem werden die Daten eines jeden EDR-Gerätes in separaten Ordnern gespeichert, welche nochmals nach Art der Daten untergliedert sind.

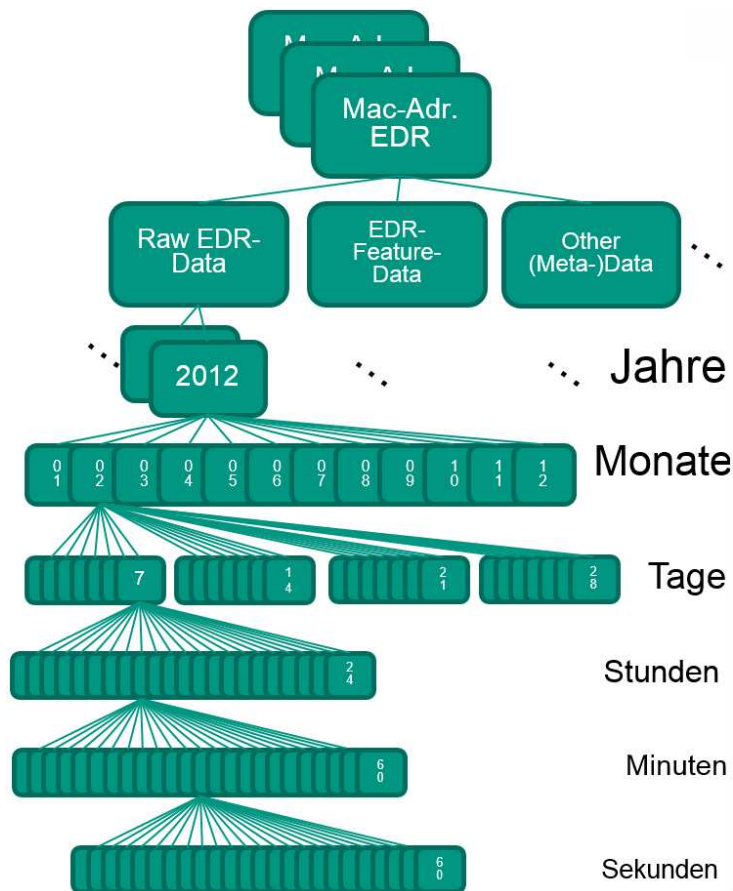


Abbildung 5.1: Neue Ordnerhierarchie zur Ablage der EDR-Daten.

## 5.2 Entwicklung eines Datenselektionswerkzeugs für HDFS-Daten

Vor jeder Verarbeitung von Daten – etwa in Form einer Analyse – muss stets eine Auswahl der zu verarbeitenden Daten getroffen werden. Für Zeitreihendaten kann das beispielsweise eine bestimmte Zeitspanne sein. Es kann aber auch nach anderen Merkmalen ausgewählt werden, etwa nach der Erfüllung bestimmter Kriterien, die sich je nach Datenart unterscheiden. Die Auswahl muss für jeden Verarbeitungsprozess durchgeführt werden und stellt somit eine häufig durchzuführende Aufgabe dar. Eine interaktive Auswahl kann je nach Struktur und Volumen der Daten und nach Anzahl auszuwählender Elemente und der Kombination von Auswahlkriterien schnell komplex bzw. zeitaufwendig werden und Spezialwissen über Infrastruktur und Anwendung erfordern. Speziell für die Zeitreihen in den EDR-Messdaten wurde im Rahmen dieser Arbeit daher

ein Auswahlwerkzeug für Anwender entwickelt, welches eine komfortable Selektion von Zeitreihendaten, insbesondere EDR-Daten, nach anwendungsspezifischen Kriterien erlaubt. Mögliche Kriterien sind z. B. der Wochentag der Messung, das eingesetzte Messgerät, enthaltene Namenszusätze in Datei- und Ordernamen, der Datentyp etc. Die Kriterien und zusätzliche Filter können über eine grafische Benutzeroberfläche (siehe Abbildung 5.2) festgelegt werden. Anschließend wird der Zeitbereich gewählt, in welchem die ausgewählten Kriterien angewandt werden sollen.

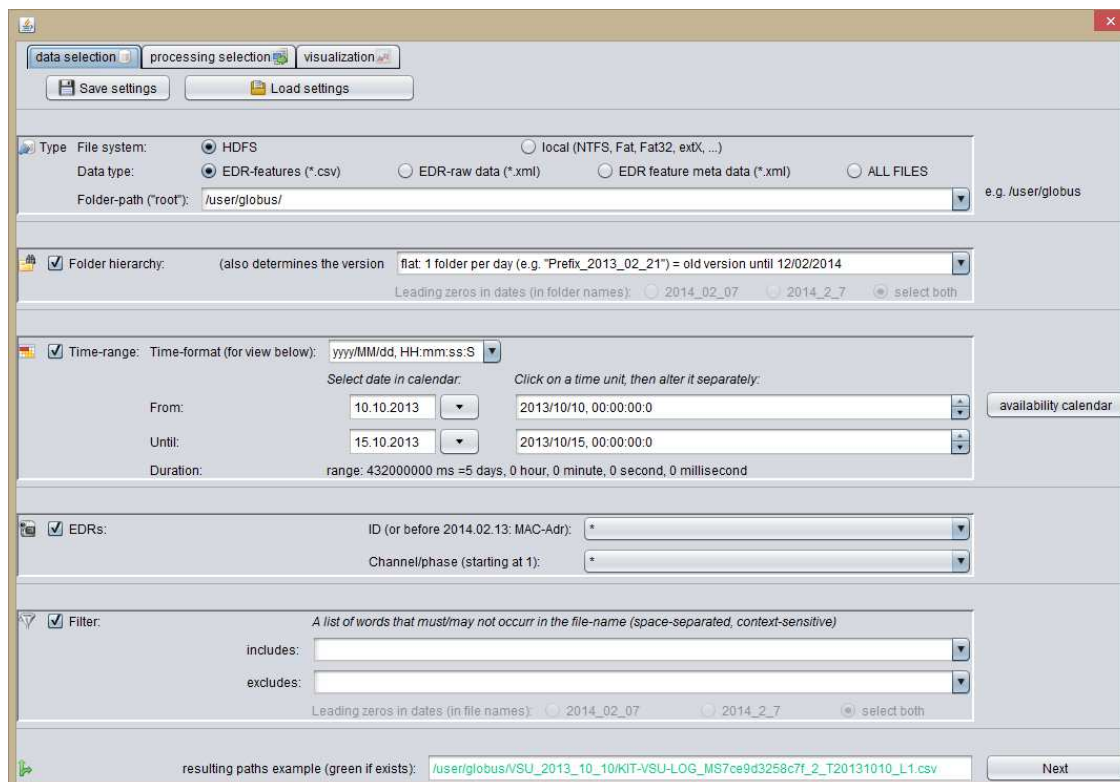
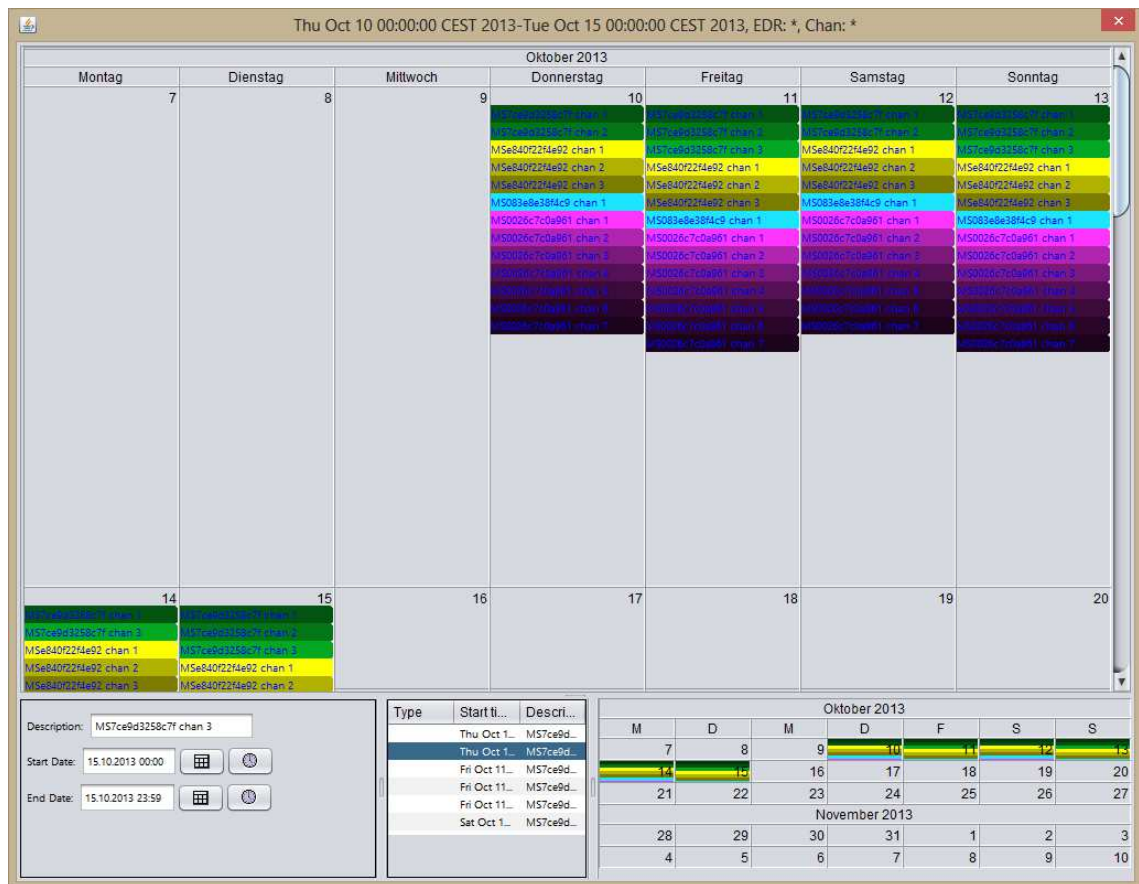


Abbildung 5.2: Werkzeug zur erweiterten EDR-Datenauswahl.

Da nahezu alle Analysen über längere Zeitspannen zusammenhängender, konsekutiver Zeitreihendaten erfolgen, wird ein Mechanismus benötigt, mit welchem die Vollständigkeit der Daten im Sinne eines Vorliegens aller benötigten Datensätze überprüft werden kann. Hierzu wurde im Rahmen dieser Arbeit eine kalenderbasierte Ansicht entwickelt, die in Abbildung 5.3 zu sehen ist und über den Button rechts neben der Auswahl des Zeitbereichs (in Abbildung 5.2) gestartet wird. Dem Nutzer wird ein Kalender für den gewählten Zeitbereich präsentiert, in welchem alle tatsächlich vorhandenen EDR-Dateien angezeigt werden, die den gewählten Kriterien entsprechen. Für jeden Tag ist ersichtlich, welches Messgerät welche Kanäle aufgezeichnet hat. Dabei stehen unterschiedliche Ansichten bereit: eine grafische Wochenansicht, eine Monatsansicht und eine Liste. Je nach gewähltem Zeitbereich und der Anzahl ausgewählter Dateien besitzt jede Ansicht ihre eigenen Vor- und Nachteile bei der Darstellung. In Abbildung 5.3 ist ein Auswahlzeitraum

von sechs Tagen zu sehen. Die verfügbaren EDR-Daten innerhalb der Zeitspanne werden für jeden Tag angezeigt, wobei Daten desselben Messgeräts die gleiche Farbe erhalten. Über die Farbhelligkeit können die einzelnen Kanäle desselben Geräts unterschieden werden.

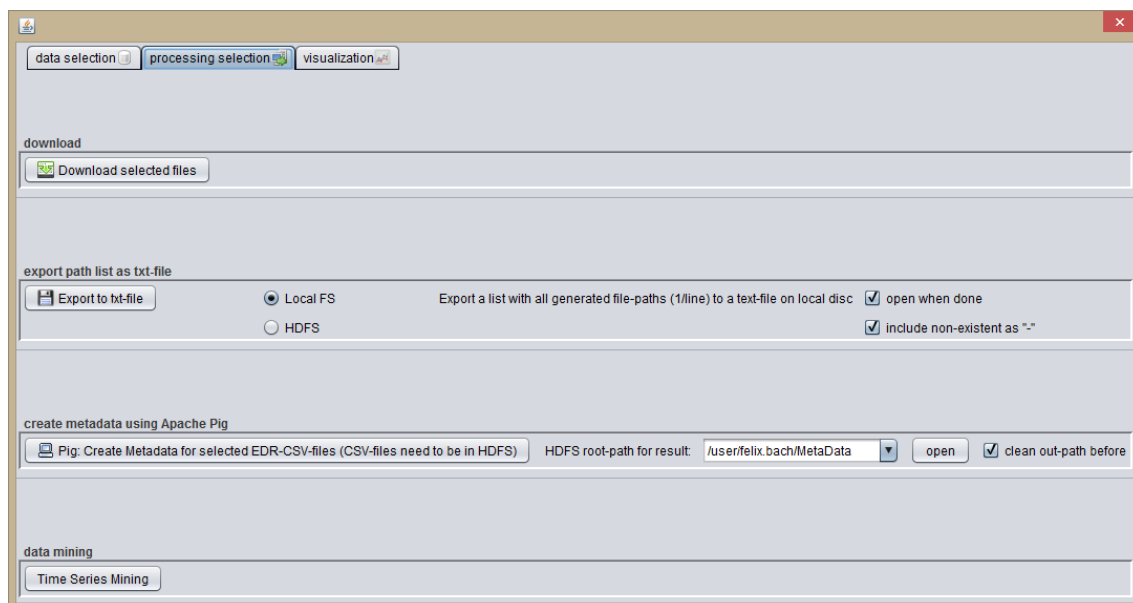


**Abbildung 5.3:** Kalender-Ansicht über einen Auswahlzeitraum von sechs Tagen zur Prüfung der Verfügbarkeit von EDR-Daten innerhalb der Zeitspanne, mit Farbcodierung der Messgeräte und Helligkeitscodierung einzelner Kanäle.

Sobald alle Kriterien festgelegt wurden und der Nutzer die tatsächliche Existenz aller Dateien für einen lückenlosen Zeitbereich überprüft hat, wird am unteren Ende der Eingabemaske in Abbildung 5.2 der Pfad zu einer Datei entsprechend der ausgewählten Kriterien erzeugt und angezeigt, wobei er grün erscheint, falls die entsprechende Datei existiert und andernfalls rot, so dass die Auswahlkriterien gegebenenfalls nochmals überprüft werden können. Wurde eine Auswahl getroffen, stehen verschiedene Möglichkeiten zur Weiterverarbeitung der gewählten Daten zur Verfügung, die über den Button „Next“ (im unteren Bereich von Abbildung 5.2) oder über die Wahl eines anderen Reiters im oberen Bereich erreicht werden. Es stehen zwei Reiter zur Verfügung:

Eine Oberfläche für das Herunterladen, den Export der Dateilisten, der datenparallelen Erzeugung von Metadaten mittels Pig und die Analyse der gewählten Daten (siehe Abbildung 5.4) sowie eine Oberfläche, über welche die ausgewählten Daten unterschiedlich visualisiert werden können.

Ein Export der Dateilisten selektierter Daten ist in vielen Fällen nützlich, in denen manuelle Auswertungen durchgeführt werden müssen, da so die Datenauswahl komfortabel über die Auswahloberfläche stattfinden kann und anschließend die Dateiliste zur Weiterverarbeitung in einer Kommandozeilen-Umgebung o. ä. genutzt werden kann. Des Weiteren kann optional auch eine Liste mit fehlenden Dateien oder eine Liste nur mit tatsächlich existierenden Dateien erzeugt werden. Die Erzeugung von Metadaten mittels Pig (siehe Abs. 3.4), inklusive der SAX-Codierung und statistischen Auswertung, kann über den „Pig...“-Button direkt veranlasst werden. Weitere häufig durchzuführende Analyseanwendungen können in die Oberfläche integriert werden, so dass sie direkt auf die gewählten Daten angewendet werden können und somit auch Nutzern zur Verfügung stehen, welche kein Spezialwissen über die zugrundeliegende verteilte Infrastruktur besitzen.



**Abbildung 5.4:** Dialog für die Weiterverarbeitung und Analyse ausgewählter Daten.

Für die Visualisierung der selektierten Zeitreihen-Dateien können einzelne oder alle Kanäle aus einer Auswahlliste gewählt werden. Je nach gewähltem Zeitbereich und Kanal wird automatisch eine passende Form der Visualisierung gestartet (siehe Abs. 3.3), welche auch die Darstellung von Zeitbereichen über Dateigrenzen hinaus ermöglicht.



## 5.3 Analyse der Stromnetz-Messdaten des KIT

In diesem Abschnitt wird eine konkrete Anwendung des *FraScaTi*-Systems zur skalierbaren explorativen Analyse im Bereich der Stromnetzanalyse vorgestellt. Hierzu werden die EDR-Stromnetz-Messdaten des KIT exploriert und analysiert. Das anwendungsspezifische Vorgehen wird diskutiert und es wird aufgezeigt, welche Vorteile und Besonderheiten sich durch den Ansatz ergeben. Ausgewählte Analyseergebnisse und ihre Bedeutung im Anwendungsfeld der Analyse von Stromnetz-Messdaten werden diskutiert.

### 5.3.1 Definition von Anwendungsfällen

Es wurden zwei Anwendungsfälle im Bereich der Beurteilung der Spannungsqualität definiert, für welche die bisher gemäß der Norm *DIN EN 61000-4-30* bzw. *VDE 0847-4-30* einzusetzenden Methoden und Verfahren keine befriedigenden Ergebnisse lieferten:

- F1: Detektion von Transienten in EDR-Rohdaten
- F2: Detektion kurzer Spannungseinbrüche in EDR-Merkmalen.

Im Folgenden werden beide kurz erklärt.

**F1: Detektion von Transienten in EDR-Rohdaten:** Hier wurden die EDR-Rohdaten aus hochfrequenten Spannungsmessungen nach Transienten (siehe Anhang B.2.2) durchsucht. Transienten sind schwer zu detektieren, da aktuelle Normen meist auf der Berechnung von Merkmalen beruhen, welche über mehrere Perioden gemittelt werden (wie z. B. in der Norm *DIN EN 61000-4-30* bzw. *VDE 0847-4-30*). Hierzu wurde die in Abs. 3.7.7 vorgestellte *SortList*-Methode zur Discord-Detektion eingesetzt. Zunächst wurden Testdaten aus EDR-Spannungsmessungen erzeugt und anschließend untersucht. Sie enthielten Transienten über sehr kurze Zeiträume mit jeweils unterschiedlicher Ausprägung.

**F2: Detektion kurzer Spannungseinbrüche in EDR-Merkmalen:** Hier wurden Spannungseinbrüche durch Exploration der Merkmalsdaten detektiert. Zunächst wurden relevante Merkmale und deren Ausprägungen aufgrund eines bekannten Spannungseinbruchs ermittelt. Nach der Aufstellung geeigneter Kriterien aufgrund dieses Einbruchs wurde nach weiteren Einbrüchen gesucht, die den Kriterien entsprechen. Die Resultate mussten von bloßen Messfehlern bzw. Ausreißern unterschieden werden.

### 5.3.2 F1: Detektion von Transienten in EDR-Rohdaten

In Anwendungsfall F1 wurde die Anwendbarkeit des entwickelten Discord/Motif-Detektionsverfahrens *SortList* auf die Detektion von Transienten in den EDR-Spannungs-Messdaten untersucht. Hierzu wurden zunächst Testdaten aus EDR-Spannungs-Messdaten vorbereitet, welche Transienten über sehr kurze Zeiträume (unter 2  $\mu$ s) an jeweils zwei Stellen und mit jeweils unterschiedlicher Ausprägung enthielten. Im Anschluss daran wurden experimentell geeignete Parameter bestimmt, mit welchen die Transienten zuverlässig detektiert werden konnten.

**Testdaten:** Alle Testdaten ( $Tr\_1$  bis  $Tr\_4$ ) wurden aus demselben EDR-Datensatz namens `KIT-EDR-EXPORT_EDR0001_T20140222-000003_L1` erzeugt, welcher ausschließlich die erste Phase der Spannung enthielt und zur leichteren Editierbarkeit als CSV-Datei aus dem EDR-Gerät *EDR0001* exportiert wurde. Die sinusförmige Spannung der ersten Phase pendelt zwischen ca. 333 und -333 Volt und enthält das übliche Sensorrauschen und Unregelmäßigkeiten der Sinusschwingung. Da keine Datensätze mit vorkommenden Transienten bekannt waren, wurde an Position 8173 und 8174 in jedem Testdatensatz unterschiedlicher Wert ( $w_1$  bis  $w_4$ ) zu den ursprünglichen Werten addiert und derselbe Wert an Position 8185 und 8186 subtrahiert. Der erste Transient *Trans1* umfasst also insgesamt vier geänderte Werte (jeweils zwei aufeinanderfolgend). Des Weiteren wurde ein weiterer Transient *Trans2* in alle Datensätze eingebracht, wobei hier an Position 11416 derselbe Wert ( $w_1$  bis  $w_4$ ) subtrahiert und an Position 11429 addiert wurde, so dass für *Trans2* nur zwei Werte von den Ursprungswerten abweichen. Die Werte  $w_1$  bis  $w_4$  sind abnehmend, so dass  $Tr\_1$  signifikante und  $Tr\_4$  kaum erkennbare Transienten enthält.

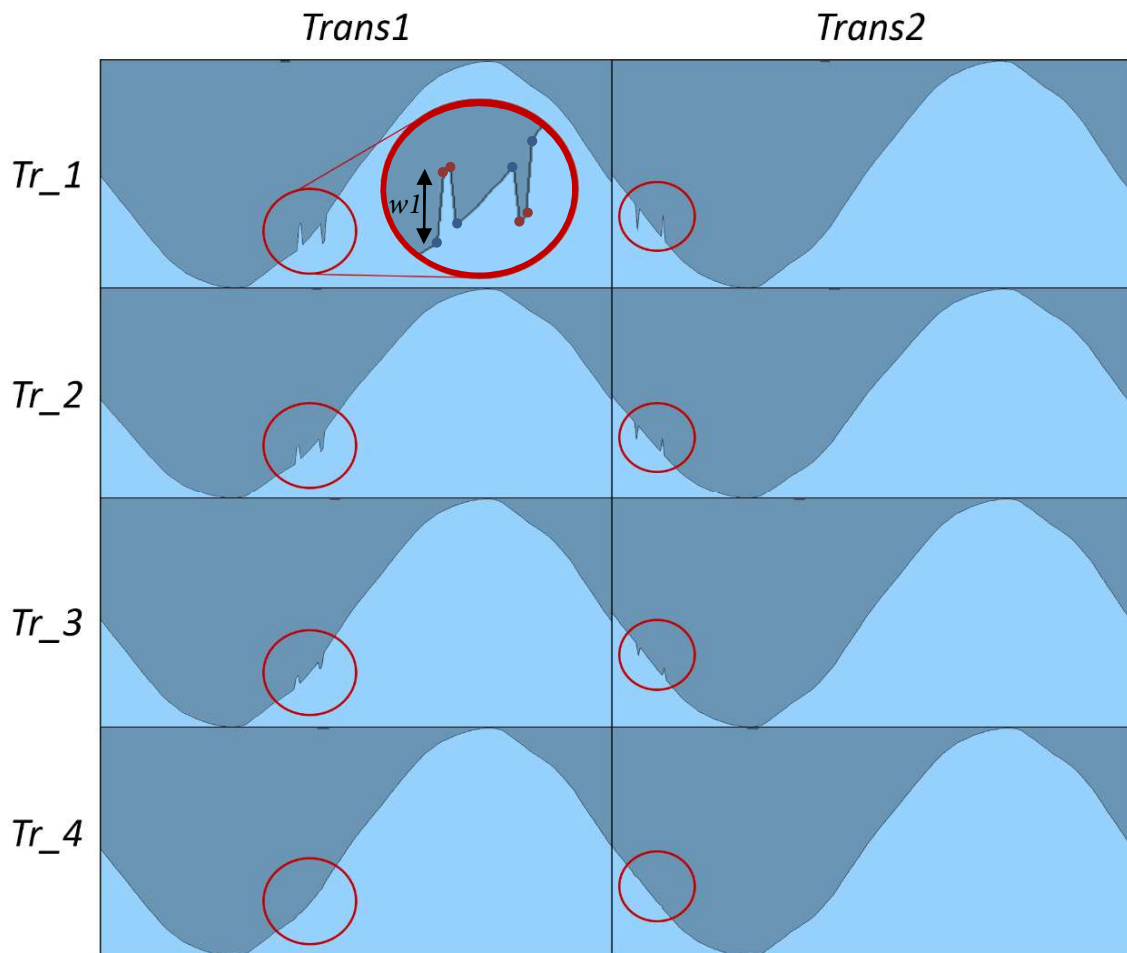
Die Eigenschaften der vier Testdatensätze und der jeweils eingefügten Transienten können aus Tabelle 5.3 und Abbildung 5.5 entnommen werden.

**Tabelle 5.3:** Aus EDR-Daten erzeugte Testdaten mit unterschiedlich ausgeprägten Transienten.

<i>Datensatz</i>		$Tr\_1$	$Tr\_2$	$Tr\_3$	$Tr\_4$
<i>Differenzwert</i> ( $w_1$ bis $w_4$ )	[V]	70	50	30	4

In Abbildung 5.5 ist deutlich erkennbar, dass es sich hier um besonders schwer detektierbare Transienten handelt. Der Mittelwert der jeweiligen Periode ändert sich kaum durch die Transienten, da sie einen negativen und einen positiven Ausschlag mit gleicher Amplitude enthalten. Au-

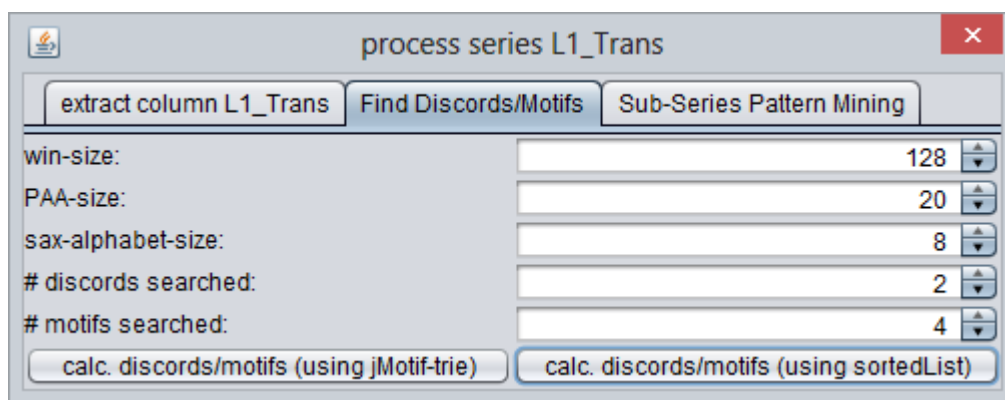
ßerdem wird weder ein zusätzlicher Nulldurchgang verursacht, noch eine Änderung der Frequenz. Im Fall  $Tr_4$  sind die Transienten nicht einmal visuell in der Kurvendarstellung sichtbar, da keine zusätzlichen lokalen Maxima oder Minima entstehen.



**Abbildung 5.5:** Darstellung der jeweiligen Periode mit den eingebrachten Transienten (*Trans1* links und *Trans2* rechts, rot markiert) in den Testdaten  $Tr_1$  bis  $Tr_4$ , mit Vergrößerung von *Trans1* in  $Tr_1$ .

**Experimentelle Parameterbestimmung:** Um geeignete Parameter für das in Abs. 3.7.7 eingeführte *SortList*-Verfahren zu bestimmen, mit welchen eine erfolgreiche und fehlerfreie Bestimmung der Transienten in  $Tr_1$  bis  $Tr_4$  möglich ist, wurden zahlreiche Parameter experimentell erprobt. Dabei wurde darauf geachtet, die leistungsbestimmenden Parameter (hauptsächlich *PAA-Größe* und *SAX-Alphabetgröße*) möglichst klein zu halten, um einen hohen Effizienzgrad der Suche zu erreichen, wodurch das Verfahren sich besser für eine Skalierung mit zunehmend größeren Datenmengen eignet und deutlich kürzere Laufzeiten ermöglicht. Prinzipiell konnte hier die bekannte Periodendauer von 256 Einzelwerten ausgenutzt werden. Als Fenstergröße des

Fensterverschiebungsalgorithmus wurden 128 Einzelwerte festgelegt, also eine halbe Periode. Geringe Abweichungen hiervon zeigten jedoch nur einen geringen Effekt auf die Qualität der Resultate, so dass davon auszugehen ist, dass das Verfahren auch bei leichten Schwankungen der Periodendauer anwendbar ist. Die beiden Parameter für die Anzahl darzustellender Diskords bzw. Motifs sind für die Suche mit dem *SortList*-Verfahren prinzipiell irrelevant, da die gesamte Liste der Teilsequenzen bei jedem Suchlauf vollständig sortiert wird, so dass eigentlich nur vier Parameter festgelegt werden müssen. Um die Ergebnisse signifikant darstellen zu können, wurde die Anzahl der darzustellenden (besten) Discords hier auf zwei festgelegt, so dass Fehldetektionen leicht erkannt werden können. Eine fehlerfreie Detektion aller Transienten ohne falsche Positive war mit den Parametern möglich, die aus Abbildung 5.6 entnommen werden können.

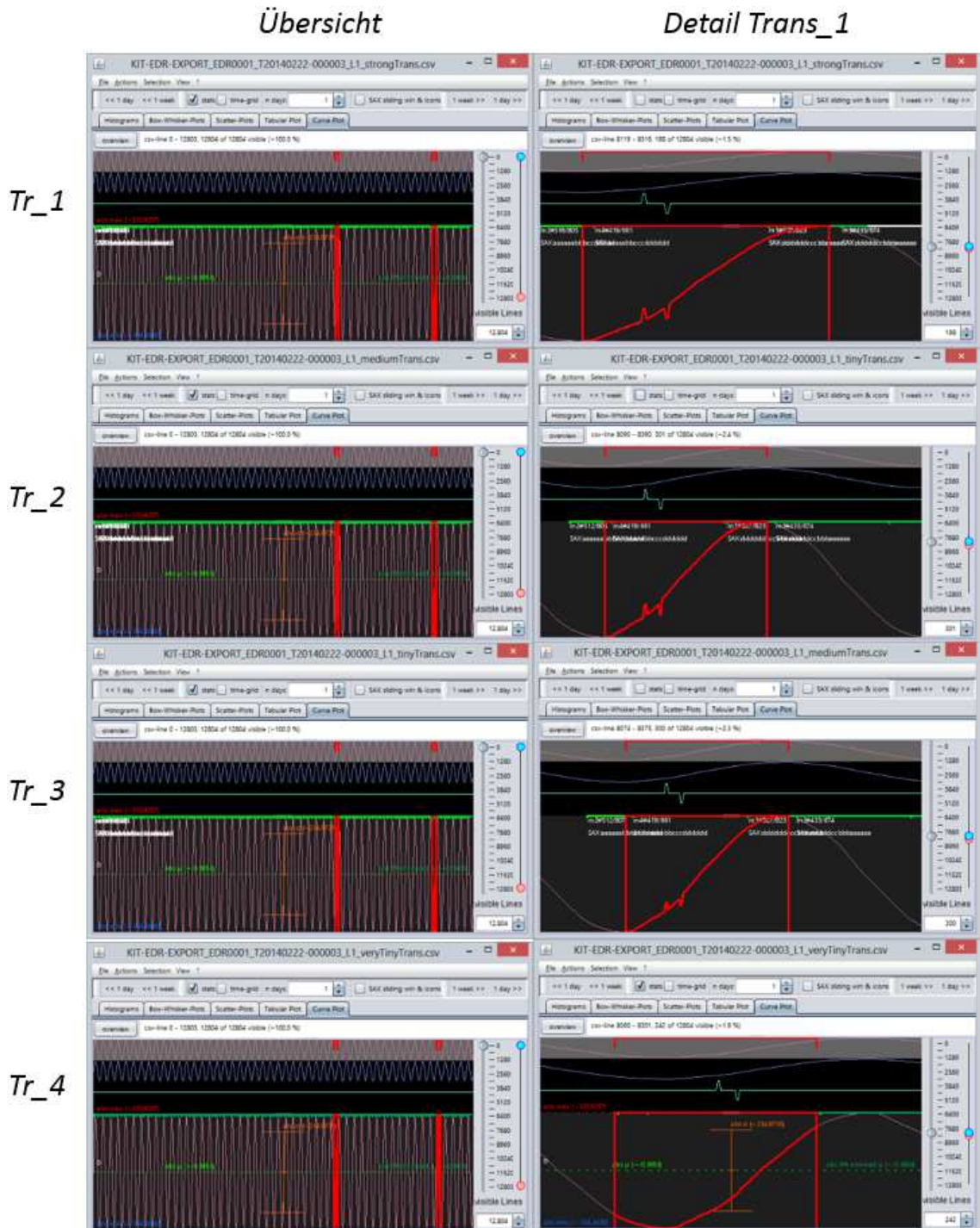


**Abbildung 5.6:** Parameterdialog mit experimentell ermittelten Parametern, welche eine fehlerfreie Detektion der Transienten *Trans1* und *Trans2* in allen Testdaten ermöglichen.

**Detektionsergebnisse:** Die Ergebnisse der Detektion mit den gewählten Parametern unterscheiden sich für alle Testdaten *Tr\_1* bis *Tr\_4* leicht hinsichtlich der beiden „besten“ gefundenen Discord-Sequenzen und deren exakten Positionen (siehe Code 5.4 und die Zusammenfassung der Discord-Darstellungen in *ViAT* in Abbildung 5.7). Die Fundstellen innerhalb der jeweiligen Periode stimmen jedoch in etwa überein, die Abweichungen sind sehr gering (abhängig von Fenstergröße und PAA-Größe). Bei anderen Verfahren wird meist ein Mittelwert über mehrere Perioden gebildet, was zu vergleichsweise ungenauen Positionsangaben führt.

Im Wesentlichen lässt sich durch die Ergebnisse aussagen, dass alle Transienten erfolgreich detektiert wurden. Interessanterweise reicht es aus, 128 `double`-Werte zu einer symbolischen Sequenz der Länge 20 mit acht möglichen Symbolen zusammenzufassen, um die Transienten zu detektieren, was die Informationsverarbeitung aufgrund der extremen Reduktion der Dimensionalität erheblich beschleunigt. Obwohl in *Trans\_1* lediglich vier und in *Trans\_2* lediglich zwei der 128 Werte innerhalb eines Fensters von den ursprünglichen Werten abweichen, führt dies zu

abweichenden Symbolsequenzen. Das erlaubt eine sehr effiziente Erkennung von Transienten, die sowohl für ein Echtzeit-Monitoring als auch in historischen Messdaten funktioniert.



**Abbildung 5.7:** Detektionsresultate für alle Datensätze Tr\_1 bis Tr\_4, links: Übersicht mit je zwei detektierten Transienten (rot), rechts Vergrößerung des durch Trans1 verursachten Discords (roter Kurvenabschnitt mit Umrahmung).

Besonders hervorzuheben ist die Tatsache, dass auch die Abweichungen in *Tr\_4* erfolgreich detektiert werden, obwohl die eingefügte Abweichung von den Originalwerten hier sehr gering war (214,920242 V statt 218,920242 V, eine Abweichung um nur 4 V, was ca. 0.6 % des Wertebereichs [-333, 333] entspricht) und keine lokalen Extremwerte erzeugte. Bei der experimentellen Parametersuche war dieser Fall am schwersten zu detektieren, während für *Tr\_1* bis *Tr\_3* viele Parameter zu Detektionserfolgen führten, da sich unterscheidbare Symbolsequenzen ergaben.

```
Discords, as <sax>, <offset>, <distance to closest neighbor>,
last to first:

Tr_1 (strong transients):
discord "aaabbcddffggggggfff", at 8174, distance: 0.022325654
discord "eccbbbbbbccdefghhhh", at 11425, distance: 0.023824550
total time using method sortedList: 0m 0s 231ms

Tr2 (medium transients):
discord "aaabbcddffggggggfff", at 8174, distance: 0.021378984
discord "eccbbbbbbccdefghhhh", at 11425 distance: 0.022325655
total time using method sortedList: 0m 0s 227ms

Tr_3 (small transients):
discord "bbabbbbcccdeffgghhhh", at 8124, distance: 0.020195645
discord "eecbbbbbbccdeggghhh", at 11420, distance: 0.020668980
total time using method sortedList: 0m 0s 226ms

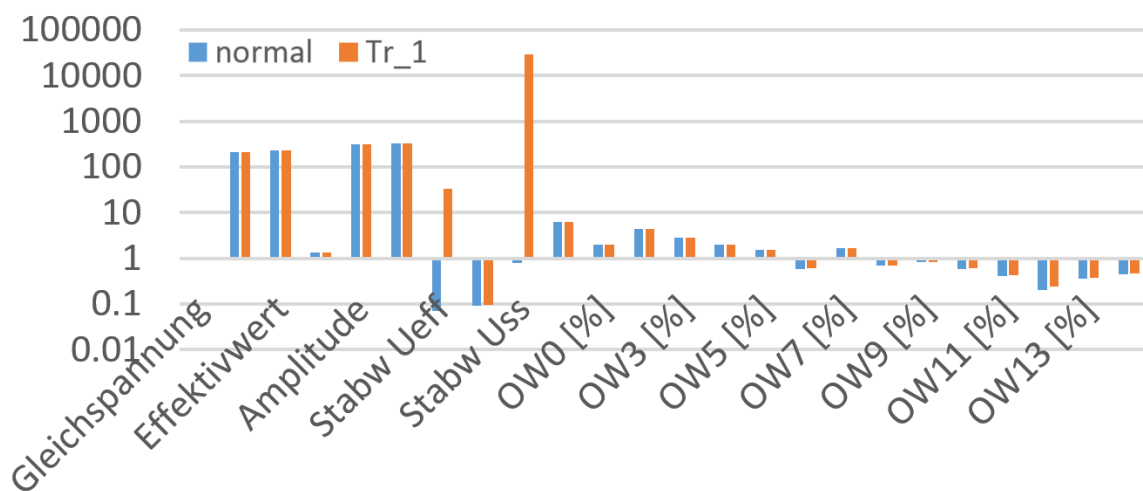
Tr_4 (tiny transients):
discord "bbbbbbccddeffgghhhh", at 8123, distance: 0.019091195
discord "hhhggfecbbbbbbccddf", at 11448, distance: 0.019170085
total time using method sortedList: 0m 0s 240ms
```

**Code 5.4:** Zusammengefasste Konsolenausgaben der Discord-Detektionsresultate für *Tr\_1* bis *Tr\_4*, mit den jeweiligen Discord-Sequenzen, Fundstellen und Pseudodistanzen (basierend auf der Position in der sortierten Liste).

**Vergleich mit anderen Verfahren, u. a. gemäß DIN EN 61000-4-30:** Für eine Einschätzung der Leistungsfähigkeit der Detektion von Transienten und allgemein auffälligen Abweichungen (wie in *TR\_4*) mit der *SortList*-Methode wurden dieselben Testdatensätze mittels verschiedener anderer Verfahren analysiert. Hierzu wurden die in 5.1.3 beschriebenen Merkmale für die einzelnen *Tr\_1* bis *Tr\_4* auf drei verschiedene Arten erzeugt:

- *M\_50P*: über 50 Perioden (wie in den sekundlichen EDR-Merkmalen)
- *M\_10P*: über zehn Perioden (gemäß *DIN EN 61000-4-30*)
- *M\_1P*: über eine Periode (um die maximale Genauigkeit des Ansatzes zu erreichen).

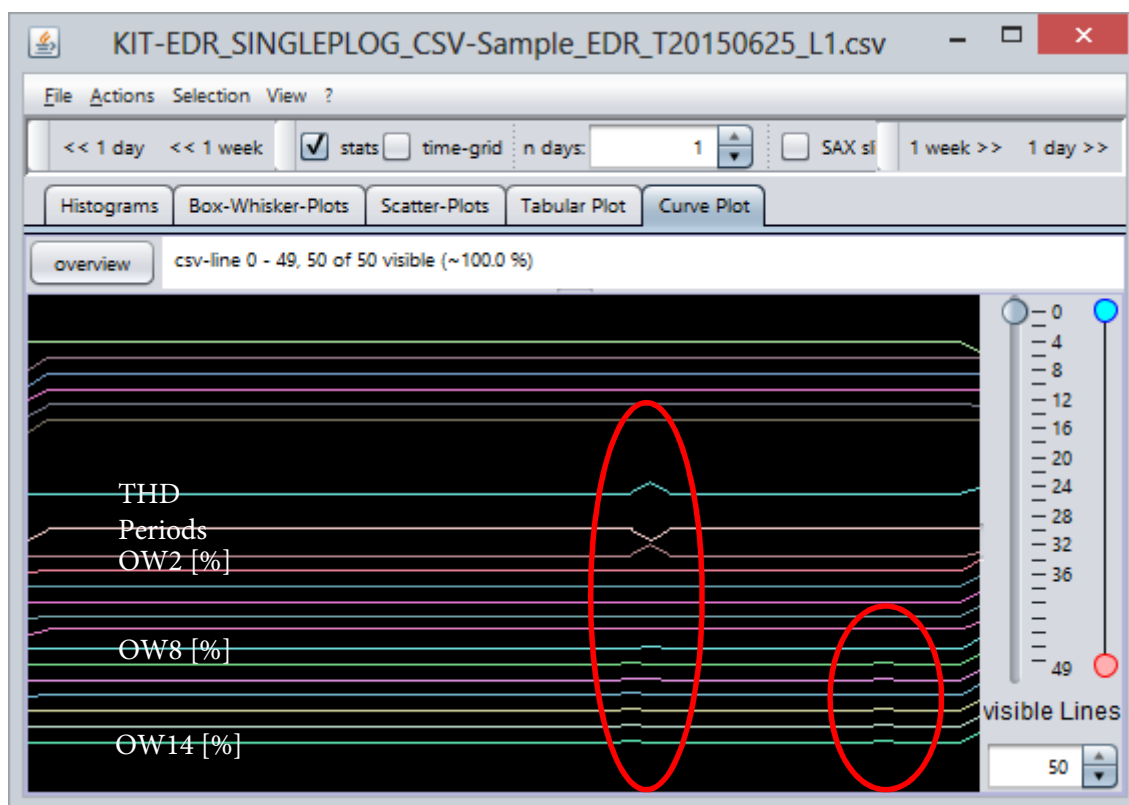
**Vergleich mit  $M_{50P}$ :** Da  $M_{50P}$  für die Testdaten mit 50 enthaltenen Perioden nur einen einzigen Wert für jedes Merkmal liefert, können lediglich die Einzelwerte mit denen einer Auswertung eines Datensatzes ohne eingebrachte Transienten verglichen werden. Das Balkendiagramm in Abbildung 5.8 zeigt die entsprechenden Werte für  $Tr_1$  (orange) im Vergleich zu Testdaten ohne Transienten (blau). Die beiden extrem voneinander abweichenden Werte (Stabw Ueff und Stabw Uss) scheinen durch Berechnungsfehler zustande zu kommen, die durch das Fehlen vorangehender und nachfolgender Perioden entstehen können. Lediglich OW12 [%] zeigt einen merklichen und nachvollziehbaren Unterschied der Werte (0,23681 in  $Tr_1$  und 0,20622 im Datensatz ohne Transienten). Das Ergebnis ist aufgrund der geringen Abweichung jedoch so zu bewerten, dass bei einer Merkmalsberechnung über 50 Perioden keine zuverlässige Detektion von Transienten möglich ist.



**Abbildung 5.8:** Balkendiagramm zum Vergleich der resultierenden Merkmalswerte für Auswertung von  $Tr_1$  und denselben Testdaten ohne Transienten mit  $M_{50P}$  – Auffällige Unterschiede zeigen Stabw Ueff, Stabw Uss und OW12 [%].

**Vergleich mit  $M_{10P}$ :**  $M_{10P}$  hat potentiell 41 Einzelwerte ( $50 - (10 - 1)$ ) für einen Testdatensatz, jedoch ist jeweils der erste und letzte Wert ungültig, da hierzu gemäß *DIN EN 61000-4-30* die jeweils vorangehende und nachfolgende Periode in der Berechnung berücksichtigt werden müssen, die beide hier nicht verfügbar sind. Daher ist die Kurvendarstellung der Merkmale nicht sonderlich aussagekräftig. Auf jeden Fall kann aber festgehalten werden, dass bei 41 Merkmalswerten für 50 Perioden keine genaue Positionsangabe möglich ist, nicht einmal die Perioden mit enthaltenen Transienten können durch das Vorgehen ermittelt werden. Da die Transienten *Trans1* und *Trans2* so gewählt wurden, dass sich der Mittelwert der Periode nicht verändert, bietet  $M_{10P}$  keine zuverlässige Detektion für derartige Transienten.

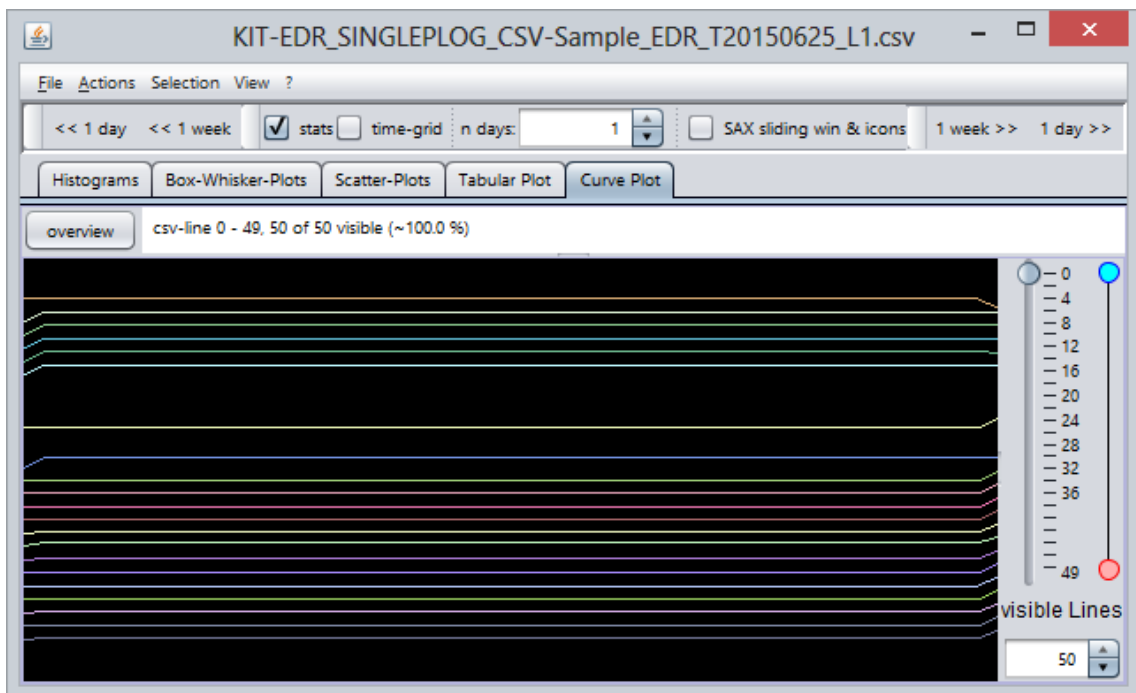
**Vergleich mit  $M_{IP}$ :**  $M_{IP}$  liefert fünfzig Einzelwerte für einen Testdatensatz und erlaubt eine Kurvendarstellung. Abbildung 5.9 zeigt das Ergebnis der Auswertung von  $Tr_1$ . Sichtbar ist v. a. der erste Transient  $Trans_1$ . Er macht sich in den Kurven für THD, Periods und OW2 [%] stark bemerkbar und beeinflusst auch weitere Oberschwingungen (OW8 [%] bis OW14 [%]).  $Trans_2$  wirkt sich hingegen nur sehr geringfügig, v. a. auf OW9 [%], OW10 [%] und OW12 [%] bis OW14 [%] aus. Es kann jedoch festgehalten werden, dass mit  $M_{IP}$  die in  $Tr_1$  enthaltenen starken Transienten prinzipiell detektiert werden können. Allerdings sind die abweichenden Werte stets Einzelwerte, so dass sie nicht eindeutig von Messfehlern und Ausreißern unterschieden werden können, was eine Detektion auf Basis von  $M_{IP}$  sehr fehleranfällig macht. Auf Basis von  $M_{IP}$  kann die Position von Transienten in den Originaldaten aufgrund der Mittelung über eine Periode auch maximal auf eine Periode genau ermittelt werden.



**Abbildung 5.9:** Merkmalskurven der Auswertung von  $Tr_1$  mit  $MIP$  in ViAT, mit Markierung der Bereiche mit Transienten (rot) und Bezeichnungen auffälliger Merkmale (weißer Text links).

Eine Auswertung des Testdatensatzes mit den kleinsten Transienten ( $Tr_3$ ) zeigt bereits wesentlich kleinere Abweichungen, die kaum noch von den allgemeinen Werteschwankungen unterschieden werden können. In  $Tr_4$ , dem Datensatz mit den geringsten Abweichungen sind keinerlei Auffälligkeiten durch eine Auswertung mehr feststellbar (siehe Abbildung 5.10).





**Abbildung 5.10:** Übersichtsdarstellung des Auswertungsergebnisses von Tr\_4 durch M\_1P in ViAT ohne sichtbare Ausschläge (auch bei extremer Vergrößerung).

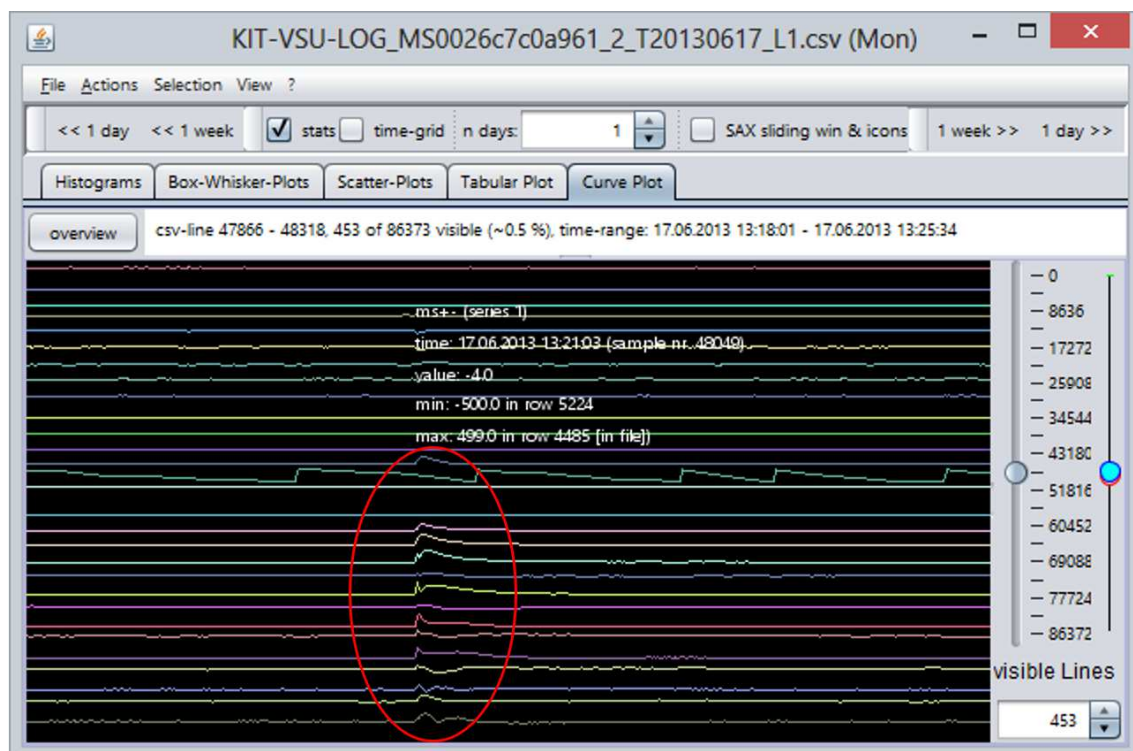
### 5.3.3 F2: Detektion kurzer Spannungseinbrüche in EDR-Merkmalen

In Anwendungsfall F2 wurde das *FraScaTi*-System eingesetzt, um einen bekannten kurzen Spannungseinbruch im Stromnetz des KIT Campus Nord durch explorative Analyse der EDR-Merkmalen- und Messdaten näher zu untersuchen und Merkmale zu identifizieren, die sich zur Detektion derartiger Ereignisse eignen. Aufgrund der gewonnenen Erkenntnisse sollte anschließend nach weiteren, unbekanntem Spannungseinbrüchen gesucht werden.

**Einordnung kurzer Spannungseinbrüche und der EDR-Merkmale:** Kurze Spannungseinbrüche, die sich über mehrere Perioden ausdehnen, lassen sich aufgrund von Merkmalen gemäß des IEC-Standards *DIN EN 61000-4-30* bzw. *VDE 0847-4-30* erkennen. Der Standard definiert Mess- und Auswertalgorithmen für Geräte und Verfahren zur Messung der Stromqualität und definiert drei Genauigkeits-Klassen von Messverfahren: Verfahren gemäß *Klasse A* müssen höchsten Genauigkeitsansprüchen genügen und gleiche Resultate bei gleichen Eingangssignalen erzeugen. Verfahren der *Klasse S* eignen sich hingegen ausschließlich für statistische Kontroll-Messungen und sind gegenüber *Klasse A* weniger stringent definiert. Die *Klasse B* definiert die Anforderungen an Verfahren, die sich lediglich zur qualitativen Beurteilung der Stromqualität eignen und nur geringen Genauigkeits- und Korrektheitsansprüchen genügen. Merkmale gemäß der *Klasse*

A werden auch direkt in den EDR-Geräten vorberechnet (siehe Abs. 5.1.3) und als sekundliche Aggregatwerte für jede Phase in einer Datei für einen Messtag gespeichert. In eben diesen EDR-Merkmalen sollten also auch kurze Spannungseinbrüche zu einer merklichen Veränderung einzelner Merkmale führen.

**Bekannter Spannungseinbruch:** Um Merkmale zu identifizieren, welche sich zur Detektion von Spannungseinbrüchen eignen und so einen Zusammenhang zu den EDR-Rohmessdaten herzustellen, wurde ein bekannter Spannungseinbruch mithilfe des *FraScaTi*-Systems ausgewertet. Der Einbruch war durch die Abteilung *Technische Infrastruktur und Dienste (TID)* des KIT festgestellt worden, welche starke Spannungsschwankungen im Netz des KIT Campus Nord erfasst und aufzeichnet. Bekannt waren lediglich der ungefähre Zeitpunkt und die ungefähre Dauer des Einbruchs. Es wurde vermutet, dass die Ursache in den vorgelagerten Übertragungsnetzen (im 110 kV- oder 400 kV-Netz) zu suchen sei. Der genaue Zeitpunkt des Einbruchs war leider nur auf die Minute genau bekannt, die Dauer betrug wenige Millisekunden („ca. 30 ms“). In der UTC-Zeit, wie sie bei der EDR-Erfassung verwendet wird, fand der Spannungseinbruch um 13:21 Uhr UTC statt, also um 12:21 MEZ, am 17.06.2013.



**Abbildung 5.11:** Übersicht über die EDR-Merkmalzeitreihen im Zeitraum des bekannten Spannungseinbruchs mit roter Umrandung einiger auffälliger Merkmalsverläufe.

**Exploration der Merkmalszeitreihen:** Als erstes wurden die EDR-Merkmalen für den 17.06.2013 mithilfe des *ViAT*-Moduls exploriert. Dabei wurden Auffälligkeiten in mehreren Merkmalen festgestellt (siehe Abbildung 5.11), wie sie prinzipiell durch einen kurzen Spannungseinbruch ausgelöst werden können. Signifikante Auffälligkeiten zeigten sich v. a. in den Merkmalen, die in Abbildung 5.12 aufgeführt sind. Die Abbildung enthält auch Sparklines<sup>42</sup>, die den Kurvenverlauf jeder Merkmalszeitreihe zum Zeitpunkt des Einbruchs und das anschließende Einschwingverhalten zeigen, sowie die Position der entsprechenden Zeitreihe in der *ViAT*-Visualisierung (von oben) und das Verlaufsmuster (p steht hier für einen positiven und n für einen negativen Trend).

**Identifikation geeigneter Merkmale für eine Detektion:** Alle aufgeführten Merkmale zeigen im Ereigniszeitraum ein auffälliges Verhalten. Im Gesamtverlauf der Kurven weisen jedoch viele der Merkmale auch an weiteren Stellen starke Schwankungen auf, die nicht trivial vom auffälligen Kurvenverlauf während des Spannungseinbruchs unterschieden werden können. Für eine zuverlässige Detektion sollten Merkmale ausgewählt werden, die ausschließlich während des Ereigniszeitraums vom Normalverhalten abweichen. Das Normalverhalten der Merkmale ist jedoch nicht vollständig bekannt, so dass nicht garantiert werden kann, dass im Datensatz nicht weitere Spannungseinbrüche enthalten sind, die nicht durch TID festgestellt wurden. Eine Detektion basierend auf der Position der Maximalwerte bzw. der Überschreitung fester Grenzwerte kommt nur für einige der auffälligen Merkmale in Frage, da in den meisten Zeitreihen weitere lokale Maxima mit höherem Betrag enthalten sind. Das lässt sich leicht anhand der in *ViAT* enthaltenen Anzeige von Aggregatstatistiken zeigen. Abbildung 5.13 veranschaulicht dies anhand der THD-Kurve, welche die harmonische Gesamtverzerrung [158] repräsentiert.

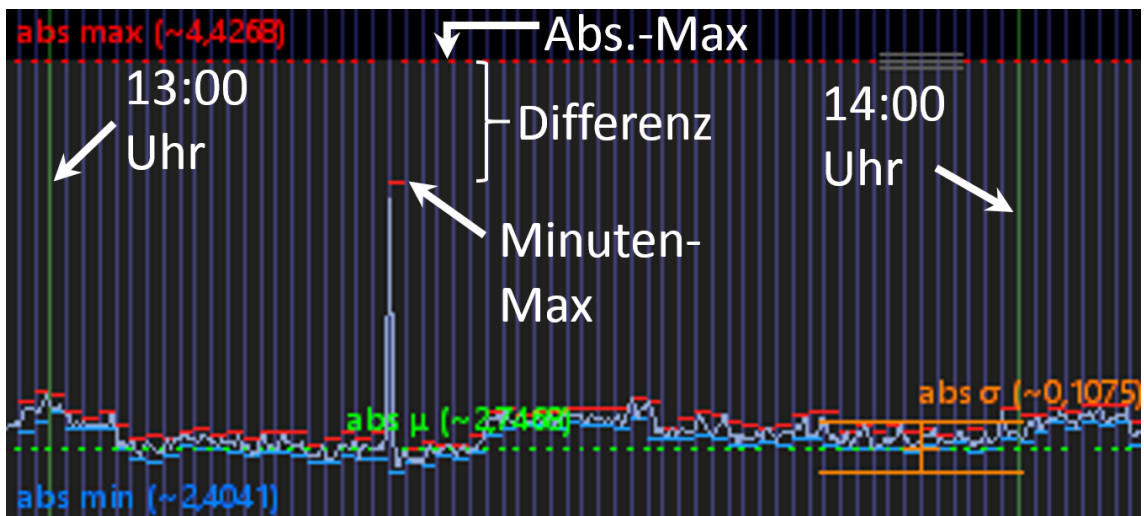
Auf diese Weise lassen sich die Merkmale identifizieren, deren globaler Maximalwert sich an der selben Position befindet wie die Ausschläge des Spannungseinbruchs und die daher prinzipiell eine grenzwertbasierte Detektion zulassen. Dazu gehören ausschließlich einzelne Oberschwingungen, deren Tages-Maximalwerte in Tabelle 5.4 zusammen mit Uhrzeit des Auftretens aufgeführt sind. Die Tabelle enthält zum Vergleich auch entsprechende Werte und Auftrittszeiten für den Vortag des selben EDR-Geräts.

---

<sup>42</sup> Sparklines sind kleine Zeitreihendarstellungen mit typografischer Auflösung und der Größe eines Wortes, welche sich z. B. in Fließtext integrieren lassen. Sie wurden von Edward Tufte in seinem Buch *Beautiful Evidence* [297] erstmalig eingeführt und sind mittlerweile eine weitverbreitete kompakte Zeitreihendarstellung, welche z. B. in Standardsoftware wie *Microsoft Excel* integriert ist.

<b>Merkmal</b>	<b>Sparkline</b>	<b>Position (v.o.)</b>	<b>Verlauf</b>
Gleichspannung		4	pnpn
Gleichrichtwert		5	n
Effektivwert		6	n
Scheitelfaktor		7	pnp
Amplitude		8	n
Abs. MaxV		9	n
Stabw Ueff		10	p
Stabw Uss		12	p
THD		13	p
OW2 [%]		17	p
OW3 [%]		18	p
OW4 [%]		19	p
OW5 [%]		20	p
OW6 [%]		21	p
OW7 [%]		22	p
OW8 [%]		23	p
OW9 [%]		24	p
OW10 [%]		25	p
OW11 [%]		26	pnp
OW12 [%]		27	pnpnp
OW13 [%]		28	p
OW14 [%]		29	pnpn

**Abbildung 5.12:** EDR-Spannungs-Merkmale mit signifikanter Änderung zum Zeitpunkt des bekannten Spannungseinbruchs (13:20:57 Uhr bis 13:22:45 Uhr).

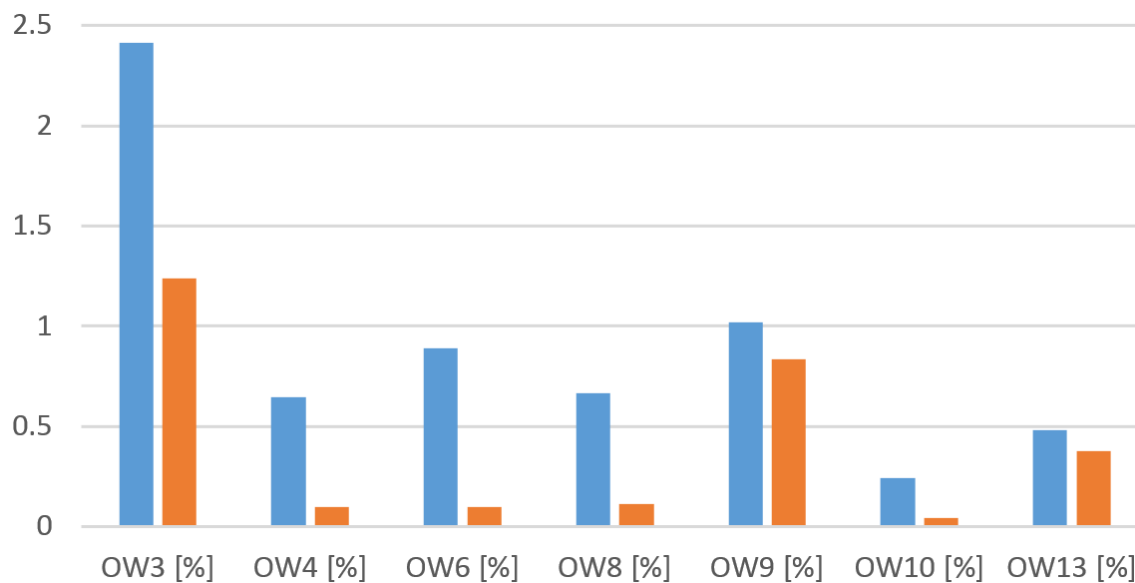


**Abbildung 5.13:** Ausschnitt der THD-Kurve in ViAT mit minütlichen Aggregatstatistiken (Max rot, Min blau), das lokale Maximum während des Spannungseinbruchs ist kleiner als das absolute Maximum.

**Tabelle 5.4:** Merkmale mit Tagesmaximum zur Zeit des Spannungseinbruchs am 17.06.2013 im Vergleich zum Vortag, mit bedingter Formatierung der Zellhintergründe der Maximalwerte. Eigene Tabelle.

Merkmal	Max. am Tag des Spannungseinbruchs	Uhrzeit des Max. a. T. d. Sp.	Max. am Vortag	Uhrzeit d. Max. a. Vortag
OW3 [%]	2.4133	13:21:06	1.2403	12:04:30
OW4 [%]	0.6457	13:21:07	0.0976	04:34:20
OW6 [%]	0.8889	13:21:04	0.0987	04:21:18
OW8 [%]	0.6676	13:21:05	0.1101	04:21:18
OW9 [%]	1.0178	13:21:04	0.8351	14:33:29
OW10 [%]	0.2413	13:21:04	0.0428	22:28:43
OW13 [%]	0.4821	13:21:07	0.3769	14:32:21

Abbildung 5.14 zeigt den Vergleich der Maximalwerte zusätzlich als Balkendiagramm. Bei Betrachtung der Werte und ihrer Positionen fällt auf, dass OW9 [%] und OW13 [%] sich an den beiden Tagen weniger signifikant unterscheiden als alle anderen Oberschwingungen, für welche der jeweilige Maximalwert zum Zeitpunkt des Einbruchs etwa den doppelten Betrag (OW 3 [%]) oder höher (OW4 [%], OW6 [%], OW8 [%], OW10 [%]) aufweist. Außerdem tritt das Tagesmaximum von OW6 [%] und OW8 [%] am Vortag zur gleichen Zeit auf.



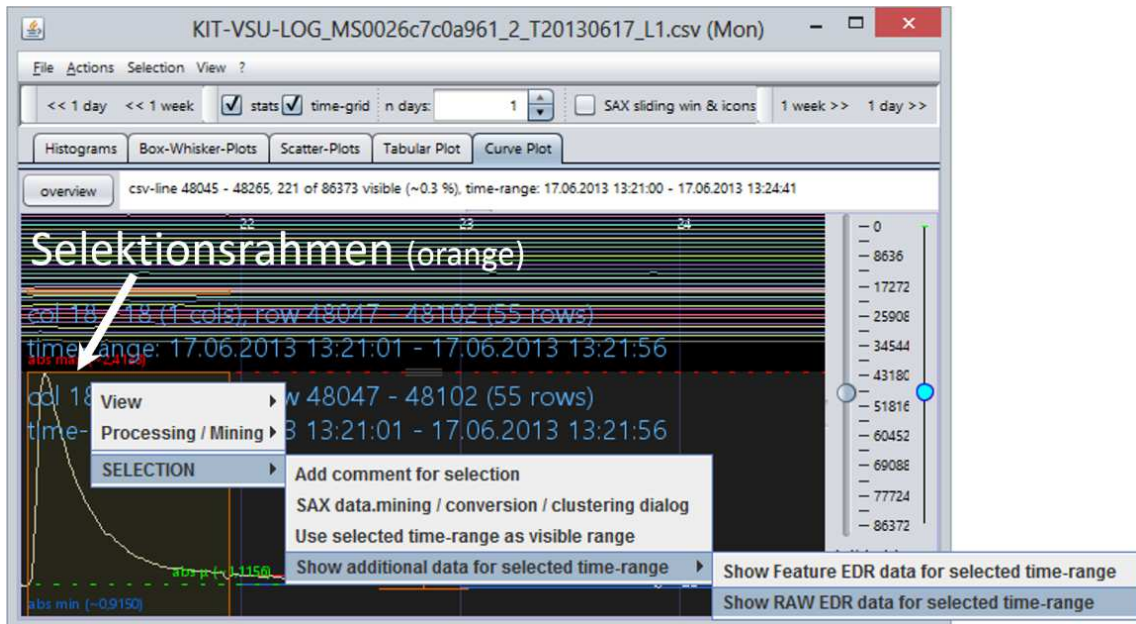
**Abbildung 5.14:** Balkendiagramm zum Vergleich der Tages-Maximalwerte der Oberschwingungen (blau: Tag des Spannungseinbruchs, orange: Vortag), welche zur Zeit des Spannungseinbruchs ein Tagesmaximum aufweisen.

**Mögliche Detektion für Spannungseinbrüche<sup>43</sup> aufgrund gleichzeitiger Maximalwerte bestimmter Oberschwingungen:** Es liegt nahe, eine Detektion von Spannungseinbrüchen auf dem gleichzeitigen Auftreten von Maxima für OW3 [%], OW4 [%] und OW10 [%] zu basieren. Hierfür können natürlich auch lokale Maxima statt der Tagesmaxima gewählt werden, solange dabei für alle drei Merkmale gleichzeitig ein festzulegender Grenzwert überschritten wird, der höher als das durchschnittliche Tagesmaximum sein sollte, um Fehldetektionen zu vermeiden.

**Identifikation des exakten Auftrittszeitpunkts und der Dauer des Spannungseinbruchs:** Um den genauen Zeitpunkt des Einbruchs herauszufinden, müssen die hochfrequenten Spannungsmessdaten für den entsprechenden Zeitraum (Minute 21) untersucht werden. Diese lassen sich in *ViAT* komfortabel anzeigen, indem über das Kontextmenu der entsprechende Eintrag gewählt

<sup>43</sup> Die Detektion von Spannungseinbrüchen ist natürlich auf die über eine Sekunde berechneten EDR-Merkmalen beschränkt.

wird (siehe Abbildung 5.15). Daraufhin wird aus dem gewählten Zeitbereich und dem Dateipfad der aktuell geladenen Merkmalsdatei automatisch der entsprechende Dateipfad der EDR-Messdaten erzeugt<sup>44</sup>, welche die Daten für denselben Kanal und dasselbe EDR-Gerät enthalten. Existieren die Daten, werden sie geladen und visualisiert. Das entsprechende Untermenü ist nur sichtbar, wenn es sich bei den aktuell geladenen Daten um EDR-Daten handelt und ein Zeitbereich selektiert wurde.

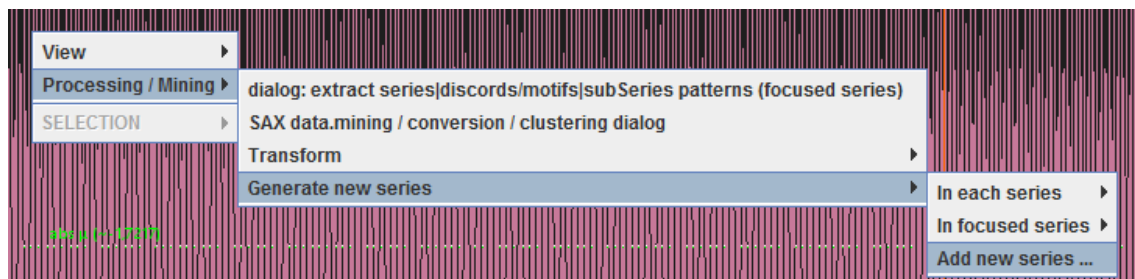


**Abbildung 5.15:** Aufruf der Anzeige der Spannungs-Messdaten im selektierten Zeitbereich über das Kontextmenü von ViAT.

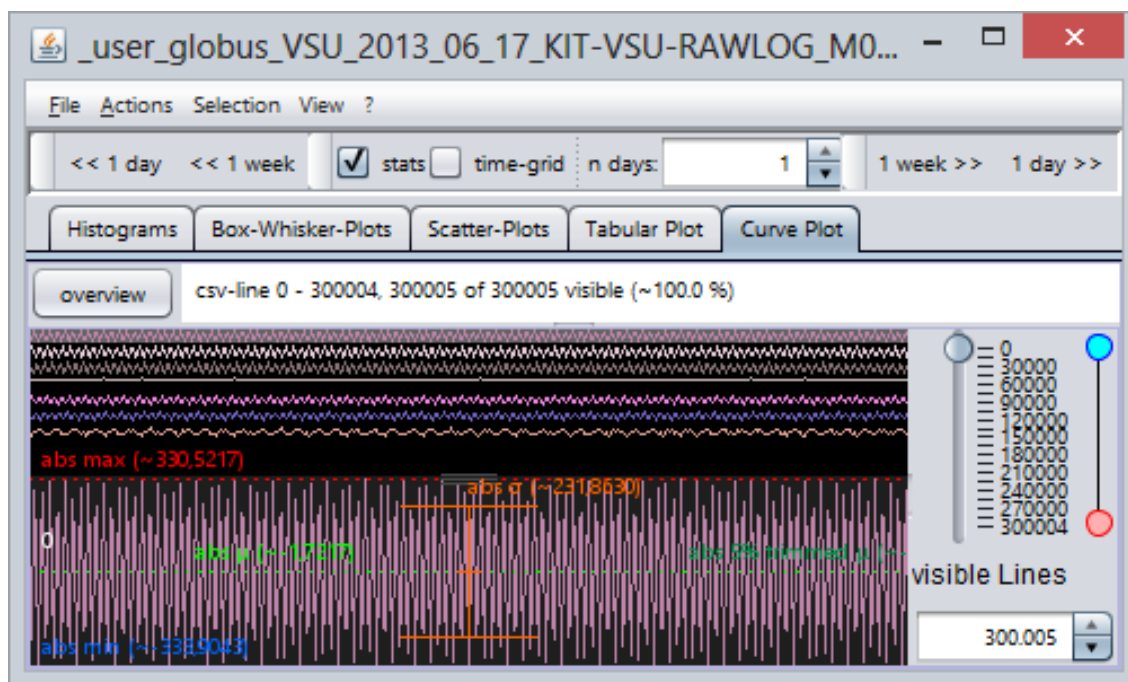
Die Exploration der hochfrequenten Spannungs-Messdaten einer Minute (siehe Abbildung 5.17) stellt eine besondere Herausforderung dar, da über 300 000 Messwerte für jeden Kanal (drei Phasen Spannung, drei Phasen Strom und der Strom im Nullleiter) angezeigt werden und sich die einzelnen Perioden der Sinuskurven der Spannung nicht merklich unterscheiden. Bei der Kurvendarstellung der gesamten Minute kommt es zusätzlich zu Aliasingproblemen. Die Zeitbereichstatistiken helfen hier nicht weiter, da die Unterschiede der Maxima viel zu gering sind, um visuell erkennbar zu sein. Für diesen Fall wurde eine zusätzliche Funktionalität zur Erzeugung

<sup>44</sup> Das ist aufgrund der uneinheitlichen Versionen der Datei- und Ordnernamensgebung, die sich während der langjährigen EDR-Messungen ergeben haben, nicht trivial, da zur Ermittlung der vorliegenden Version verschiedene Kombination (z. B. mit und ohne führende Nullen bei Zeit- und Datumsangaben) geprüft werden müssen. Die automatische Erzeugung funktioniert nur für Daten, die in HDFS vorliegen, nicht für lokale Daten, da der genaue Zeitbezug ansonsten nicht ermittelbar ist.

und Darstellung von Hilfs-Zeitreihen in *ViAT* integriert, welche die explorative Analyse sinus-ähnlicher (und weiterer periodischer) Zeitreihen in vielen Fällen erleichtern. Der Vorgang kann bequem über das Kontextmenü von *ViAT* veranlasst werden (siehe Abbildung 5.16).



**Abbildung 5.16:** Erzeugung und Anzeige zusätzlicher Zeitreihen über das Kontextmenü von *ViAT*.

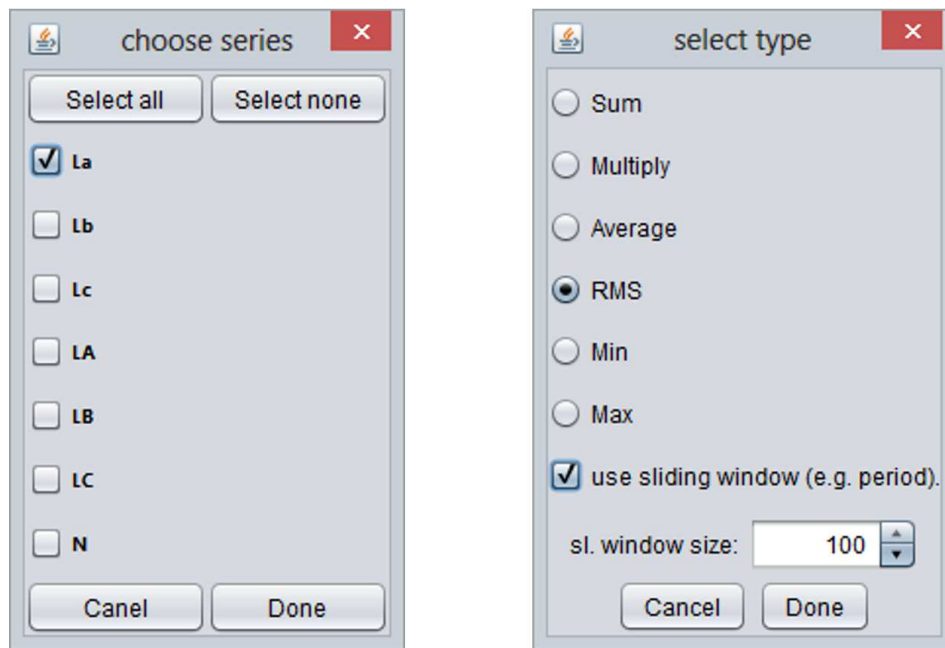


**Abbildung 5.17:** Überblicksdarstellung über eine Minute an Spannungs-Messdaten (unten Detaildarstellung der ersten Phase).

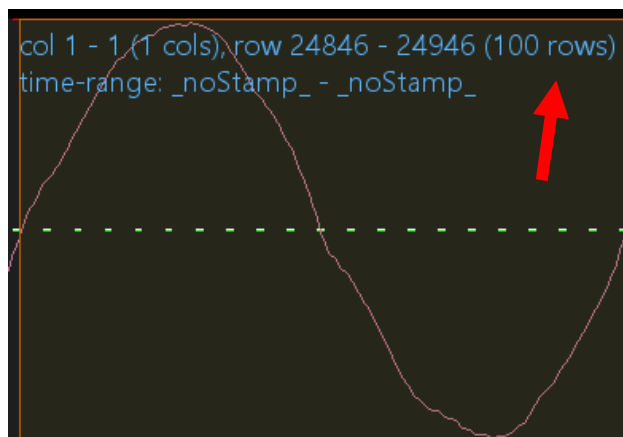
Über die in Abbildung 5.18 gezeigten Dialoge können verschiedene Arten neuer Zeitreihen erzeugt werden (siehe Abs. 3.6.2). In diesem Fall bietet sich die Berechnung des Effektivwerts an, welcher sich bei einem Spannungseinbruch signifikant verringern sollte. Um jedoch eine Zeitreihe gleicher Länge wie die Messdaten zu erhalten, wird hier abweichend von der im Standard *DIN EN 61000-4-30* vorgeschlagenen Berechnung über mehrere Perioden (10 bzw. 12) ein parametrisierbarer Fensterverschiebungsalgorithmus verwendet, wobei die Fenstergröße auf die Anzahl der Messwerte festgelegt wird, die innerhalb einer Periode zu erwarten ist. Da das Messgerät,



mit dem die Messung durchgeführt wurde, im Sommer 2013 noch mit 5 kHz betrieben wurde, enthält eine Priode hier 100 Werte. Bei der aktuellen Abtastrate von 12,8 kHz enthält eine Periode entsprechend 256 Messwerte.



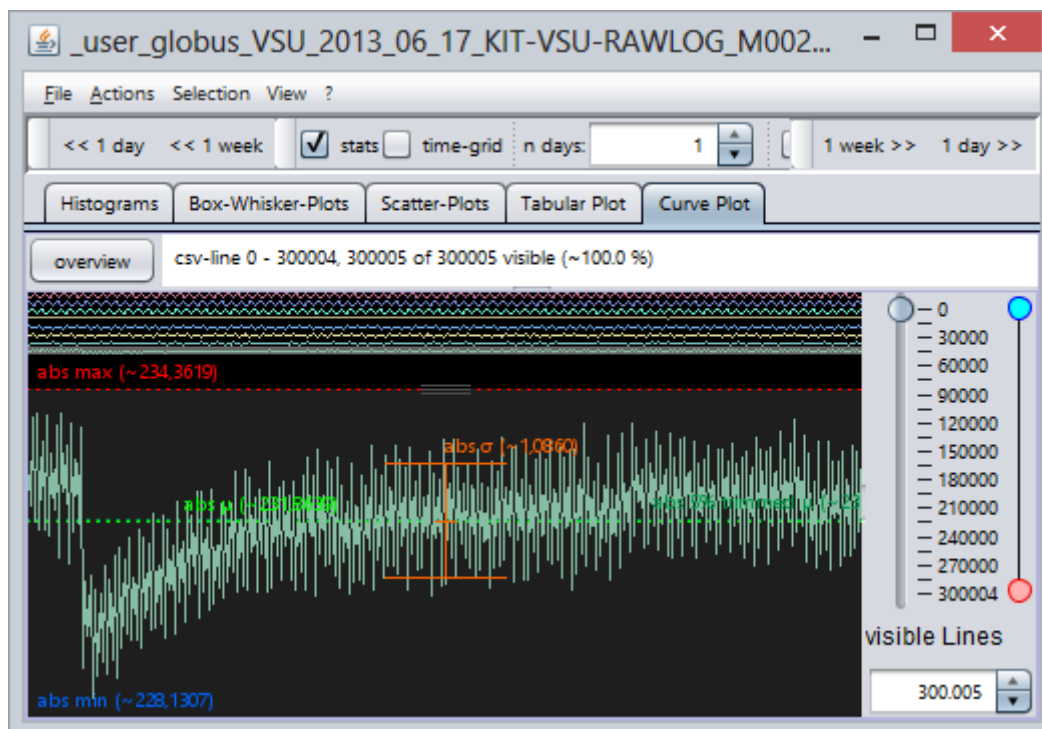
**Abbildung 5.18:** Dialoge zur Erzeugung neuer Zeitreihen in *ViAT*, Auswahl der Zeitreihe(n) aus der/denen neue Daten erzeugt werden sollen (links) und Auswahl der Berechnung und der Parameter (rechts).



**Abbildung 5.19:** Ermittlung der Messwerte einer Periode in *ViAT*.

Der Wert kann in *ViAT* komfortabel ermittelt werden, indem eine Periode selektiert wird, da innerhalb des Selektionsrahmens textuelle Informationen über die Auswahl angezeigt werden

(siehe Abbildung 5.19). Es kann natürlich auch eine vielfache Periodenlänge gewählt werden, um die RMS-Kurve etwas zu glätten (schließlich ist mit geringen Frequenzschwankungen zu rechnen), doch an dieser Stelle soll der genaue Zeitpunkt des Spannungseinbruchs ermittelt werden, weshalb eine Periodenlänge festgelegt wird. Zunächst wird eine neue RMS-Zeitreihe für die erste Phase erzeugt. Nach Berechnung der neuen Zeitreihe wird sie sofort in *ViAT* angezeigt (siehe Abbildung 5.20).

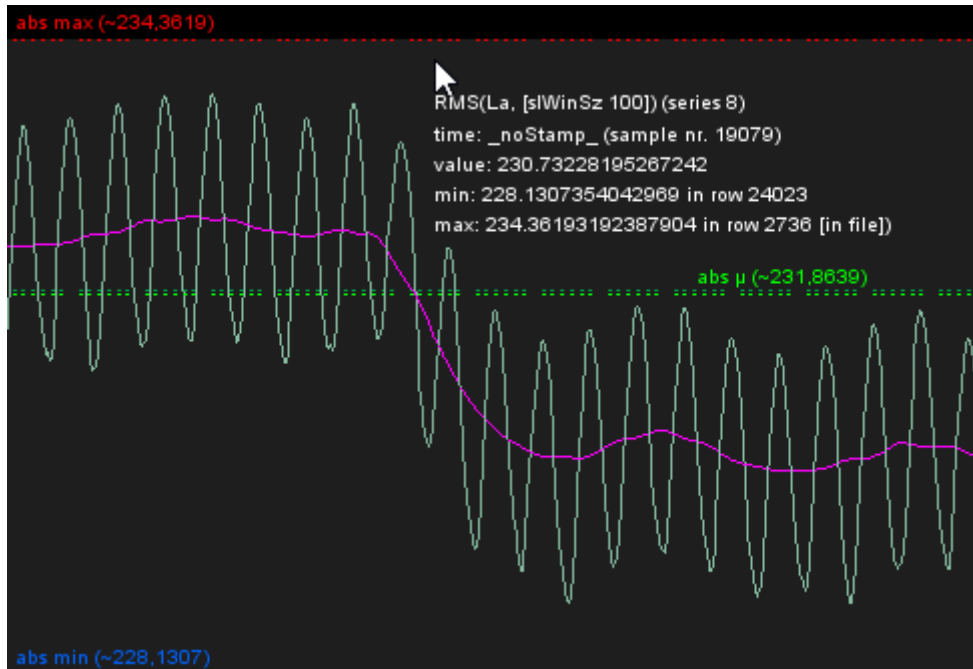


**Abbildung 5.20:** Kurvendarstellung der über eine Fenstergröße von 100 Messwerten (eine Periode) berechneten RMS-Zeitreihe.

Die Kurve zeigt einen deutlich erkennbaren, plötzlichen Einbruch. Über den Dialog in Abbildung 5.16 kann für die Kurve noch ein zusätzlicher gleitender Mittelwert über 100 Werte berechnet und angezeigt werden, dessen Kurve überlagert angezeigt wird (siehe Abbildung 5.21). Durch Drehen des Mauseis bei gehaltener Strg-Taste lässt sich schnell und stufenlos der Bereich unter dem Mauszeiger vergrößern. Der genaue Zeitpunkt kann in *ViAT* leicht durch Positionierung des Mauszeigers an der entsprechenden Position ermittelt werden, wodurch textuelle Detailinformation angezeigt wird, welche u. a. die Position in der Zeitreihe enthält (weißer Text in Abbildung 5.21). Die ermittelte Position ist der 19079-te Messwert, was bei der Abtastrate von 5 kHz 3,8158 Sekunden entspricht. Somit ergibt sich ein relativ genauer Zeitpunkt für den Spannungseinbruch:

13:21:03:816 UTC

Die Dauer lässt sich hingegen nicht genau bestimmen. Bei näherer Betrachtung des Verlaufs in Abbildung 5.20 dauert der Einschwingvorgang mindestens eine halbe Minute.



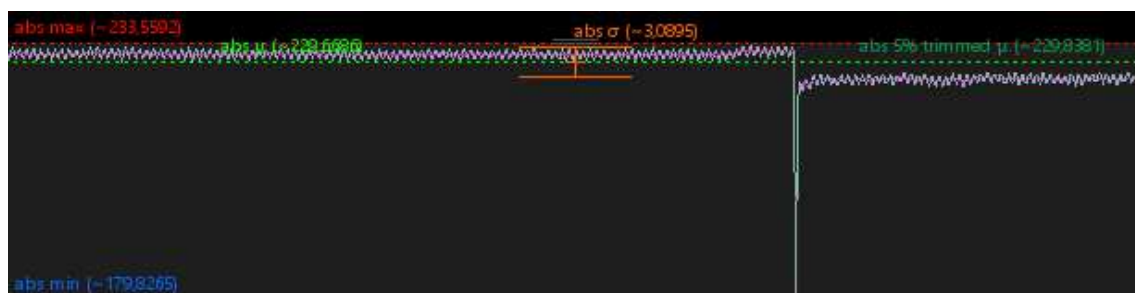
**Abbildung 5.21:** Detailausschnitt der erzeugten RMS-Kurve der ersten Phase (türkis) in *ViAT* mit zusätzlich erzeugter Kurve eines gleitenden Mittelwerts (pink) und textueller Positionsinformation unterhalb des Mauszeigers (weiß).

**Weitere Auffälligkeiten am selben Tag:** Durch ähnliches exploratives Vorgehen wurde im gleichen Tagesdatensatz der EDR-Merkmale ein weiterer, früherer und noch signifikanterer Spannungseinbruch entdeckt, der sich um 12:24:36 Uhr UTC ereignete. Da im vorigen Fall die aus den Spannungs-Messdaten erzeugte RMS-Kurve der ersten Phase einen guten Anhaltspunkt bot, wurde der sekundliche Verlauf des RMS (also des Effektivwerts) in den Merkmalsdaten nochmals geprüft (siehe Abbildung 5.22). Besonderes Interesse galt dabei der Stelle mit dem Tagesminimum (ca. 224,8 V), dessen Position um 12:24:36 Uhr UTC durch schrittweise Betrachtung und Vergrößerung – zuerst der stündlichen und anschließend der minütlichen Aggregatstatistiken – ermittelt wurde. Kurz darauf, um 12:37:40 Uhr UTC fand sich das Tagesmaximum (ca. 237,6 V).

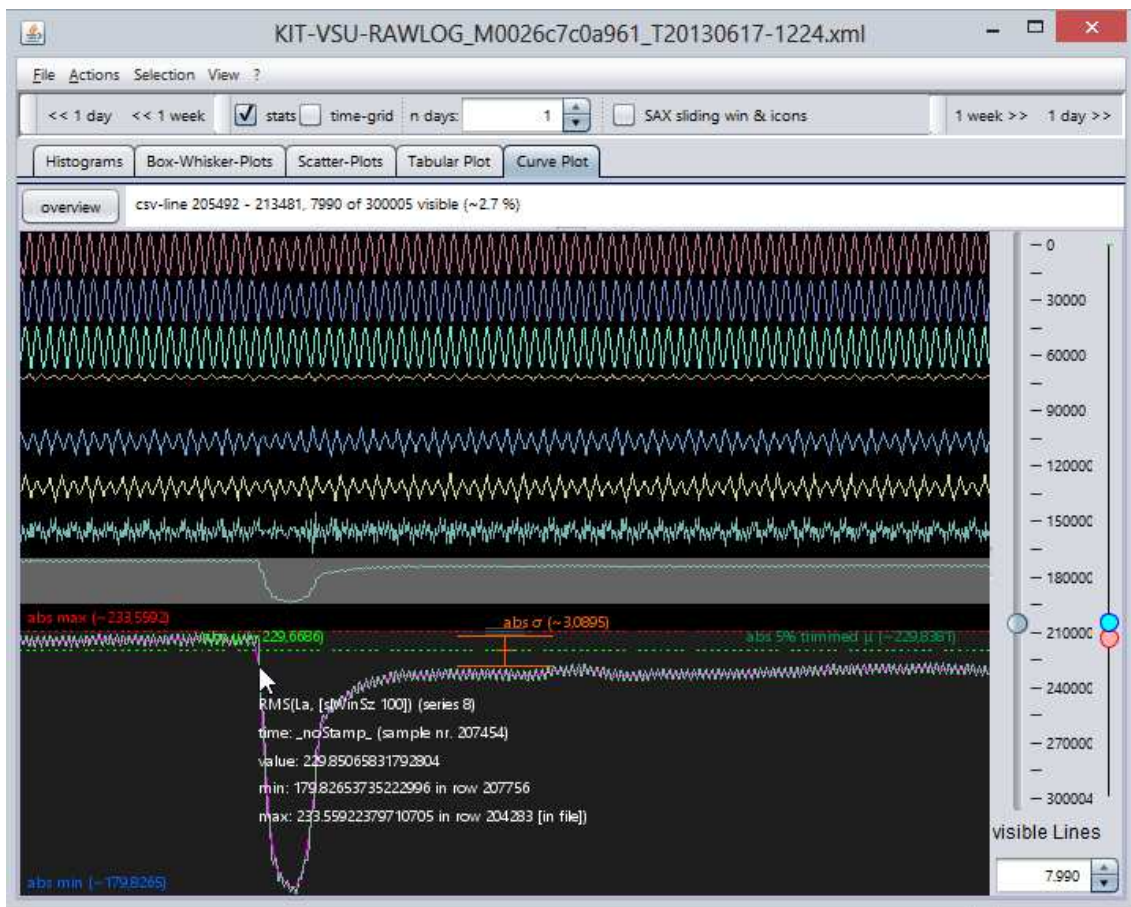
Eine Untersuchung der hochfrequenten Spannungsmessung, wiederum mit Erzeugung einer RMS-Kurve analog zum vorigen Fall (siehe Abbildung 5.23), zeigte einen enormen Spannungseinbruch, der in Abbildung 5.24 zu sehen ist und bei dem die Spannung der ersten Phase sich ab 12:24:41:491 Uhr UTC ca. 12 Sekunden lang talartig um etwa 77 % verringerte und sich dann über etwa 26 Sekunden (noch deutlich unterhalb des Mittelwerts) bis ca. 12:25:19:703 Uhr UTC wieder stufenartig einschwang.



**Abbildung 5.22:** Vergrößerter Ausschnitt (ca. 20 s) des Effektivwerts in den EDR-Merkmalen am selben Tag, welcher sowohl das Tagesminimum als auch das Tagesmaximum enthält und auf einen weiteren Spannungseinbruch hindeutet.



**Abbildung 5.23:** Überblicksdarstellung der erzeugten RMS-Kurve der ersten Spannungsphase über eine Minute (12:24 Uhr bis 12:25 Uhr UTC) mit weiterem Spannungseinbruch.



**Abbildung 5.24:** Beginn des talartigen Spannungseinbruchs gegen 12:24:41:491 Uhr UTC, unten mit Detailansicht der RMS-Kurve. Der Einbruch ist visuell in allen drei Phasen sichtbar (obere drei Kurven).

### 5.3.4 Bewertung

Das entwickelte *FraScaTi*-System wurde zur explorativen Analyse der EDR-Stromnetz-Messdaten angewandt. Hierzu wurden zwei Anwendungsfälle definiert: Die Detektion kurzer Spannungseinbrüche in EDR-Merkmalen (F1) und die Detektion von Transienten in EDR-Rohdaten (F2).

Bei der Bearbeitung von F1 zeigte sich das entwickelte *SortList*-Verfahren zur Detektion von Discords und Motifs als leistungsfähiges Werkzeug bei der Suche nach Transienten in großen Spannungs-Messdaten. Selbst kleinste Abweichungen (in den Testdaten *Tr\_4*), die keine lokalen Minima und Maxima aufwiesen und daher streng genommen keine Transienten darstellen, konnten noch zuverlässig als ungewöhnliche Muster in den langen Zeitreihen detektiert werden. Dabei wurden experimentell Parameter für das *SortList*-Verfahren ermittelt, die sehr effiziente Suchläufe bei gleichzeitig zuverlässiger Erkennung aller Transienten möglich machen.

Bei der Bearbeitung von F2 wurde das interaktive ViAT-Modul zur explorativen Analyse eines Spannungseinbruchs eingesetzt, dessen ungefährer Zeitpunkt aufgrund von Informationen der Abteilung Technische Infrastruktur und Dienste bekannt war. Die Flexibilität der interaktiven Handhabung ermöglichte es, schnell die entsprechende Position in den EDR-Merkmalen aufgrund von Auffälligkeiten im Verlauf der Merkmalsausprägungen zu finden und Merkmale zu identifizieren, die prinzipiell eine merkmalsbasierte Detektion von Spannungseinbrüchen erlauben. Außerdem konnten durch Exploration der hochfrequenten Spannungsdaten schnell der genaue Zeitpunkt sowie die Dauer des Ereignisses bestimmt werden. Ein weiterer, weitaus umfangreicherer Spannungseinbruch konnte aufgrund der Zeitbereichstatistiken gefunden werden, welche schnelles Auffinden und eine effektive Navigation zu statistisch auffälligen Bereichen in den Zeitreihendaten ermöglichen.

## 6 Zusammenfassung

Das Ziel der vorliegenden Arbeit bestand darin, ein neues Konzept zur skalierbaren explorativen Analyse großer Zeitreihendaten zu entwickeln, das sich auf am KIT gemessene Stromnetz-Messdaten anwenden lässt. Hierzu sollten Methoden, Verfahren und Datenrepräsentationen zur explorativen Analyse umfangreicher Zeitreihen unter dem Aspekt der Skalierbarkeit mit der Datenmenge ausgewählt und neu entwickelt werden. Ein Gesamtsystem zur effizienten Verwaltung und Analyse großer Zeitreihendaten sollte erarbeitet und evaluiert werden, welches neue Verfahren zur interaktiven Exploration und automatischen Musterdetektion mit Methoden der datenparallelen verteilten Verarbeitung kombiniert und in einem leistungsfähigen Framework integriert.

Hierzu wurde in Kapitel 2 ein neues Konzept zur skalierbaren Auswertung großer Zeitreihendaten vorgestellt. Es basiert auf der verteilten Speicherung der Rohdaten und der aufgrund der Berücksichtigung der Datenlokalität sehr effizienten datenparallelen Vorberechnung abgeleiteter Metadaten auf einem Hadoop-Cluster sowie damit verbundenen neuen und modifizierten explorativen Analysemethoden. Die abgeleiteten Metadaten beinhalten eine dimensionsreduzierte symbolische Zeitreihenrepräsentation (SAX), die ähnlichkeitsbasierte Analyseverfahren ermöglicht wie z. B. die Suche nach seltenen und häufigen Mustern (sog. *Discords* und *Motifs*). Die Analyse großer multivariater Zeitreihen wird in eine neuartige interaktive Visualisierungsumgebung eingebettet, die Nutzern unterschiedlicher Fachbereiche eine kollaborative und einfache Anwendung der komplexen Infrastruktur und der neuen Verfahren ermöglicht.

Die Umsetzung der neuen skalierbaren Methoden und die Integration als Gesamtsystem namens *FraScaTi* (von engl. **F**ramework for **S**calable **T**ime series **A**nalysis) wurden in Kapitel 3 beschrieben. Nach der Beschreibung der Voraussetzungen und der eingesetzten technischen Infrastruktur am Steinbuch Centre for Computing (SCC) des Karlsruher Instituts für Technologie (KIT) und einer Bewertung der Leistungsfähigkeit dieser Architektur wurden zunächst neu entwickelte Werkzeuge und Schnittstellen zur Datenorganisation und -Verarbeitung vorgestellt. Anschließend wurden die neue interaktive Zeitreihenexplorationsumgebung namens *ViAT* und neue integrierte SAX-basierte Analysewerkzeuge sowie eine angepasste Volltextsuche in den Metadaten vorgestellt.

In Kapitel 4 wurde die Funktions- und Leistungsfähigkeit für die einzelnen Verfahren und das Gesamtsystem *FraScaTi* sowie die Skalierbarkeit mit großen Datenmengen experimentell evaluiert und nachgewiesen. Der *SAX-Sequenzeditor* bietet neue, nützliche Analysemöglichkeiten und ermöglicht eine schnelle interaktive Suche nach zuvor unbekanntem Mustern, die im Vergleich mit dem einzigen ähnlichen Verfahren namens *VizTree* [179] Auftrittshäufigkeiten für wesentlich

längere Teilsequenzen übersichtlich visualisieren kann. Die Einbettung modifizierter SAX-Sequenz-Histogramme bietet dabei eine zusätzliche Visualisierung der Häufung von Sequenzen. Zur *Detektion von Discords und Motifs* wurde gezeigt, dass der modifizierte Algorithmus *SortList* bei vergleichbarer Leistungsfähigkeit eine geringere Laufzeitkomplexität aufweist und dadurch im Vergleich zur Variante HOTSAX [143] wesentlich schneller arbeitet (ca. um Faktor 1200), was erstmals eine automatische Detektion von Discords und Motifs ermöglicht, die für eine interaktive Visualisierung geeignet ist. Die Strukturanalyse *MetaSAX* erlaubt eine neue Strukturbeschreibung großer Zeitreihendaten auf abstraktem Niveau und liefert direkte semantische Aussagen über die Zeitreihe, die zur Ähnlichkeitsbewertung (z. B. Klassifizierung, Clustering, inhaltsbasierte unscharfe Suche) hilfreich sind. Für die datenparallele Verarbeitung auf dem Hadoop-Cluster wurde die Performanz der verteilten Berechnung und die hohe Kompressionsrate resultierender Metadaten (ca. 0,75 % gegenüber Rohdaten) aufgezeigt, die eine Visualisierung und explorative Analyse auf einem lokalen Arbeitsplatz-Rechner erlaubt, wohingegen die Rohdaten bisher nur verteilt gespeichert und analysiert werden können. Die Visualisierung mit *ViAT* ist hochperformant und ermöglicht u. a. durch die eingeführten Ansichten und adaptiven Zeitbereichstatistiken eine Exploration großer Mengen an Datenpunkten, deren Anzahl die zur Verfügung stehenden Monitor-Pixel übersteigt.

Anhand der Anwendung des Systems auf umfangreiche Stromnetz-Messdaten des KIT wurde in Kapitel 5 die Nützlichkeit des *FraScaTi*-Systems im Themenfeld der Stromnetzanalyse demonstriert. Dabei wurde die Erhebung der Stromnetzdaten am KIT durch am Institut für Angewandte Informatik (IAI) entwickelte, im KIT-Stromnetz verteilte Messgeräte (*Electronic Data Recorder*, EDR) erläutert. Zur Anwendung von *FraScaTi* wurde eine neu entwickelte Organisations- und Datenselektions-Software für EDR-Daten vorgestellt. Anhand einiger Anwendungsfälle wurde *FraScaTi* erfolgreich eingesetzt, um transiente Spannungseinbrüche in EDR-Rohdaten und EDR-Merkmalen zu detektieren. Dabei zeigte sich das entwickelte *SortList*-Verfahren zur Detektion von Discords und Motifs als leistungsfähiges Werkzeug, das selbst kleinste Abweichungen, die keine lokalen Minima und Maxima aufwiesen und daher streng genommen keine Transienten darstellen, noch zuverlässig als ungewöhnliche Muster in den langen Zeitreihen detektieren kann. Dabei wurden experimentell Parameter für das *SortList*-Verfahren ermittelt, die sehr effiziente Suchläufe bei gleichzeitig zuverlässiger Erkennung aller Transienten möglich machen.

Das interaktive *ViAT*-Modul wurde u. a. zur explorativen Analyse eines Spannungseinbruchs eingesetzt, dessen ungefährer Zeitpunkt aufgrund von Informationen der Abteilung Technische Infrastruktur und Dienste bekannt war (Fall F2). Die Flexibilität der interaktiven Handhabung ermöglichte es, schnell die entsprechende Position in den EDR-Merkmalen aufgrund von Auffälligkeiten im Verlauf der Merkmalsausprägungen zu finden und Merkmale zu identifizieren, die prinzipiell eine merkmalsbasierte Detektion von Spannungseinbrüchen erlauben. Außerdem konnten durch Exploration der hochfrequenten Spannungsdaten schnell der genaue Zeitpunkt



sowie die Dauer des Ereignisses bestimmt werden. Ein weiterer, weitaus umfangreicherer Spannungseinbruch konnte aufgrund der Zeitbereichstatistiken gefunden werden, welche schnelles Auffinden und eine effektive Navigation zu statistisch auffälligen Bereichen in den Zeitreihendaten ermöglichen.

Die wesentlichen Ergebnisse der Arbeit sind:

1. Entwicklung eines neuen Konzeptes zur skalierbaren explorativen Analyse großer Zeitreihendaten
2. Ableitung von Verfahren zur Verwaltung und Verarbeitung großer Zeitreihendaten zur
  - 2.1. Skalierbaren Speicherung und Verwaltung von Zeitreihendaten auf verteilter Infrastruktur
  - 2.2. Datenparallele „verteilte“ Auswertung großer Zeitreihendaten im Hadoop Cluster
3. Aufbau von Schnittstellen zum verteilten Dateisystem HDFS
  - 3.1. Neuer spezieller Dateiexplorer mit effizienter Navigation in großen Dateihierarchien und vielen Einzeldateien, Datentransfer- und -manipulationsfunktionen
  - 3.2. Schnelles Einlesen verteilter Daten zur direkten interaktiven Visualisierung
4. Herleitung neuer skalierungsfähiger Verfahren und Methoden zur Zeitreihenanalyse und zur SAX-basierten Ähnlichkeitsbewertung und -Darstellung
  - 4.1. Interaktiv parametrisierbare SAX-Codierung mit automatischer Visualisierung
  - 4.2. SAX-basiertes Clustering mit Dendrogramm-Darstellung
  - 4.3. SAX-Strukturcharakteristik-Editor *MetaSAX* für abstrakte Strukturbeschreibungen
  - 4.4. Visualisierungsgestütztes interaktives Subsequenz-Mining (*SAX-Sequenzeditor*)
  - 4.5. Modifizierte SAX-Sequenz Histogramme („intelligente Icons“)
5. Interaktive Werkzeuge für synchronisierte multivariate Zeitreihenvisualisierungen und explorative Analysen (Visualisierungsmodul *ViAT*)
  - 5.1. Einheitliche Visualisierungen und Analysen für unterschiedliche Datentypen
  - 5.2. Skalierbare interaktive Kurvendarstellung zur Darstellung umfangreicher Zeitreihendaten mit erweitertem Funktionsumfang
  - 5.3. Tabellarische Farbdarstellung zur gleichzeitigen Parallel-Darstellung vieler Zeitreihen
  - 5.4. Interaktiv parametrisierbare Box-Whisker-Plots zur aggregierten Verlaufsdarstellung
  - 5.5. Scatterplots zur interaktiven Korrelationsanalyse mit Optionen zum zeitlichen Versatz
  - 5.6. Interaktive Sliding Window basierte SAX-Codierung direkt in der Visualisierung
  - 5.7. Zeitreihenhistogramme mit farblicher Verlaufsdarstellung, welche ein Nachvollziehen zeitlicher Abläufe im Histogramm ermöglicht
6. Ableitung eines modifizierten Algorithmus zur Schnell-Detektion seltener und häufiger Muster (Discords, Motifs) mit verbesserter Leistungsfähigkeit (Laufzeitverkürzung um Faktor 1200) bei vergleichbarer Funktionsfähigkeit im Anwendungsbereich, was einen Einsatz in interaktiven Visualisierungen erstmals ermöglicht (*SortList*).

7. Umsetzung des neuen Konzepts auf einer verteilten Hadoop Rechencluster-Architektur
  - 7.1. Experimentelle Performanzuntersuchung, Vergleich mit Mehrkernarchitekturen, Nachweis der besseren Skalierbarkeit des datenparallelen Ansatzes und der Eignung für die Aufgabenstellung
  - 7.2. Schnittstellen zur verteilten Data-Intensive Computing (DIC) Umgebung
  - 7.3. Neue Client-Anwendung zur Darstellung, Suche und Exploration von Daten und Zeitreihen mit Schnittstellen zum Hadoop Cluster, mit integrierten Analyse- und Visualisierungsfunktionen
8. Evaluation der Einzelkomponenten und des Analyseframeworks *FraScaTi* mit
  - 8.1. Nachweis der jeweiligen Funktionsfähigkeit sowie
  - 8.2. Nachweis der Leistungsfähigkeit der neuen Verfahren und von *FraScaTi*
9. Anwendung des Analyseframeworks *FraScaTi* auf Stromnetz-Messdaten des KIT
  - 9.1. Erfolgreiche Detektion von Transienten durch Analyse mit neuem *SortList-Algorithmus* und Nachweis der Vorteile im Vergleich mit aktuellen Normverfahren
  - 9.2. Ermittlung eines bereits bekannten Spannungseinbruchs mit dem Analyseframework *FraScaTi* mit verbesserter Zeitgenauigkeit und Ableitung von Suchkriterien
  - 9.3. Detektion eines neuen, zuvor unbekanntem Spannungseinbruchs durch Anwendung der explorativen Analyse.

Die Verfahren und Methoden, die in dieser Arbeit präsentiert wurden, eröffnen neue Möglichkeiten der explorativen Analyse großer Zeitreihendaten. Das Potenzial, das in den neuen Analysemöglichkeiten steckt, kann durch eine Weiterentwicklung der Ansätze noch erweitert werden. Außerdem sind Anwendungen in vielen Bereichen möglich, die in dieser Arbeit nicht berücksichtigt wurden. Deshalb werden im Folgenden Vorschläge für Optimierungen, Weiterentwicklungen und Anwendungen gemacht, welche in zukünftigen Forschungsarbeiten erarbeitet werden können.

Ein wichtiges Forschungsthema besteht in der Weiterentwicklung verteilter datenparalleler Verarbeitungsarchitekturen wie Apache Hadoop, was die stark zunehmende Einführung solcher Systeme in vielen wissenschaftlichen Rechenzentren, aber auch zum unternehmerischen Produktiv-einsatz zeigt. Neue Architekturen und Dateisysteme erweitern die Kompatibilität unterschiedlicher Systeme, z. B. zur besseren Integration mit verschiedenen Dateisystemen und HPC-Clustern (z. B. *Hadoop on Lustre* von Intel). Es werden zunehmend vielversprechende Alternativen zu *Hadoop MapReduce* vorgestellt (z. B. *Apache Spark*) oder auf *Hadoop MapReduce* basierende neue Softwareframeworks für zusätzliche Verarbeitungsansätze und Analysesysteme. An die sich daraus ergebenden neuen Verarbeitungsmöglichkeiten können die hier vorgestellten Verfahren und Methoden angepasst werden, um die Skalierbarkeit von Datenanalysen, insbesondere skalierbare Zeitreihenanalysen, voranzutreiben. In der vorliegenden Arbeit wurden neue Methoden für un-

scharfe ähnlichkeitsbasierte Suchen und Musterdetektionen eingeführt bzw. modifiziert und weiterentwickelt, die auf einer symbolischen Repräsentation von Zeitreihen (SAX) basieren. Sie bieten große Vorteile gegenüber klassischen Methoden, welche auf reellwertigen Zeitreihen operieren, da sie Analysen auf einem wesentlich höheren Abstraktionsniveau ermöglichen. Die Verfahren wurden parallel zu den klassischen Ansätzen entwickelt und nutzen Mittel aus unterschiedlichsten Bereichen wie *Data-Mining*, *Bioinformatik* (z. B. *DNS-Sequenzanalyse*) sowie *Text-* und *Content-Mining*, um neue Verfahren zur Auswertung großer Datenmengen abzuleiten. Die neuen Verfahren stellen keine Konkurrenz zu den wohldefinierten und gut erforschten klassischen Zeitreihenanalysemethoden dar, sondern erweitern den Möglichkeitsraum zur Durchführung neuartiger Analysen. In diesem Umfeld steckt enormes Potenzial für Weiterentwicklungen und Kombinationen mit den klassischen Methoden. Die symbolische Repräsentation bietet neue Möglichkeiten zur automatischen Ähnlichkeitsbewertung der Zeitreihencharakteristik und -Struktur und zur Musterdetektion, die den intuitiven Fähigkeiten des Menschen näher kommen als alle anderen Verfahren. In diesem Bereich bestehen z. B. Möglichkeiten zur Entwicklung semantischer Zeitreihensegmentierungsverfahren, zu der in der vorliegenden Arbeit bereits vielversprechende strukturorientierte Methoden entwickelt wurden, sowie zur Erweiterung der automatischen Ähnlichkeitserkennung. Ein Gedanke, der während der Entwicklung von *FraScaTi* häufig aufkam, aber nicht weiterverfolgt werden konnte, war dass auch „teilweise inverse“ Ähnlichkeit in Auswertungen und Suchen eingehen sollte. Klassische Verfahren werten „synchrone“ Zeitreihen, die an ähnlichen Stellen ihr Verhalten ändern, als stark unterschiedlich, sofern das neu auftretende Verhalten nicht an allen Stellen übereinstimmt. Bei unscharfen Suchen nach Zusammenhängen zwischen unterschiedlichen Zeitreihen ist synchrones Verhalten jedoch von enormem Interesse, um Korrelationen zu identifizieren, auch wenn z. B. Teilabschnitte mit Trends unterschiedlicher Ausrichtung enthalten sind.

Speziell bei der Analyse multivariater Zeitreihen können neue Methoden zur Ähnlichkeitsbewertung von Zeitabschnitten erarbeitet werden, welche die Zuverlässigkeit der Detektion von Auffälligkeiten enorm verbessern können. Hierzu müssen parametrisierbare Verfahren entwickelt werden, welche anwendungsspezifische Gewichtungen der einzelnen Merkmale erlauben. Im Anwendungsbereich der Stromnetzanalyse sind zahlreiche Weiterentwicklungen und Anpassungen für *FraScaTi*, insbesondere für *ViAT* denkbar. Hierzu sollte ein Plugin-System eingeführt werden, das eine Integration anwendungsspezifischer Visualisierungen und Auswertungen vereinfacht, die z. B. Grenzwertüberschreitungen und bekannte bedeutsame Muster und Kurvenverläufe visuell hervorhebt. Gerade Verfahren wie die vorgestellte Detektion von Transienten haben enormes Optimierungspotential, wenn anwendungsspezifisches Vorwissen im Analyseprozess berücksichtigt wird. Eine zusätzliche nützliche Funktion in diesem und anderen Bereichen ist eine anwendungsspezifische Variante der beschriebenen datenparallelen Metadaten-Erzeugung. Die Metadaten könnten dann im Bereich der Stromdatenanalyse zusätzlich zu den statistischen Zu-

sammenfassungen Positionen mit Grenzwertüberschreitungen, detektierte Transienten, Zeitabschnitte mit auffälligen Spannungsänderungen und Flicker, Spannungsunsymmetrien, auffälligen Oberschwingungen usw. beinhalten. Zu den möglichen Anwendungsbereichen des *FraScaTi*-Systems und der vorgestellten Methoden zählen alle Bereiche, in denen große Zeitreihen- bzw. Sequenzdaten anfallen, wie z. B. bei der Auswertung groß angelegter physikalischer Experimente, astronomischer Beobachtungsdaten und Messungen, Wetter- und seismische Daten etc., aber auch die Analyse von Daten, die bereits in symbolischer Form vorliegen, wie z. B. DNS-Sequenzen, Texte (Plagiatserkennung, semantische Zusammenhänge etc.), Quellcode u. v. m.

# Abbildungsverzeichnis

<b>Abbildung 1.1:</b>	<i>Frühe Kurvendarstellung der Inklinationen planetarer Orbits aus dem zehnten Jahrhundert n. Chr., laut E. Tuftes das älteste bekannte Beispiel eines Versuches, Werte graphisch darzustellen. (Quelle: [296], Original: H. Gray Funkhouser, „A Note on a Tenth Century Graph“, Osiris (1936), S. 260-262).....</i>	7
<b>Abbildung 2.1:</b>	<i>Konzept des Gesamtsystems –Komponenten der verteilten Datenverarbeitung, Datenflüsse und Benutzerschnittstellen (eigene Arbeiten rot hervorgehoben).....</i>	18
<b>Abbildung 2.2:</b>	<i>Konzept zur Datenreduktion großer Messzeitreihen und Ableitung durchsuchbarer Metadaten.....</i>	19
<b>Abbildung 2.3:</b>	<i>Automatische Metadatenextraktion und manuelle Suchanfragen.....</i>	20
<b>Abbildung 2.4:</b>	<i>Codierung einer Kurve in SAX-Symbole anhand eines Beispiels. Quelle: Eigene Darstellung, in Anlehnung an [52]. .....</i>	204
<b>Abbildung 2.5:</b>	<i>a) Idealisierendes ungewöhnliches Muster (Discord) in periodischer Zeitreihe und b) wiederkehrende Muster (Motifs). Identisch zu Abbildung A.3. Quelle: [79], modifiziert.....</i>	207
<b>Abbildung 3.1:</b>	<i>Vergleich der Laufzeiten mit Pig (Hadoop) und mit zwei Mehrkernrechnern.....</i>	39
<b>Abbildung 3.2:</b>	<i>Oberfläche mit Dateihierarchie des entwickelten HDFS-Explorers.....</i>	42
<b>Abbildung 3.3:</b>	<i>Darstellung eines EDR-Tagesordners für den 01.01.2015 im HDFS-Explorer mit hyperbolischer Baumdarstellung der Dateihierarchie. (Screenshot).....</i>	44
<b>Abbildung 3.4:</b>	<i>Vorgeschlagene hyperbolische Baumdarstellung der Dateihierarchie im HDFS-Explorer, links: EDR-Jahresordner 2015 mit drei Monaten, rechts: EDR-Monatsordner für Februar 2015.....</i>	45
<b>Abbildung 3.5:</b>	<i>Interaktive Darstellung langer Dateilisten als Fischaugen-Menü im HDFS-Explorer... ..</i>	46
<b>Abbildung 3.6:</b>	<i>Darstellung der Ableitungs- (rote Doppelpfeile links) und Verweiskette (einfache graue Pfeile rechts) bei der Erzeugung von XML-Metadaten aus EDR-Stromnetz-Messdaten (oben).....</i>	48
<b>Abbildung 3.7:</b>	<i>Beispielhafter kommentierter Ausschnitt einer Metadaten-datei im XML-Format (SAX-Strings teilw. gekürzt).....</i>	49
<b>Abbildung 3.8:</b>	<i>Zeitreihenvisualisierung in ViAT mit ausgewählter Kurvendarstellung, kommentierter Bildschirmabzug.....</i>	55
<b>Abbildung 3.9:</b>	<i>Ausschnitt einer multivariaten Kurvenansicht mit einem Ereignis, das im Verlauf fast aller Kurven sichtbar ist.....</i>	57
<b>Abbildung 3.10:</b>	<i>Multivariate Tagesübersicht über EDR-Merkmalen mit gestapelter Kurvendarstellung und aktiviertem Zeitraster.....</i>	58
<b>Abbildung 3.11:</b>	<i>Zwei Kurvendarstellungen derselben Zeitreihe: Mit Min-Max Skalierung (oben) ist die Kurvencharakteristik aufgrund zahlreicher Ausreißer kaum erkennbar, jedoch mit <math>\mu \pm 2\sigma</math> Skalierung (unten).....</i>	59
<b>Abbildung 3.12:</b>	<i>Erzeugung eines neuen Benutzerkommentars zur aktuellen Bereichsauswahl über das Kontextmenü.....</i>	60
<b>Abbildung 3.13:</b>	<i>Auswahldialog über sichtbare Kurven.....</i>	61
<b>Abbildung 3.14:</b>	<i>Auswahl der durchzuführenden Transformation bei der Erzeugung einer neuen Zeitreihe. Eigene Darst.....</i>	61

<b>Abbildung 3.15:</b> Darstellung der ersten Phase der Spannungsmessung (obere Kurve) und des erzeugten RMS (untere Kurve), in welchem sich deutlich ein Spannungseinbruch zeigt, der in der oberen Originalkurve nicht zu sehen ist. ....	63
<b>Abbildung 3.16:</b> Berechnung und Darstellung eines gleitenden Mittelwertes, mit Dialog zur Auswahl des Summationsbereichs (rechts oben), Kurve mit überlagertem gleitenden Mittelwert (pink, unten) und Detaildarstellung der beiden Kurven (links oben).....	63
<b>Abbildung 3.17:</b> Beispiel einer Eingabemaske für Nutzerkommentare zu einer Bereichs- auswahl – hier für Ereignisse in EDR-Merkmalsdateien, mit bereits automatisch vorausgefüllten Feldern. ....	70
<b>Abbildung 3.18:</b> Multivariate tabellarische Farbflächendarstellung in ViAT, a) mit Grauwerten, b) mit Blau-Rot- und c) mit Grün-Rot-Verlauf. ....	72
<b>Abbildung 3.19:</b> Box-Whisker-Plot der unterschiedlichen Wertebereiche von sieben Zeitreihen (hier EDR-Rohdaten mit Spannung und Strom in je drei Phasen und Strom im Neutralleiter). ....	73
<b>Abbildung 3.20:</b> Box-Whisker Zeitreihendarstellung der harmonischen Gesamtverzerrung (THD) über einen Messtag mit 16 Zeitbereichen (oben) im Vergleich zur Kurvendarstellung mit stündlichen Zeitbereichsstatistiken (unten). ....	74
<b>Abbildung 3.21:</b> Streudiagramm zur Visualisierung des Zusammenhanges zwischen der prozentualen zweiten Oberschwingung (OW2) und der harmonischen Gesamtverzerrung (THD) der Spannung aus EDR-Merkmalssdaten für einen Tag, ohne Zeitversatz. ....	75
<b>Abbildung 3.22:</b> Streudiagramm als Lag-Plot der vierten harmonischen Schwingung, mit zeitlichem Versatz von acht Einzelwerten für auf der vertikalen Achse aufgetragene Werte. Die diagonal verteilte Streuung deutet auf eine nicht-zufällige Wertefolge hin.....	76
<b>Abbildung 3.23:</b> Zeit-Histogramm über 24 Stunden der neunten Oberschwingung (OW9) der Spannung einer EDR-Merkmalssdatei mit Wertebereichsaufteilung in 64 Bins und zeitlicher Farbkodierung durch 24 Farbstufen.....	78
<b>Abbildung 3.24:</b> Effekt der Zeitbereichsverschiebung von Beginn (oben, rot) nach Ende (unterstes Bild, blau) der neunten Oberschwingung auf das Bereichs-Histogramm, als Bildsequenz. ....	79
<b>Abbildung 3.25:</b> Dialog zur SAX-Codierung, mit Vergrößerungsansicht der Kontrollelemente zur Parametereinstellung (oben) und der farbcodierten Textanzeige des Resultats (rechts). ....	81
<b>Abbildung 3.26:</b> Dendrogramm des SAX-basierten hierarchischen Clustering nach Ähnlichkeit zwischen den einzelnen Merkmalszeitreihen aus einem EDR-Merkmalssdatensatz für einen Messtag. ....	84
<b>Abbildung 3.27:</b> Dendrogramm des SAX-basierten „vertikalen“ Clusterings von Zeitbereichen der Merkmalszeitreihen aus einem EDR-Merkmalssdatensatz für einen Messtag. ....	85
<b>Abbildung 3.28:</b> Dialog zur Auswahl des Clustering-Typs. ....	86
<b>Abbildung 3.29:</b> Erläuterung der SAX-Sequenzbildung als Eingangsdaten für ein hierarchisches Clustering, mit markierter Zeile für Zeitreihen und markierter Spalte für multivariate Zeitbereiche.....	86
<b>Abbildung 3.30:</b> Text-Editor zur Visualisierung SAX-basierter Strukturanalysen, vor der Segmentierung der SAX-Sequenz. ....	87
<b>Abbildung 3.31:</b> Textuelle Sequenzdarstellung mit Hervorhebung der Segmente durch Farbkodierung, Hoch- bzw. Tiefstellung und Unter- bzw. Durchstreichung des Textes. ....	89

<b>Abbildung 3.32:</b> Darstellung der ermittelten Strukturcharakteristik aus einer SAX-Sequenz (oben) und gegenübergestellte Ansicht der Originalkurve (unten).....	89
<b>Abbildung 3.33:</b> Musterselektionstabelle des SAX-Sequenzeditors (unten, blaue Linien markieren die Selektion) und Textfeldern für Suchmuster und Gesamtsequenz (oben), hier mit Suchsequenzlänge acht und Alphabetgröße vier.....	94
<b>Abbildung 3.34:</b> SAX-Sequenzeditor mit Suchsequenzlänge acht und Alphabetgröße vier, hier mit eingeblendeten Teilsequenzhistogrammen (oben).....	95
<b>Abbildung 3.35:</b> Teilsequenz bbcd <sub>b</sub> der Suchsequenz (oben) und durchsuchte Sequenz (unten, mit rot markierter Fundstelle an Position 9) entsprechend Abbildung 4.34. Die Auftrittshäufigkeit der Teilsequenz bbcd <sub>b</sub> der Suchsequenz bestimmt die Färbung der Zelle in Zeile b, Spalte 5 im Editor.....	95
<b>Abbildung 3.36:</b> Farbcodierung der normalisierten Auftrittshäufigkeiten im SAX-Sequenzeditor.....	96
<b>Abbildung 3.37:</b> Erweitertes Intelligentes Icon mit textueller und Kurvendarstellung der Sequenzen, hier mit Alphabetgröße vier und Teilsequenzlänge drei.....	98
<b>Abbildung 3.38:</b> Suche nach Discords und Motifs über das ViAT-Kontextmenü.....	99
<b>Abbildung 3.39:</b> Dialog zur Einstellung der Suchparameter für eine Detektion von Discords bzw. Motifs.....	99
<b>Abbildung 3.40:</b> Darstellung der Testdaten zur Detektion von Discords und Motifs in ViAT mit je einem Discord im markierten Bereich.....	100
<b>Abbildung 3.41:</b> ViAT-Darstellung detektierter Discords (rot, pink) und Motifs (grün, blau) in den sinusförmigen Testdaten (mit zusätzlicher Markierung von Elementen in orange). .....	101
<b>Abbildung 3.42:</b> Detaillierte Darstellung der Region mit detektierten Discords.....	102
<b>Abbildung 3.43:</b> Darstellung der in allen Test-Zeitreihen detektierten Discords in der Kurvendarstellung (oben) und in der tabellarischen Farbfelddarstellung (unten, zusammen mit den Motifs).....	105
<b>Abbildung 4.1:</b> Schematische Darstellung des fertigen Gesamtsystems für skalierbare Zeitreihenanalysen (eigene Arbeiten rot markiert).....	110
<b>Abbildung 4.2:</b> Ableitung von SAX-Teilsequenzen mit einem Fensterverschiebungsalgorithmus.....	113
<b>Abbildung 4.3:</b> Dialog zur Eingabe der gemeinsamen Suchparameter für die Detektionsvarianten HOTSAX und SortList.....	116
<b>Abbildung 4.4:</b> Suchergebnis unter Verwendung der Parameter aus Abbildung 4.3 mit der HOTSAX-Implementierung (oben) und der SortList-Implementierung (unten).....	117
<b>Abbildung 4.5:</b> Suchergebnis unter Beibehaltung der Parameter, mit der Implementierung SortList, ohne Ausschluss aufeinanderfolgender identischer Teilsequenzen.....	118
<b>Abbildung 4.6:</b> Resultate der Suchläufe in den erweiterten Testdaten T1_lang (Länge: 88141 Samples), mit HOTSAX (links) und SortList (rechts).....	119
<b>Abbildung 4.7:</b> Vergrößerter Ausschnitt des SortList-Resultats mit zwei nicht überlappenden (links) und zwei überlappenden Discords (rechts, rot).....	121
<b>Abbildung 4.8:</b> Diagramm zum Laufzeitvergleich zwischen HOTSAX (grüne Kurve) und SortList (blaue Kurve) aufgrund gemittelter Laufzeiten über drei Testdurchläufe pro Testdatensatz, mit linearer vertikaler Zeitskala (Diagramm oben) und logarithmischer Zeitskala (Diagramm unten), sowie einer tabellarischen Listung von Eigenschaften der jeweiligen Testdatensätze (unten).....	123

<b>Abbildung 4.9:</b>	<i>Resultate der Suchen in den Testdaten T2 (oben) und T3 (unten) mit dem SortList-Ansatz mit zwei bzw. vier detektierten Discords (die Anzeige von Motifs wurde der Übersicht halber deaktiviert). Eine entsprechende Suche mit dem HOTSAX-Ansatz detektierte keine Discords. Die Suchparameter wurden nicht verändert und entsprechen Abbildung 4.3. ....</i>	<i>124</i>
<b>Abbildung 4.10:</b>	<i>ViAT-Kurvendarstellung der in den EDR-Merkmalmetadaten enthaltenen SAX-Repräsentation.....</i>	<i>128</i>
<b>Abbildung 4.11:</b>	<i>Datenstruktur eines einzelnen datenparallel erzeugten Metadatensatzes über eine EDR-Merkmaldatei der dritten Spannungsphase. ....</i>	<i>129</i>
<b>Abbildung 4.12:</b>	<i>Performanzindikatoren bei Anzeige von X_64 in ViAT (Bildschirmabzug aus dem Prozessmanager ProcessExplorer von Sysinternals). (Screenshot).....</i>	<i>131</i>
<b>Abbildung 4.13:</b>	<i>Beispiel einer Unterabtastung bei der Kurvendarstellung in ViAT, bei der ohne Darstellung von Zeitbereichsstatistiken kurze Ereignisse verborgen bleiben, was dadurch ersichtlich ist, dass die Kurve nicht an das absolute Maximum (rote gepunktete Linie) anschließt. ....</i>	<i>132</i>
<b>Abbildung 4.14:</b>	<i>Beispiel einer Unterabtastung bei der Kurvendarstellung in ViAT mit verborgenden lokalen Maxima, die durch Einblendung von Zeitbereichsstatistiken sichtbar werden (z.B. bei blauem Pfeil). ....</i>	<i>133</i>
<b>Abbildung 4.15:</b>	<i>Vergrößerter Ausschnitt der zweiten Minute der ersten Stunde der Beispielkurve mit klar identifizierbarer Herkunft des zuvor aufgefallenen stündlichen Maximums.....</i>	<i>134</i>
<b>Abbildung 5.1:</b>	<i>Neue Ordnerhierarchie zur Ablage der EDR-Daten. ....</i>	<i>145</i>
<b>Abbildung 5.2:</b>	<i>Werkzeug zur erweiterten EDR-Datenauswahl. ....</i>	<i>146</i>
<b>Abbildung 5.3:</b>	<i>Kalender-Ansicht über einen Auswahlzeitraum von sechs Tagen zur Prüfung der Verfügbarkeit von EDR-Daten innerhalb der Zeitspanne, mit Farbcodierung der Messgeräte und Helligkeitscodierung einzelner Kanäle. ....</i>	<i>147</i>
<b>Abbildung 5.4:</b>	<i>Dialog für die Weiterverarbeitung und Analyse ausgewählter Daten. ....</i>	<i>148</i>
<b>Abbildung 5.5:</b>	<i>Darstellung der jeweiligen Periode mit den eingebrachten Transienten (Trans1 links und Trans2 rechts, rot markiert) in den Testdaten Tr_1 bis Tr_4, mit Vergrößerung von Trans1 in Tr_1. ....</i>	<i>151</i>
<b>Abbildung 5.6:</b>	<i>Parameterdialog mit experimentell ermittelten Parametern, welche eine fehlerfreie Detektion der Transienten Trans1 und Trans2 in allen Testdaten ermöglichen. ....</i>	<i>152</i>
<b>Abbildung 5.7:</b>	<i>Detektionsresultate für alle Datensätze Tr_1 bis Tr_4, links: Übersicht mit je zwei detektierten Transienten (rot), rechts Vergrößerung des durch Trans1 verursachten Discords (roter Kurvenabschnitt mit Umrahmung).....</i>	<i>153</i>
<b>Abbildung 5.8:</b>	<i>Balkendiagramm zum Vergleich der resultierenden Merkmalswerte für Auswertung von Tr_1 und denselben Testdaten ohne Transienten mit M_50P – Auffällige Unterschiede zeigen Stabw Ueff, Stabw Uss und OW12 [%]. Quelle: Eigene Darstellung.....</i>	<i>155</i>
<b>Abbildung 5.9:</b>	<i>Merkmalskurven der Auswertung von Tr_1 mit M1P in ViAT, mit Markierung der Bereiche mit Transienten (rot) und Bezeichnungen auffälliger Merkmale (weißer Text links). ....</i>	<i>156</i>
<b>Abbildung 5.10:</b>	<i>Übersichtsdarstellung des Auswertungsergebnisses von Tr_4 durch M_1P in ViAT ohne sichtbare Ausschläge (auch bei extremer Vergrößerung). Quelle: Eigene Darstellung.....</i>	<i>157</i>



<b>Abbildung 5.11:</b> Übersicht über die EDR-Merkmalzeitreihen im Zeitraum des bekannten Spannungseinbruchs mit roter Umrandung einiger auffälliger Merkmalsverläufe. .....	158
<b>Abbildung 5.12:</b> EDR-Spannungs-Merkmale mit signifikanter Änderung zum Zeitpunkt des bekannten Spannungseinbruchs (13:20:57 Uhr bis 13:22:45 Uhr). ....	160
<b>Abbildung 5.13:</b> Ausschnitt der THD-Kurve in ViAT mit minütlichen Aggregatstatistiken (Max rot, Min blau), das lokale Maximum während des Spannungseinbruchs ist kleiner als das absolute Maximum. ....	161
<b>Abbildung 5.14:</b> Balkendiagramm zum Vergleich der Tages-Maximalwerte der Oberschwingungen (blau: Tag des Spannungseinbruchs, orange: Vortag), welche zur Zeit des Spannungseinbruchs ein Tagesmaximum aufweisen. ....	162
<b>Abbildung 5.15:</b> Aufruf der Anzeige der Spannungs-Messdaten im selektierten Zeitbereich über das Kontextmenü von ViAT. ....	163
<b>Abbildung 5.16:</b> Erzeugung und Anzeige zusätzlicher Zeitreihen über das Kontextmenü von ViAT. ....	164
<b>Abbildung 5.17:</b> Überblicksdarstellung über eine Minute an Spannungs-Messdaten (unten Detaildarstellung der ersten Phase). ....	164
<b>Abbildung 5.18:</b> Dialoge zur Erzeugung neuer Zeitreihen in ViAT, Auswahl der Zeitreihe(n) aus der/denen neue Daten erzeugt werden sollen (links) und Auswahl der Berechnung und der Parameter (rechts). ....	165
<b>Abbildung 5.19:</b> Ermittlung der Messwerte einer Periode in ViAT. ....	165
<b>Abbildung 5.20:</b> Kurvendarstellung der über eine Fenstergröße von 100 Messwerten (eine Periode) berechneten RMS-Zeitreihe. ....	166
<b>Abbildung 5.21:</b> Detailausschnitt der erzeugten RMS-Kurve der ersten Phase (türkis) in ViAT mit zusätzlich erzeugter Kurve eines gleitenden Mittelwerts (pink) und textueller Positionsinformation unterhalb des Mauszeigers (weiß). ....	167
<b>Abbildung 5.22:</b> Vergrößerter Ausschnitt (ca. 20 s) des Effektivwerts in den EDR-Merkmalen am selben Tag, welcher sowohl das Tagesminimum als auch das Tagesmaximum enthält und auf einen weiteren Spannungseinbruch hindeutet. ....	168
<b>Abbildung 5.23:</b> Überblicksdarstellung der erzeugten RMS-Kurve der ersten Spannungsphase über eine Minute (12:24 Uhr bis 12:25 Uhr UTC) mit weiterem Spannungseinbruch. .....	168
<b>Abbildung 5.24:</b> Beginn des talartigen Spannungseinbruchs gegen 12:24:41:491 Uhr UTC, unten mit Detailansicht der RMS-Kurve. Der Einbruch ist visuell in allen drei Phasen sichtbar (obere drei Kurven). ....	169
<b>Abbildung A.1:</b> Darstellung der Z-Normalisierung (von engl. zero mean and unit energy) anhand eines Beispiels. ....	A-6
<b>Abbildung A.2:</b> Ablauf unscharfer Ähnlichkeitssuchen in Zeitreihen. , in Anlehnung an [199]. ....	A-11
<b>Abbildung A.3:</b> a) Idealisierendes ungewöhnliches Muster (Discord) in periodischer Zeitreihe und b) wiederkehrende Muster (Motifs). Identisch zu Abbildung 2.5. Quelle: [79], modifiziert. ....	A-15
<b>Abbildung A.4:</b> Probleme bei der Distanzberechnung zweier Kurven - a) punktweise Differenz entspricht der Wahrnehmung, b) Versatz im Wertebereich, c) zusätzlicher Zeitversatz. ....	A-22
<b>Abbildung A.5:</b> Zeitreihenrepräsentationen. Quelle: [176], modifiziert. ....	A-28

<b>Abbildung A.6:</b>	Vergleich der Ableitung von Zeitreihenrepräsentationen. Quelle: [181]. Links oben: Originalzeitreihe, links unten: Vertikale und horizontale Quantisierung, rechts Resultate unterschiedlicher Approximationen. ....	A-30
<b>Abbildung A.7:</b>	Reduktion einer Zeitreihe mit 128 Datenpunkten zu einer PAA- Sequenz mit acht Mittelwerten als lineare Kombination einfacher Basisfunktionen. Quelle: [65].....	A-32
<b>Abbildung A.8:</b>	Aliasing bei Unterabtastung sinusähnlicher Zeitreihen.....	A-38
<b>Abbildung A.9:</b>	SpiralGraph Ansatz nach Weber et al., angewandt auf einen Datensatz zum Stromverbrauch. Quelle: [171].....	A-400
<b>Abbildung A.10:</b>	Bildschirmabzug der Software VizTree. Quelle: [171]. ....	A-411
<b>Abbildung A.11:</b>	Beispiel einer CGR der menschlichen Beta-Globin Region im elften Chromosom mit 73 357 Basen. Quelle: [167], modifiziert. ....	A-433
<b>Abbildung A.12:</b>	Darstellung der Funktionsweise von FCGRs der Länge eins (l. o.) bis acht (r. u.): Die Häufigkeiten für Sequenzen werden als Grauwerte in einer tabellarischen Darstellung codiert. Quelle: [66].....	A-433
<b>Abbildung A.13:</b>	Beispiele für Zeitreihenabschnitte und abgeleitete Intelligente Icons. Ein Werteeinbruch unterscheidet eine der Zeitreihen (siehe Pfeil) von allen anderen, was sich in der Icon-Darstellung wiederfindet. Quelle: [170].....	A-455
<b>Abbildung A.14:</b>	Entwicklung der weltweiten Kapazität für Datenspeicherung 1986-2007, gegliedert nach Medientyp. Quelle: [195], <a href="http://martinhibert.net">http://martinhibert.net</a> (12.04.2015), modifiziert.	A-48
<b>Abbildung A.15:</b>	Wachstum der Datenmenge weltweit, gegliedert nach inhaltlicher Struktur. Quelle: IDC [198].....	A-49
<b>Abbildung A.16:</b>	Wörterwolke, extrahiert aus 287 gesammelten englischsprachigen Publikationen zu Hadoop. Quelle: Eigene Darstellung <sup>38</sup> .....	A-566
<b>Abbildung A.17:</b>	Zeitstrahl zu Bucherscheinungen über Hadoop (Titel teilweise gekürzt, enthält eine angekündigte Erscheinung für 2016). Quelle: Eigene Darstellung, generiert mit Zotero <sup>38</sup> , modifiziert. ....	A-57
<b>Abbildung A.18:</b>	Darstellung des Ablaufs einer MapReduce Operation. Quelle: [14]. ....	A-600
<b>Abbildung B.1:</b>	Aufbau des deutschen Stromnetzes (Quelle: BMWi/dena, modifiziert). ....	B-4
<b>Abbildung B.2:</b>	Aufteilung der Versorgungsstörungen nach Vorfalltyp (Quelle: [94]).....	B-9
<b>Abbildung B.3:</b>	Typische kurze Spannungseinbrüche, oben: Sinus-Periode mit Transienten (rot markiert), unten: plötzlicher Spannungseinbruch über mehrere Perioden. ....	B-11
<b>Abbildung B.4:</b>	a) bis d): Ideale Sinusschwingungen (blau) mit harmonischen Schwingungen mehrfacher Grundfrequenz (orange), e): Überlagerungssignal (rot). Quelle: [3], modifiziert. ....	B-12
<b>Abbildung B.5:</b>	Kurvendarstellung von Strom im 3-Phasensystem: je zwei Perioden pro Phase mit 50 Hz (r, g, b) und sechs Perioden einer überlagerten Schwingung mit 150 Hz (schwarz), der dritten Harmonischen. ....	B-13

# Tabellenverzeichnis

<b>Tabelle 2.1:</b>	<i>Relevante Eigenschaften im Data-Mining gebräuchlicher Zeitreihenrepräsentationen, Angaben teilw. aus [258] (identisch zu Tabelle A.2).</i>	22
<b>Tabelle 2.2:</b>	<i>SAX-Distanztabelle für ein Alphabet mit drei Symbolen. Entspricht Tabelle A.3.</i>	226
<b>Tabelle 3.1:</b>	<i>Laufzeitvergleich von Pig (Hadoop) und zwei Mehrkernrechnern.</i>	38
<b>Tabelle 3.2:</b>	<i>Varianz der Laufzeit in der Hadoop-Umgebung mit Pig über fünf Testläufe mit bedingter Formatierung der Zellhintergründe.</i>	40
<b>Tabelle 3.3:</b>	<i>Genauigkeitsstufen der SAX-Sequenzen und SAX-Parameter.</i>	50
<b>Tabelle 3.4:</b>	<i>Identifizierte Zustandsübergänge bei der SAX-basierten Strukturanalyse</i>	92
<b>Tabelle 3.5:</b>	<i>Eigenschaften der Testdaten zur Detektion von Discords und Motifs.</i>	100
<b>Tabelle 3.6:</b>	<i>Detektierte Motifs in den sinusartigen Testdaten.</i>	103
<b>Tabelle 3.7:</b>	<i>Detektierte Discords in den sinusartigen Testdaten.</i>	104
<b>Tabelle 4.1:</b>	<i>Fallabhängige Komplexität des Quicksort-Algorithmus [287].</i>	115
<b>Tabelle 4.2:</b>	<i>Vergleich der Laufzeit der Discord/Motif-Detektion mit dem HOTSAX- und SortList-Ansatz.</i>	119
<b>Tabelle 4.3:</b>	<i>Gemessene datenabhängige Laufzeiten der Discord/Motif-Suche in den diversen Testdaten für den SortList- (blau) und HOTSAX-Ansatz (grün).</i>	122
<b>Tabelle 4.4:</b>	<i>Test mit lokalen Testdatensätzen zur Evaluation des Visualisierungsmoduls ViAT. Quelle: Eigene Messung.</i>	130
<b>Tabelle 5.1:</b>	<i>Merkmalsbezeichnungen in den EDR-Merkmalen des IAI und deren Bedeutung.</i>	142
<b>Tabelle 5.2:</b>	<i>Formeln zur Berechnung der sekundlichen EDR-Merkmale aus den hochfrequenten EDR-Messdaten (gemäß [192]).</i>	143
<b>Tabelle 5.3:</b>	<i>Aus EDR-Daten erzeugte Testdaten mit unterschiedlich ausgeprägten Transienten</i>	150
<b>Tabelle 5.4:</b>	<i>Merkmale mit Tagesmaximum zur Zeit des Spannungseinbruchs am 17.06.2013 im Vergleich zum Vortag, mit bedingter Formatierung der Zellhintergründe der Maximalwerte. Eigene Tabelle.</i>	161
<b>Tabelle A.1:</b>	<i>Fälle von Zeitreihen-Ähnlichkeitsanfragen</i>	A-9
<b>Tabelle A.2:</b>	<i>Relevante Eigenschaften im Data-Mining gebräuchlicher Zeitreihenrepräsentationen, Angaben teilw. aus [258] (identisch zu Tabelle 2.1).</i>	A-29
<b>Tabelle A.3:</b>	<i>Beispiele für Big Data und deren Wachstum in unterschiedlichen Bereichen (Quellen: [110], [137]).</i>	A-49
<b>Tabelle B.1:</b>	<i>Finanzielle Verluste durch Stromausfälle</i>	B-8
<b>Tabelle B.2:</b>	<i>Klassen ganzzahlig vielfacher Oberschwingungen</i>	B-13
<b>Tabelle B.3:</b>	<i>Symmetrische Phasenkomponenten von Oberschwingungen nach [4].</i>	B-14

# Code-Verzeichnis

<b>Code 3.1:</b>	<i>Java Funktionsbeispiel, um ein Pig-Skript auf dem Hadoop-Cluster auszuführen.....</i>	<i>37</i>
<b>Code 3.2:</b>	<i>Darstellung des Aufbaus eines „data“-Tags der EDR-Rohdaten anhand eines Beispiels (mit gekürztem Datenblock).....</i>	<i>37</i>
<b>Code 3.3:</b>	<i>Aufbau einer Pig-UDF in Java. ....</i>	<i>38</i>
<b>Code 3.4:</b>	<i>Java-Code des geschachtelten Aufrufs zur neuen SAX-Codierung inkl. RLE-Codierung bei der Metadatenerzeugung auf dem Hadoop-Cluster. ....</i>	<i>51</i>
<b>Code 3.5:</b>	<i>RLE-Codierung von zwei SAX-Beispielsequenzen.....</i>	<i>51</i>
<b>Code 3.6:</b>	<i>Resultierende XML-Metadaten aus einem Bereichskommentar in EDR-Merkmalsdaten. ....</i>	<i>68</i>
<b>Code 3.7:</b>	<i>Java-Code zur Berechnung der Proximitätsmatrix. ....</i>	<i>83</i>
<b>Code 3.8:</b>	<i>Detailliertes Resultat der Strukturanalyse als XML. ....</i>	<i>90</i>
<b>Code 3.9:</b>	<i>XML-Darstellung der Sequenz-Segmente als Resultat der Strukturanalyse.....</i>	<i>92</i>
<b>Code 3.10:</b>	<i>Java-Code zur Normalisierung der Auftrittshäufigkeit von Teilsequenzen. ....</i>	<i>96</i>
<b>Code 3.14:</b>	<i>Ausschnitt der Konsolenausgabe bei Durchführung einer Indizierung einer Metadatendatei für EDR-Merkmalsdaten. ....</i>	<i>107</i>
<b>Code 4.1:</b>	<i>Konsolenausgabe der Discord/Motif-Detektion in den Testdaten T1_lang, mit dem SortList-Ansatz (oben, blau) und dem HOTSAX-Ansatz (unten, grün), zusammengeführt und formatiert. ....</i>	<i>120</i>
<b>Code 5.1:</b>	<i>Beispiel für einen Daten-Block (rot hervorgehoben, gekürzt) einer EDR-Messwert-Datei.....</i>	<i>139</i>
<b>Code 5.2:</b>	<i>Beispiel für den Beginn einer EDR-Messwert-Datei im XML-Format mit Metainformationen zur Messung. ....</i>	<i>140</i>
<b>Code 5.3:</b>	<i>Beispiel für den Beginn einer EDR-Merkmals-Datei (Zeilen gekürzt).....</i>	<i>141</i>
<b>Code 5.4:</b>	<i>Zusammengefasste Konsolenausgaben der Discord-Detektionsresultate für Tr_1 bis Tr_4, mit den jeweiligen Discord-Sequenzen, Fundstellen und Pseudodistanzen (basierend auf der Position in der sortierten Liste). ....</i>	<i>154</i>

# Abkürzungsverzeichnis

KIT	Karlsruher Institut für Technologie
ANKA	Angströmquelle Karlsruhe
GPI	Geophysikalisches Institut
IAI	Institut für Angewandte Informatik
IAM	Institut für Angewandte Materialien
IMR	Institut für Mess- und Regelungstechnik
IMT	Institut für Mikrostrukturtechnik
IPE	Institut für Prozessdatenverarbeitung und Elektronik
SCC	Steinbuch Centre for Computing
TID	Abteilung Technische Infrastruktur und Dienste

## Allgemeine Abkürzungen

APCA	Adaptive Abschnittsweise Konstante Approximation (engl. Adaptive Piecewise Constant Approximation)
API	Application Programming Interface, englisch für Anwendungsprommierschnittstelle
BDEW	Bundesverband der Energie- und Wasserwirtschaft
CDH	Cloudera-Hadoop Distribution
CFPs	Closed Flexible Patterns
CHEB	Rekursive Tschebyscheff-Polynome
CSV	Comma-Separated Values oder auch Character-Separated Values.
DBMS	Datenbankmanagementsystem
DCT	Diskrete Cosinus Transformation
EDR	Energy Data Recorder
DFT	Diskrete Fourier Transformation

## Abkürzungsverzeichnis

DIC	Data-Intensive Computing
DTW	Dynamic Time Warping
DWT	Discrete Wavelet Transform
EDA	Explorative Datenanalyse
EMS	Energie-Management System
FAST	Fahrzeugsystemtechnik
FCGR	Frequency Chaos Game Representation
FraScaTi	Framework for Scalable Time Series Data Management and Analysis
GPA	Grid Protection Alliance
GridKa	Grid Computing Centre Karlsruhe
HDFS	Hadoop File System
HPC	High-Performance Computing
HWT	Haar-Wavelet-Transformation
KDD	Knowledge Discovery From Databases
KNN	K-Nächste-Nachbarn (engl. K-Nearest-Neighbors)
LCSS	Längsten Gemeinsamen Teilsequenz (Longest Common Sub-Sequence)
LHC	Large Hadron Collider
LSDF	Large-Scale Data Facility
MDL	Minimale Deskriptions-Länge
MIME	Multipurpose Internet Mail Extensions
NERC	North American Electric Reliability Corporation
NTP	Network Time Protocol, einem
openPDC	Open Source Phasor Data Concentrator
PAA	Abschnittsweise Aggregat-Approximation
PEM	Privacy-enhanced Electronic Mail
PLA	Abschnittsweise Lineare Approximation
PMU	Phasor Measurement Unit
POSIX	Portable Operating System Interface
PPS	Puls pro Sekunde

RLE	Lauf längencodierung
RMS	Root Mean Square
SAX	Symbolische Aggregat-Approximation
SCADA	Supervisory Control and Data Acquisition
SPDC	Super Phasor Concentrators
SSD	Solid-State-Disks
SVD	Singulärwertzerlegung
THD	Total Harmonic Distortion
TVA	Tennessee Valley Authority
UDF	User defined function
UTC	Coordinated Universal Time
ViAT	Visual Analysis of Time Series
VPN	LAN-zu-LAN VPN-Tunnel
WLCG	Worldwide LHC Computing Grid
WP	Verzerrungspfad
XML	Extensible Markup Language
ZR	Zeitreihe

# Literaturverzeichnis

- [1] AGRAWAL, R. ; FALOUTSOS, C. ; SWAMI, A. N.: Efficient Similarity Search In Sequence Databases. In: *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, FODO '93*. London, UK : Springer-Verlag, 1993. S. 69–84
- [2] AHMED, N. ; NATARAJAN, T. ; RAO, K. R.: Discrete Cosine Transform. In: *Selected Papers on Visual Communication: Technology and Applications* Bd. 13 (1990), S. 145 ff.
- [3] AIGNER, W. ; MIKSCH, S. ; SCHUMANN, H. ; TOMINSKI, C.: *Visualization of time-oriented data, Human-computer interaction series*. London : Springer, 2011.
- [4] AL AGHBARI, Z.: Effective image mining by representing color histograms as time series. In: *Journal of Advanced Computational Intelligence* Bd. 13 (2009), Nr. 2. S. 109 ff .
- [5] ALMOTAIRI, S. I. ; CLARK, A. J. ; DACIER, M. ; LEITA, C. ; MOHAY, G. M. ; PHAM, V. H.; THONNARD, O. ; ZIMMERMANN, J.: Extracting inter-arrival time based behaviour from honeypot traffic using cliques. In: *Proceedings 5th Australian Digital Forensics Conference (2007)*. S. 79-87, Perth, Western.
- [6] ANDROULAKIS, I. ; WU, J. ; VITOLO, J. ; ROTH, C.: Selecting maximally informative genes to enable temporal expression profiling analysis. In: *Proc. of Foundations of Systems Biology in Engineering (2005)*.
- [7] ANDROULAKIS, I. P.: New approaches for representing, analyzing and visualizing complex kinetic mechanisms. In: *Computer Aided Chemical Engineering* Bd. 20 (2005), S. 235–240
- [8] *Apache Lucene - Features*. URL <http://lucene.apache.org/core/features.html>. - abgerufen am 2015-06-11.
- [9] *Apache Lucene - Welcome to Apache Lucene*. URL <https://lucene.apache.org/>. - abgerufen am 2015-06-11.
- [10] *Apache Nutch<sup>TM</sup>* -. URL <http://nutch.apache.org/>. - abgerufen am 2015-03-19.
- [11] *Apache Solr*. URL <https://lucene.apache.org/solr/>. - abgerufen am 2015-06-11.
- [12] ARAKI, Y. ; ARITA, D. ; TANIGUCHI, R.-I.: Motion motif extraction from high-dimensional motion information. In: *Proc. of the 12th Korea-Japan Joint Workshop on Frontiers of Computer Vision, 2006*. S. 92–97
- [13] ARIZMENDI, C. M.: Fractal Analysis for Social Systems. In: *arXiv preprint adap-org/9910001 (1999)*.
- [14] ARMSTRONG, T. ; OATES, T.: Riptide: Segmenting data using multiple resolutions. In: *Development and Learning, 2007. ICDL 2007*. S. 306–311
- [15] ARMSTRONG, T. ; OATES, T.: Undertow: Multi-level segmentation of real-valued time series. In: *Proc. of the national conference on artificial intelligence*. Bd. 22 : Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007. S. 1842 ff.



- [16] AUSSCHUSS FÜR BILDUNG, FORSCHUNG UND TECHNIKFOLGENABSCHÄTZUNG DES DEUTSCHEN BUNDESTAGES: *TA-Projekt: Gefährdung und Verletzbarkeit moderner Gesellschaften – am Beispiel eines großräumigen und langandauernden Ausfalls der Stromversorgung* (Bericht über Technikfolgenabschätzung). TAB-Arbeitsbericht Nr. 141, Büro für Technikfolgen-Abschätzung beim Deutschen Bundestag; 2010.
- [17] AXELBERG, P. G.V. ; BOLLEN, M. H. J.: Trace of Flicker Sources by Using the Quantity of Flicker Power. In: *Power Delivery, IEEE Transactions on* (2008), Nr. 1, S. 465–471.
- [18] BACH, F. ; ÇAKMAK, H. ; MAAB, H. ; KÜHNAPFEL, U.: Power Grid Time Series Data Analysis with Pig on a Hadoop Cluster compared to Multi Core Systems. In: *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013*. Belfast, Ireland : IEEE Computer Society, 2013. S. 208–212
- [19] BACKUS, J.: Can Programming be Liberated from the Von Neumann Style?: A Functional Style and Its Algebra of Programs. In: *Commun. ACM* Bd. 21 (1978), Nr. 8, S. 613–641
- [20] BAEZA-YATES, R.: *Modern information retrieval*. New York : Harlow, England : ACM Press ; Addison-Wesley, 1999.
- [21] BAGGINI, A.: *Handbook of Power Quality*. 1. Auflage. Chichester, England ; Hoboken, NJ : John Wiley & Sons, 2008.
- [22] BAKALOV, P. ; HADJIELEFTHERIOU, M. ; TSOTRAS, V. J.: Time Relaxed Spatiotemporal Trajectory Joins. In: *Proceedings of the 13th Annual CM International Workshop on Geographic Information Systems, GIS '05*. New York, USA : ACM, 2005. S. 182–191
- [23] BALAGULA, Y. ; SAKULIN, M. ; KOROVKIN, N.: Fractal theory and voltage flicker evaluation. In: *Power Tech, 2005 IEEE Russia, 2005*, S. 1–6
- [24] BARBOSA, F. ; RODRIGUES, A.: Range Queries over Trajectory Data with Recursive List of Clusters: A case study with Hurricanes data. In: *Proc. of Geographic Information Science Research UK (GISRUK 2009)* (2009).
- [25] BEDERSON, B. B.: Fisheye menus. In: *Proceedings of the 13th annual ACM symposium on User interface software and technology* : ACM, 2000. S. 217–225
- [26] BELLMAN, R.: The theory of dynamic programming. In: *Bulletin of the American Mathematical Society* Bd. 60 (1954), Nr. 6, S. 503–515
- [27] BELLMAN, R.: *Dynamic programming, Princeton landmarks in mathematics*. Princeton, NJ : Princeton University Press, 2010.
- [28] BELLMAN, R. ; KALABA, R.: On adaptive control processes. In: *IRE Transactions on Automatic Control* Bd. 4 (1959), Nr. 2, S. 1–9
- [29] BENGFORT, B. ; KIM, J.: *Data Analytics with Hadoop* : Oreilly & Assoc Inc, 2015.
- [30] BERGMANN, L.; SCHÄFER, C. : *Mechanik – Akustik – Wärmelehre* : Walter de Gruyter, Berlin, 1945.
- [31] *Bericht zum Zustand der leitungsgebundenen Energieversorgung im Winter 2012/13* (Zustandsbericht Energieversorgung) : Bundesnetzagentur, 2013.

- [32] BLUMBERG, R. ; ATRE, S.: The problem with unstructured data. In: *DM REVIEW* Bd. 13 (2003), S. 42–49
- [33] BOLLEN, M. H. J. ; GU, I. Y. H.: Power Engineering Letters. In: *IEEE Transactions on Power Delivery* Bd. 20 (2005), Nr. 2, S. 1199 ff.
- [34] LE, C. D. ; BOLLEN, M. H. J.: Analysis of power disturbances from monitoring multiple levels and locations in a power system. In: *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*, 2012, S. 1–7
- [35] BOLLEN, M. H. J. ; HÄGER, M.: Power quality: interactions between distributed energy resources, the grid, and other customers. In: *Electric Power Quality and Utilisation Magazine* Bd. 1 (2005), Nr. 1, S. 51–61
- [36] BOLLEN, M. H. J. ; RIBEIRO, P. ; GU, I. YH ; DUQUE, C. A.: Trends, challenges and opportunities in power quality research. In: *European Transactions on Electrical Power* Bd. 20 (2010), Nr. 1, S. 3–18
- [37] BOLLT, E. M. ; SKUFCA, J. D. ; MCGREGOR, S. J.: Control entropy: A complexity measure for nonstationary signals. In: *Mathematical Biosciences and Engineering* Bd. 6 (2009), Nr. 1, S. 1–25
- [38] BORTHAKUR, D.: The hadoop distributed file system: Architecture and design. In: *Hadoop Project Website* Bd. 11 (2007), S. 21 ff.
- [39] BÖSCH, F.: *Mediengeschichte: vom asiatischen Buchdruck zum Fernsehen* : Campus Verlag, 2011.
- [40] BREKKE, K. ; ESTEVES, J. ; BOLLEN, M. ; HABER, A. ; WESTERGAARD, T. ; KOLESSAR, R.: Monitoring of and regulations on quality of electricity supply in European countries. In: *Proceedings CIRED International Conference on Electricity Distribution, Praque, 2009*. S. 1-4
- [41] BRIGOLA, R.: *Fourier-Analysis und Distributionen: Eine Einführung mit Anwendungen*. Hamburg : edition swk, 2013.
- [42] *Bundesnetzagentur - Pressemitteilung - Zuverlässigkeit der Stromversorgung auf konstant hohem Niveau* (Pressebericht). Bonn, Germany : Bundesnetzagentur, 2014.
- [43] *Bundesregierung | Glossar zu Energie*. URL <http://www.bundesregierung.de/Content/DE/StatischeSeiten/Breg/FAQ/faq-energie.html>. - abgerufen am 2014-11-28.
- [44] BUNDESREPUBLIK DEUTSCHLAND: Fortschrittsbericht nach Artikel 22 der Richtlinie 2009/28/EG zur Förderung der Nutzung von Energie aus erneuerbaren Quellen (2013).
- [45] BURLINGAME, N.: *The Little Book of BIG DATA, 2012 Edition* : New Street Communications, LLC, 2012.
- [46] CAI, Y. ; NG, R.: Indexing spatio-temporal trajectories with Chebyshev polynomials. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* : ACM, 2004, S. 599–610
- [47] ÇAKMAK, H. ; MAAß, H. ; BACH, F. ; KÜHNAPFEL, U.: A New Framework for the Analysis of Large Scale Multi-Rate Power Data (2014). In: *KIT Scientific Working Papers 21*: KIT, Karlsruhe, 2014. S. 1-42

- [48] CAMERRA, A. ; PALPANAS, T. ; SHIEH, J. ; KEOGH, E.: iSAX 2.0: Indexing and mining one billion time series. In: *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 2010. S. 58–67
- [49] CANDAN, K. S. ; ROSSINI, R. ; WANG, X. ; SAPINO, M. L.: sDTW: computing DTW distances using locally relevant constraints based on salient feature alignments. In: *Proceedings of the VLDB Endowment* Bd. 5 (2012), Nr. 11, S. 1519–1530
- [50] CARLIS, J. V. ; KONSTAN, J. A.: Interactive visualization of serial periodic data. In: *Proceedings of the 11th annual ACM symposium on User interface software and technology*, 1998, S. 29–38
- [51] CASSISI, C. ; MONTALTO, P. ; ALIOTTA, M. ; CANNATA, A. ; PULVIRENTI, A.: Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining. In: KARAHOCA, A. (Hrsg.): *Advances in Data Mining Knowledge Discovery and Applications* : InTech, 2012.
- [52] CASSON, L.: *Libraries in the Ancient World* : Yale University Press, 2002.
- [53] CASTRO, N. ; AZEVEDO, P.: Multiresolution Motif Discovery in Time Series. In: *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010.
- [54] CELLY, B. ; ZORDAN, V.: Animated people textures. In: *Proc. of 17th International Conference on Computer Animation and Social Agents (CASA)*, Genf, 2004.
- [55] CHAKRABARTI, K. ; KEOGH, E. ; MEHROTRA, S. ; PAZZANI, M.: Locally adaptive dimensionality reduction for indexing large time series databases. In: *ACM Transactions on Database Systems (TODS)* Bd. 27 (2002), Nr. 2, S. 188–228
- [56] CHAN, K.-P. ; FU, A.W.-C.: Efficient time series matching by wavelets. In: *15th International Conference on Data Engineering, 1999. Proceedings*, 1999, S. 126–133
- [57] CHANDOLA, V.: *Anomaly detection for symbolic sequences and time series data*, University of Minnesota, 2009.
- [58] CHEN, J. R.: Making subsequence time series clustering meaningful. In: *Data Mining, Fifth IEEE International Conference on* : IEEE, 2005, S. 8 ff.
- [59] CHEN, J. ; MOON, Y.-S.: Using SIFT features in palmprint authentication. In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on* : IEEE, 2008, S. 1–4
- [60] CHEN, J.-S. ; MOON, Y.-S. ; YEUNG, H.-W.: Palmprint Authentication Using Time Series. In: KANADE, T. ; JAIN, A. ; RATHA, N. K. (Hrsg.): *Audio- and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science* : Springer Berlin Heidelberg, 2005. S. 376–385
- [61] CHEN, L. ; OZSU, M. T. ; ORIA, V.: Using Multi-Scale Histograms to Answer Pattern Existence and Shape Match Queries over Time Series Data. In: *Proc. of 17th Int'l Conf. on Scientific and Statistical Database Management, Santa Barbara, CA, USA*, 2005. S. 217–226
- [62] CORMEN, T. ; BALKCOM, D.: *Analysis of quicksort*. URL <http://www.khanacademy.org>. - abgerufen am 2017-02-24. Khan Academy.
- [63] CZARNECKI, L. S.: Currents Physical Components (CPC) concept: A fundamental of power theory. In: *International School on Nonsinusoidal Currents and Compensation, 2008. ISNCC 2008*, 2008, S. 1–11

- [64] DAS, G. ; LIN, K.-I. ; MANNILA, H. ; RENGANATHAN, G. ; SMYTH, P.: Rule Discovery from Time Series. In: *KDD*. Bd. 98, 1998. S. 16–22
- [65] DAVIS, K. ; PATTERSON, D.: *Ethics of Big Data: Balancing Risk and Innovation*. 1. Aufl. : O'Reilly Media, 2012.
- [66] DEAN, J. ; GHEMAWAT, S.: MapReduce: simplified data processing on large clusters. In: *OSDI'04: Proceedings of the 6th conference on symposium on operating systems design and implementation* : USENIX Association, 2004. S. 10-10
- [67] DEAN, J. ; GHEMAWAT, SANJAY: MapReduce: simplified data processing on large clusters. In: *Communications of the ACM* Bd. 51 (2008), S. 107–113
- [68] DEROOS, D.: *Hadoop For Dummies*. 1. Aufl. : For Dummies, 2014.
- [69] DESCHAVANNE, P. J. ; GIRON, A. ; VILAIN, J. ; FAGOT, G. ; FERTIL, B.: Genomic signature: characterization and classification of species assessed by chaos game representation of sequences. In: *Molecular biology and evolution* Bd. 16 (1999), Nr. 10, S. 1391–1399
- [70] DFG, AUSSCHUSS FÜR WISSENSCHAFTLICHE BIBLIOTHEKEN UND INFORMATIONSSYSTEME - UNTERAUSSCHUSS FÜR INFORMATIONSMANAGEMENT: Empfehlungen zur gesicherten Aufbewahrung und Bereitstellung digitaler Forschungsprimärdaten: Deutsche Forschungsgemeinschaft, Bonn, 2009.
- [71] DHAR, V.: Data science and prediction. In: *Communications of the ACM* Bd. 56 (2013), Nr. 12, S. 64–73
- [72] DING, H. ; TRAJCEVSKI, G. ; SCHEUERMANN, P. ; WANG, X. ; KEOGH, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. In: *Proceedings of the VLDB Endowment* Bd. 1 (2008), Nr. 2, S. 1542–1552
- [73] DONG, B. ; ZHENG, Q. ; TIAN, F. ; CHAO, K.-M. ; GODWIN, N. ; MA, T. ; XU, H.: Performance models and dynamic characteristics analysis for HDFS write and read operations: A systematic view. In: *Journal of Systems and Software* Bd. 93 (2014), S. 132-151
- [74] DUCHENE, F. ; GARBAY, C. ; RIALLE, V.: Mining heterogeneous multivariate time-series for learning meaningful patterns: application to home health telecare. In: *arXiv preprint cs/0412003* (2004).
- [75] DUCKETT, G.: *Hadoop: Questions and Answers* : George Duckett, 2015.
- [76] DUMBILL, E.: *Planning for Big Data* : O'Reilly Media, 2012
- [77] *Energiewende im Strommarkt. Chancen nutzen – Risiken vermeiden*: Bayerischer Industrie- und Handelskammertag, München, 2014.
- [78] ERGOVIC, V. ; TONKOVIC, S. ; MEDVED, V.: Human gait data mining by symbol based descriptive features. In: *World Congress on Medical Physics and Biomedical Engineering, September 7-12, 2009, Munich, Germany* : Springer, 2009. S. 460–463
- [79] ESLING, P. ; AGON, C.: Time-series data mining. In: *ACM Computing Surveys* Bd. 45 (2012), Nr. 1, S. 1–34.
- [80] FABRI, M. ; MASCIOLI, G. ; PALONARA, G. ; PERDON, A. M. ; VIOL, S. R.: Activation and delay in fMRI brain signals of selective attention. In: *Proceedings of Int. IJCNN07 Workshop on Neurodynamics*. Orlando, Florida, USA, 2007.

- [81] FALOUTSOS, C. ; RANGANATHAN, M. ; MANOLOPOULOS, Y.: Fast Subsequence Matching in Time-series Databases. In: *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, SIGMOD '94*. New York, NY, USA : ACM, 1994. S. 419–429
- [82] FIELD, M. ; STIRLING, D. A. ; NAGHDY, F. ; PAN, Z.: Motion segmentation for humanoid control planning (2008). In: *Australasian Conference on Robotics and Automation, Australian Robotics & Automation Association, Canberra, Australia*. S 6 ff.
- [83] FRAMPTON, M.: *Big Data Made Easy: A Working Guide to the Complete Hadoop Toolset*. Berkeley, CA : Apress, 2014.
- [84] FRANKS, B.: *Taming The Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics*. 1. Aufl. : Wiley, 2012
- [85] FRIENDLY, M. ; DENIS, D.: The early origins and development of the scatterplot. In: *Journal of the History of the Behavioral Sciences* Bd. 41 (2005), Nr. 2, S. 103–130.
- [86] FU, T.-C.: A review on time series data mining. In: *Engineering Applications of Artificial Intelligence* Bd. 24 (2011), Nr. 1, S. 164–181.
- [87] GANTZ, J. ; REINSEL, D.: The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. In: *IDC iView: IDC Analyze the Future* (2012). S.1 ff.
- [88] GARCIA, A. O. ; BOUROV, S. ; HAMMAD, A. ; HARTMANN, V. ; JEJKAL, T. ; OTTE, J. C. ; PFEIFFER, S. ; SCHENKER, T. ; SCHMIDT, C. ; U. A.: Data-intensive analysis for scientific experiments at the Large Scale Data Facility. In: *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, Alaska, USA, 2011. S. 125–126
- [89] GARCIA, A. O. ; BOUROV, S. ; HAMMAD, A. ; VAN WEZEL, J. ; NEUMAIR, B. ; STREIT, A. ; HARTMANN, V. ; JEJKAL, T. ; NEUBERGER, P. ; U. A.: The Large Scale Data Facility: Data Intensive Computing for Scientific Experiments. In: *Proc. of 2011 IEEE International Parallel & Distributed Processing Symposium* : IEEE, 2011. S. 1467–1474
- [90] GATES, A.: *Programming Pig* : O'Reilly Media, 2011.
- [91] GHEMAWAT, S. ; GOBIOFF, H. ; LEUNG, S.-T.: The Google file system. In: *ACM SIGOPS operating systems review*. Bd. 37 : ACM, 2003. S. 29–43
- [92] GOLDIN, D. Q. ; KANELAKIS, P. C.: On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. In: MONTANARI, U. ; ROSSI, F. (Hrsg.): *Principles and Practice of Constraint Programming — CP '95, Lecture Notes in Computer Science* : Springer Berlin Heidelberg, 1995. S. 137–153
- [93] GOLDMAN, N: Nucleotide, dinucleotide and trinucleotide frequencies explain patterns observed in chaos game representations of DNA sequences. In: *Nucleic Acids Research* Bd. 21 (1993), Nr. 10, S. 2487–2491.
- [94] GOLDRATT, E. M.: *What is this thing called theory of constraints and how should it be implemented?* Croton-on-Hudson, N.Y : North River Press, 1990.
- [95] GOSPODNETIĆ, O. ; HATCHER, E.: *Lucene in action*. Greenwich, CT : Manning Publications, New York, USA, 2005.

- [96] GREVELER, U. ; JUSTUS, B. ; LOEHR, D.: Forensic content detection through power consumption. In: *Communications (ICC), 2012 IEEE International Conference on* : IEEE, 2012. S. 6759–6763
- [97] *GridKa - Grid Computing Centre Karlsruhe*. URL <http://www.gridka.de/cgi-bin/frame.pl?seite=/welcome.html>. - abgerufen am 2015-03-30.
- [98] GROVER, M. ; MALASKA, T. ; SEIDMAN, J.: *Hadoop Application Architectures*. 1. Aufl. : O'Reilly Vlg. Gmbh & Co., 2015.
- [99] GU, Y. H. ; BOLLEN, M. H. J.: Time-frequency and time-scale domain analysis of voltage disturbances. In: *Power Delivery, IEEE Transactions on* Bd. 15 (2000), Nr. 4, S. 1279-1284.
- [100] GUNARATHNE, T.: *Hadoop MapReduce v2 Cookbook - Second Edition* : Packt Publishing, Birmingham, UK, 2015.
- [101] HADAMARD, J.: *The early scientific work of Henri Poincaré* : Houston, 1922.
- [102] HAEMER, K. W.: Range-Bar Charts. In: *The American Statistician* Bd. 2 (1948), Nr. 2, S. 23–23
- [103] HAN, J. ; KAMBER, M.: *Data mining concepts and techniques*. Amsterdam; Boston; San Francisco, CA : Elsevier ; Morgan Kaufmann, 2006.
- [104] HARRIS, D.: *The history of Hadoop: From 4 nodes to the future of data*. URL <https://gigaom.com/2013/03/04/the-history-of-hadoop-from-4-nodes-to-the-future-of-data/>. - abgerufen am 2015-03-19.
- [105] HARTER, T. ; BORTHAKUR, D. ; DONG, S. ; AIYER, A. ; TANG, L. ; ARPACI-DUSSEAU, A. C. ; ARPACI-DUSSEAU, R. H.: Analysis of HDFS Under HBase: A Facebook Messages Case Study. In: *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST '14)*, 2014.
- [106] HE, W. ; HU, G. ; YAO, X. ; KAN, G. ; WANG, H. ; XIANG, H.: Applying multiple time series data mining to large-scale network traffic analysis. In: *2008 IEEE Conference on Cybernetics and Intelligent Systems*, 2008. S. 394–399
- [107] HE, W. ; XIANG, H. ; TANG, J.: Analog-Circuit Fault Diagnosis Using Three-Stage Preprocessing and Time Series Data Mining. In: *2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis*, 2009. S. 1–4
- [108] HEER, J. ; CARD, S. K. ; LANDAY, J. A.: prefuse: a toolkit for interactive information visualization. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05*. New York, NY, USA : ACM, 2005 S. 421–430
- [109] HEY, T. ; TANSLEY, S. ; TOLLE, K.: *The Fourth Paradigm: Data-intensive Scientific Discovery* : External Research, Microsoft Research, Redmond, USA, 2009.
- [110] HIGHLAND, F. ; STEPHENSON, J.: Fitting the Problem to the Paradigm: Algorithm Characteristics Required for Effective Use of MapReduce. In: *Procedia Computer Science, Complex Adaptive Systems 2012*. Bd. 12 (2012), S. 212–217
- [111] HILBERT, M. ; LÓPEZ, P.: The world's technological capacity to store, communicate, and compute information. In: *science* Bd. 332 (2011), Nr. 6025, S. 60–65
- [112] HIRSCHBERG, D. S.: A Linear Space Algorithm for Computing Maximal Common Subsequences. In: *Commun. ACM* Bd. 18 (1975), Nr. 6, S. 341–343

- [113] HIRSHFIELD, L. M. ; CHAUNCEY, K. ; GULOTTA, R. ; GIROUARD, A. ; SOLOVEY, E. T. ; JACOB, R. J. K. ; SASSAROLI, A. ; FANTINI, S.: Combining electroencephalograph and functional near infrared spectroscopy to explore users' mental workload. In: *Foundations of Augmented Cognition. Neuroergonomics and Operational Neuroscience* : Springer, 2009, S. 239–247
- [114] HITZLER, P. ; JANOWICZ, K.: Linked Data, Big Data, and the 4th Paradigm. In: *Semantic Web* Bd. 4 (2013), Nr. 3, S. 233–235
- [115] HOARE, C. A. R.: Quicksort. In: *The Computer Journal* Bd. 5 (1962), Nr. 1, S. 10–16
- [116] HÖCK, G.: „Dirty Power“ Oberschwingungen durch nichtlineare Verbraucher. URL [www.gmc-instruments.ch/src/download/dDirty\\_Power.pdf](http://www.gmc-instruments.ch/src/download/dDirty_Power.pdf) (2009). abgerufen am 2013-09-23
- [117] HOLMES, A.: *Hadoop in Practice*. 1. Aufl. : Manning Publications, Shelter Island, N.Y., 2012.
- [118] HOLMES, A.: *Hadoop in Practice*. 2. Aufl. : Manning Publications, Shelter Island, N.Y., 2015.
- [119] Home | BDEW | Bundesverband der Energie- und Wasserwirtschaft. URL <https://www.bdew.de/>. - abgerufen am 2015-02-17.
- [120] HUANG, Y.-W. ; YU, P. S.: Adaptive Query Processing for Time-series Data. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '99*. New York, NY, USA : ACM, 1999. S. 282–286
- [121] HUNG, N. Q. V. ; ANH, D. T.: Combining SAX and Piecewise Linear Approximation to improve similarity search on financial time series. In: *Information Technology Convergence, 2007. ISITC 2007. International Symposium on* : IEEE, 2007. S. 58–62
- [122] HUNTER, J. ; COLLEY, M.: Feature extraction from sensor data streams for real-time human behaviour recognition. In: *Knowledge Discovery in Databases: PKDD 2007* : Springer, 2007. S. 115–126
- [123] HUTCHISON, D. ; KANADE, T. ; KITTLER, J. ; KLEINBERG, J. M. ; MATTERN, F. ; MITCHELL, J. C. ; NAOR, M. ; NIERSTRASZ, O. ; PANDU R., C. ; U. A.: Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables. In: SETCHI, R. ; JORDANOV, I. ; HOWLETT, R. J. ; JAIN, L. C. (Hrsg.): *Knowledge-Based and Intelligent Information and Engineering Systems*. Bd. 6276. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. S. 420–431
- [124] IEEE Standard Definitions for the Measurement of Electric Power Quantities Under Sinusoidal, Nonsinusoidal, Balanced, or Unbalanced Conditions - Redline. In: *IEEE Std 1459-2010 (Revision of IEEE Std 1459-2000)* (2010), S. 1–50
- [125] iPDC - Free Phasor Data Concentrator. URL <http://ipdc.codeplex.com/>. - abgerufen am 2012-08-15
- [126] ISLAM, M. K. ; SRINIVASAN, A.: *Apache Oozie: The Workflow Scheduler for Hadoop*. 1. Aufl. : O'Reilly Vlg. Gmbh & Co., 2015.

- [127] ISLAM, N. S. ; RAHMAN, M. W. ; JOSE, J. ; RAJACHANDRASEKAR, R. ; WANG, H. ; SUBRAMONI, H. ; MURTHY, C. ; PANDA, D. K.: High Performance RDMA-based Design of HDFS over InfiniBand. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12*. Los Alamitos, CA, USA : IEEE Computer Society Press, 2012. S. 1-35
- [128] JAHN, J.: *Energiekonditionierung in Niederspannungsnetzen unter besonderer Berücksichtigung der Integration verteilter Energieerzeuger in schwachen Netzausläufern* : Kassel University Press GmbH, 2008.
- [129] JAIN, V.: Indexing and Mining a Billion Time series using iSAX 2.0: USC Viterbi, University of Southern California, 2012
- [130] JEFFREY, H J: Chaos game representation of gene structure. In: *Nucleic Acids Research* Bd. 18 (1990), Nr. 8, S. 2163–2170.
- [131] *JFreeChart*. URL <http://www.jfree.org/jfreechart/>. - abgerufen am 2015-05-13.
- [132] JIANG, L. ; LI, B. ; SONG, M.: The optimization of HDFS based on small files. In: *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*: IEEE, 2010. S. 912–915
- [133] JIANG, L. ; LI, B. ; SONG, M.: The optimization of HDFS based on small files. In: *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, 2010. S. 912–915
- [134] JOHNSTON, W.E.: High-speed, wide area, data intensive computing: a ten year retrospective. In: *The Seventh International Symposium on High Performance Distributed Computing, 1998. Proceedings*, 1998. S. 280–291
- [135] JURNEY, R.: *Agile Data Science: Building Data Analytics Applications with Hadoop*. 1. Aufl. : O'Reilly Media, 2013.
- [136] KAISLER, S. ; ARMOUR, F. ; ESPINOSA, J.A. ; MONEY, W.: Big Data: Issues and Challenges Moving Forward. In: *2013 46th Hawaii International Conference on System Sciences (HICSS)*, 2013. S. 995–1004
- [137] KALA KARUN, A. ; CHITHARANJAN, K.: A review on hadoop - HDFS infrastructure extensions. In: *2013 IEEE Conference on Information Communication Technologies (ICT)*, 2013. S. 132–137
- [138] KAMENKA, A.: Sechs Themen rund um Oberschwingungen und die Netzqualität in Stromversorgungsnetzen. In: *IHKS-Fachjournal*, 2013. Online: <https://www.ihks-fachjournal.de/sechs-themen-um-oberschwingungen-und-netzqualitaet-in-stromversorgungsnetzen/>. - abgerufen am 2014-02-21.
- [139] KARAMITOPOULOS, L. ; EVANGELIDIS, G.: Current Trends in Time Series Representation. In: *Proc. Panhellenic Conference in Informatics (PCI)*, Patras, Greece, 2007. S. 217–226
- [140] KARANTH, S.: *Mastering Hadoop*. Place of publication not identified : Packt Publishing, 2014.
- [141] KASTEN, E. P. ; MCKINLEY, P. K. ; GAGE, S. H.: Automated ensemble extraction and analysis of acoustic data streams. In: *Distributed Computing Systems Workshops, 2007. ICDCSW'07. 27th International Conference on* : IEEE, 2007. S. 66 ff.



- [142] KAUSHIK, R. T ; BHANDARKAR, M. ; NAHRSTEDT, K.: Evaluation and Analysis of GreenHDFS: A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System. In: *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)* : IEEE, 2010. S. 274–287
- [143] KEOGH, E. ; LIN, J. ; FU, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: *Data Mining, Fifth IEEE International Conference on*, 2005, S. 8 ff.
- [144] KEOGH, E. ; WEI, LI ; XI, X. ; LONARDI, S. ; SHIEH, J. ; SIROWY, S.: Intelligent Icons: Integrating Lite-Weight but Ubiquitous Data Mining into GUI Operating Systems. In: *Journal of Universal Computer Science* Bd. 11 (2005), Nr. 11, S. 1820–1834
- [145] KEOGH, E. ; CHAKRABARTI, K. ; PAZZANI, M. ; MEHROTRA, S.: Locally adaptive dimensionality reduction for indexing large time series databases. In: *ACM SIGMOD Record*. Bd. 30, 2001, S. 151–162
- [146] KEOGH, E. ; CHAKRABARTI, K. ; PAZZANI, M. ; MEHROTRA, S.: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. In: *Knowledge and Information Systems* Bd. 3 (2001), Nr. 3, S. 263–286
- [147] KEOGH, E. ; CHU, S. ; HART, D. ; PAZZANI, M.: Segmenting Time Series: A Survey and Novel Approach. In: *In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific : Publishing Company*, 1993. S. 1–22
- [148] KEOGH, E. ; CHU, S. ; HART, D. ; PAZZANI, M.: An online algorithm for segmenting time series. In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on* : IEEE, 2001, S. 289–296
- [149] KEOGH, E. ; HOCHHEISER, H. ; SHNEIDERMAN, B.: An Augmented Visual Query Mechanism for Finding Patterns in Time Series Data. In: CARBONELL, J. G. ; SIEKMANN, J. ; ANDREASEN, T. ; CHRISTIANSEN, H. ; MOTRO, A. ; LARSEN, H. L. (Hrsg.): *Flexible Query Answering Systems, Lecture Notes in Computer Science* : Springer Berlin Heidelberg, 2002. S. 240–250
- [150] KEOGH, E. J. ; PAZZANI, M. J.: Relevance feedback retrieval of time series data. In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* : ACM, 1999. S. 183–190
- [151] KEOGH, E. J. ; PAZZANI, M. J.: A simple dimensionality reduction technique for fast similarity search in large time series databases. In: *Knowledge Discovery and Data Mining. Current Issues and New Applications* : Springer, 2000, S. 122–133
- [152] KEOGH, E. ; KASETTY, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. In: *Data Mining and knowledge discovery* Bd. 7 (2003), Nr. 4, S. 349–371
- [153] KEOGH, E.; LIN, J.: Clustering of time-series subsequences is meaningless: implications for previous and future research. In: *Knowledge and Information Systems* Bd. 8 (2005), Nr. 2, S. 154–177
- [154] KEOGH, E. ; MUEEN, A.: Curse of Dimensionality. In: SAMMUT, C. ; WEBB, G. I. (Hrsg.): *Encyclopedia of Machine Learning* : Springer US, 2011. S. 257–258
- [155] KHAN, M. ; ASHTON, P. M. ; LI, M. ; TAYLOR, G. ; PISICA, I. ; LIU, J. ; U.A.: Parallel detrended fluctuation analysis for fast event detection on massive pmu data. In: *Smart Grid, IEEE Transactions on* Bd. 6 (2015), Nr. 1, S. 360–368.

- [156] KHARE, R. ; CUTTING, D.: *Nutch: A flexible and scalable open-source web search engine*, 2004.
- [157] KIEFER, G. ; SCHMOLKE, H.: *VDE 0100 und die Praxis Wegweiser für Anfänger und Profis*. Berlin : VDE VERLAG, 2014.
- [158] KILTER, J. ; MEYER, J. ; HOWE, B. ; ZAVODA, F. ; TENTI, L. ; MILANOVIC, J. V. ; BOLLEN, M. ; RIBEIRO, P. F. ; DOYLE, P. ; U. A.: Current practice and future challenges for power quality monitoring-CIGRE WG C4. 112 perspective. In: *Harmonics and Quality of Power (ICHQP), 2012 IEEE 15th International Conference on* : IEEE, Hong Kong, China, 2012. S. 390–397
- [159] KINJO, K. ; OZAKI, T. ; SAWAI, K. ; FURUKAWA, K.: Knowledge acquisition from time series data by association rule and network analysis. In: *Proceedings of the 19th Annual Conference of the Japanese Society for Artificial Intelligence*, 2005.
- [160] *KIT Energy Center Startseite*. URL <http://www.energy.kit.edu/>. - abgerufen am 2014-09-19.
- [161] KITAGUCHI, S.: Extracting feature based on motif from a chronic hepatitis dataset. In: *Proceedings of 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI'04)*, 2004.
- [162] KORN, F. ; JAGADISH, H. V. ; FALOUTSOS, C.: Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. In: *In SIGMOD*, 1997, S. 289–300
- [163] KRAUS, J. ; BUBLA, V. ; KUKAČKA, L.: data modeling for reduction of volume in large archives of power quality data. In: *21st International Conference on Electricity Distribution (CIRED)*, 2011.
- [164] KREIB, J.-P.: *Einführung in die Zeitreihenanalyse*. 2006. Aufl. Berlin : Springer, 2006.
- [165] KUKACKA, L. ; KRAUS, J. ; BUBLA, V. ; STEPAN, P.: CPC and IEEE power theory-application for offline waveform data analysis (2013): IET Digital Library, 2013.
- [166] KULAHCIOGLU, B. ; OZDEMIR, S. ; KUMOVA, B.: Application of symbolic Piecewise Aggregate Approximation (PAA) analysis to ECG signals. In: *Proceedings of the 17th IASTED International Conference*. Bd. 609, 2008. S. 153 ff.
- [167] KUNDUR, P.: *Power System Stability and Control*. 1ST edition. Aufl. New York : McGraw-Hill Professional, 1994.
- [168] LAM, C.: *Hadoop in Action* : Manning, 2010.
- [169] LAMPING, J. ; RAO, R. ; PIROLI, P.: A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995. S. 401–408
- [170] LAST, M. ; KANDEL, A. ; BUNKE, H.: *Data Mining in Time Series Databases*. New Jersey ; London : World Scientific Pub Co, 2004.
- [171] lbehnke/hierarchical-clustering-java · GitHub URL <https://github.com/lbehnke/hierarchical-clustering-java/>. abgerufen am 2014-06-06.
- [172] LE, M. H. J. BOLLEN C. D.: Analysis of power disturbances from monitoring multiple levels and locations in a power system (2010), S. 1–7
- [173] LEVENSHTIN, V.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In: *Soviet Physics Doklady* Bd. 10 (1966), S. 707

- [174] LI, Y. ; LIN, J.: Approximate variable-length time series motif discovery using grammar inference. In: *Proceedings of the Tenth International Workshop on Multimedia Data Mining* : ACM, 2010. S. 10 ff.
- [175] LICATA, A. ; PSARROU A.: Tokenization for Gesture Space Modelling. In: *13TH INTERNATIONAL CONFERENCE ON APPLICATIONS OF NATURAL LANGUAGE TO INFORMATION SYSTEMS, DOCTORAL SYMPOSIUM (NLDB'08-DS)*, 2008.
- [176] LIN, J. ; KEOGH, E. ; LONARDI, S. ; CHIU, B.: A symbolic representation of time series, with implications for streaming algorithms. In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, S. 2–11
- [177] LIN, J. ; KEOGH, E. ; LONARDI, S. ; LANKFORD, J. P. ; NYSTROM, D. M.: Visually mining and monitoring massive time series. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, S. 460–469
- [178] LIN, J. ; KEOGH, E.: Finding or not finding rules in time series. In: *Advances in Econometrics*. Bd. 19. Bingley : Emerald (MCB UP ), 2004. S. 175–201
- [179] LIN, J. ; KEOGH, E. ; LONARDI, S. ; LANKFORD, J. P. ; NYSTROM, D. M.: VizTree: a tool for visually mining and monitoring massive time series databases. In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30 : VLDB Endowment*, 2004. S. 1269–1272
- [180] LIN, J. ; KEOGH, E. ; LONARDI, S. ; PATEL, P.: Finding motifs in time series. In: *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002. S. 53–68
- [181] LIN, J. ; KEOGH, E. ; WEI, L. ; LONARDI, S.: Experiencing SAX: a novel symbolic representation of time series. In: *Data Mining and Knowledge Discovery* Bd. 15 (2007), Nr. 2, S. 107–144.
- [182] LIN, J. ; LI, Y.: Finding structural similarity in time series data using bag-of-patterns representation. In: *Scientific and Statistical Database Management* : Springer, 2009. S. 461–477
- [183] LIN, T.-C. ; LIN, S.-K.: Full-line FDC Diagnosis System via Signals Similarity Measure. In: *AEC/APC Symposium Asia 2009*, 2009.
- [184] LIU, H. H.: *Hadoop 2 Essentials: An End-to-End Approach*: CreateSpace Independent Publishing Platform, 2014.
- [185] LIU, H. H.: *Hadoop Essentials: A Quantitative Approach*. Charleston, S.C. : CreateSpace Independent Publishing Platform, 2012.
- [186] LIU, X. ; HAN, J. ; ZHONG, Y. ; HAN, C. ; HE, X.: Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS. In: *IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER '09*, 2009. S. 1–8
- [187] LIU, Y. ; BAO, D.-P. ; YANG, Z.-H. ; ZHAO, Y.-N. ; JIA, P.-F. ; WANG, J.-Q.: Application research of a new symbolic approximation method-SAX in time series mining. In: *Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications)* Bd. 42 (2006), Nr. 27, S. 191–193
- [188] LOEFFLER, C. ; LIGTENBERG, A. ; MOSCHYTZ, G. S.: Practical fast 1-D DCT algorithms with 11 multiplications. In: *1989 International Conference on Acoustics, Speech, and Signal Processing, 1989. ICASSP-89*, 1989, S. 988–991 Bd.2

- [189] LUBLINSKY, B. ; SMITH, K. T. ; YAKUBOVICH, A.: *Professional Hadoop Solutions*. 1. Auflage. Indianapolis : John Wiley & Sons, 2013.
- [190] Patterson, J.: Lumberyard: Time Series Indexing at Scale: OSCON 2011 - O'Reilly Conferences, July 25 - 29, 2011, Portland, OR. Online, URL: <https://conferences.oreilly.com/oscon/oscon2011/public/schedule/detail/20283>. abgerufen am 2012-11-26.
- [191] LUO, Y. ; LUO, S. ; GUAN, J. ; ZHOU, S.: A RAMCloud Storage System based on HDFS: Architecture, implementation and evaluation. In: *Journal of Systems and Software* Bd. 86 (2013), Nr. 3, S. 744–750
- [192] MAAB, H. ; ÇAKMAK, H. K. ; SÜB, W. ; QUINTE, A. ; JAKOB, W. ; MÜLLER, E. ; BÖHM, K. ; STUCKY, K. U. ; KÜHNAPFEL, U.: Introducing a New Voltage Time Series Approach for Electrical Power Grid Analysis. In: *Energy Conference and Exhibition (ENERGYCON), 2012 IEEE International*, 2012, S. 143–148
- [193] MAAB, H. ; ÇAKMAK, H. K. ; SÜB, W. ; QUINTE, A. ; JAKOB, W. ; STUCKY, K. U. ; KÜHNAPFEL, U.: Introducing the Electrical Data Recorder as a New Capturing Device for Power Grid Analysis. In: *AMPS 2012 PROCEEDINGS*, 2012.
- [194] MAAB, H. ; ÇAKMAK, H. K. ; BACH, F. ; MIKUT, R. ; HARRABI, A. ; SÜB, W. ; JAKOB, W. ; STUCKY, K.-U. ; KÜHNAPFEL, U. G. ; U. A.: Data processing of high-rate low-voltage distribution grid recordings for smart grid monitoring and analysis. In: *EURASIP Journal on Advances in Signal Processing* Bd. 2015 (2015), Nr. 1, S. 14 ff.
- [195] MAASS, H. ; ÇAKMAK, H. K. ; SÜB, W. ; QUINTE, A. ; JAKOB, W. ; STUCKY, K.-U. ; KÜHNAPFEL, U. G.: First Evaluation Results Using the New Electrical Data Recorder for Power Grid Analysis. In: *IEEE Transactions on Instrumentation and Measurement* Bd. 62 (2013), Nr. 9, S. 2384–2390
- [196] MACKEY, G. ; SEHRISH, S. ; WANG, JUN: Improving metadata management for small files in HDFS. In: *IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER '09*, 2009. S. 1–4
- [197] MACKINLAY, J.: Automating the design of graphical presentations of relational information. In: *Acm Transactions On Graphics (Tog)* Bd. 5 (1986), Nr. 2, S. 110–141.
- [198] MAKIO, K. ; TANAKA, Y. ; UEHARA, K.: Discovery of Skills from Motion Data. In: SAKURAI, A. ; HASIDA, K. ; NITTA, K. (Hrsg.): *New Frontiers in Artificial Intelligence, Lecture Notes in Computer Science* : Springer Berlin Heidelberg, 2007. S. 266–282
- [199] MALZACHER, P. ; SANDOVAL, A. ; KÖPKE, L. ; PUTZER, A. ; LANSKE, D. ; QUAST, G. ; LINDENSTRUTH, V. ; SCHMELLING, M. ; KUNZE, M. ; U. A.: Requirements for a Regional Data and Computing Centre in Germany (RDCCG) (2001). URL: <http://www.gridka.de/LHCComputing-1july01.pdf>. - abgerufen am 2015-03-30.
- [200] MARTEN, H. ; MICKEL, K.-P. ; KUPSCH, R.: A Grid Computing Centre at Forschungszentrum Karlsruhe - Response on the Requirements for a Regional Data and Computing Centre in Germany (RDCCG) (2001). URL: <http://www.gridka.de/RDCCG-answer-v8.pdf>. - abgerufen am 2015-03-30.
- [201] MARZAL, A. ; VIDAL, E.: Computation of normalized edit distance and applications. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* Bd. 15 (1993), Nr. 9, S. 926–932

- [202] MCGOVERN, A. ; ROSENDAHL, D. ; KRUGER, A. ; BEATON, M. ; BROWN, R. ; DROEGEMEIER, K.: Understanding the formation of tornadoes through data mining. In: *5th conference on artificial intelligence and its applications to environmental sciences at the American meteorological society*, 2007.
- [203] MENDELEVITCH, O. ; STELLA, C. ; EADLINE, D.: *Data Science with Hadoop*. S.l. : Addison Wesley Pub Co Inc, 2015.
- [204] MEYER, J.: Archival Services and Technologies for Scientific Data. In: *Proceedings of CHEP2013 (20th International Conference on Computing in High Energy and Nuclear Physics)*. Amsterdam, The Netherlands (2013).
- [205] MIKUT, R.: *Data Mining in der Medizin und Medizintechnik, Schriftenreihe des Instituts für Angewandte Informatik/Automatisierungstechnik an der Universität Karlsruhe (TH)*. Karlsruhe : Univ.-Verl, 2008.
- [206] MINER, D.: *Enterprise Hadoop* : O'Reilly & Assoc Inc, 2015.
- [207] MINER, D. ; SHOOK, A.: *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems*. 1. Aufl. : O'Reilly Media, 2012.
- [208] MINER, D. ; SHOOK, A.: *MapReduce Design Patterns : Building Effective Algorithms and Analytics for Hadoop and Other Systems*. 2. Aufl. : Donald Miner & Adam Shook, 2013.
- [209] *Mining Time-series with Trillions of Points: Dynamic Time Warping at scale (euklid vs dtw)*. URL <http://practicalquant.blogspot.kr/2012/10/mining-time-series-with-trillions-of.html>. - abgerufen am 2014-07-18.
- [210] MINNEN, D. ; STARNER, T. ; ESSA, I. A. ; ISBELL JR, C. L.: Improving Activity Discovery with Automatic Neighborhood Estimation. In: *IJCAI*. Bd. 7, 2007.
- [211] MOCZAR, L.: *Enterprise Hadoop and Mapreduce* : Addison Wesley Pub Co Inc, 2016.
- [212] MOERCHEN, F.: Algorithms for time series knowledge mining. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* : ACM, 2006. S. 668–673
- [213] MOLINA-MARKHAM, A. ; SHENOY, P. ; FU, K. ; CECCHET, E. ; IRWIN, D.: Private memoirs of a smart meter. In: *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, ACM New York, NY, USA, 2010. S. 61–66
- [214] MORGAN, I. ; LIU, H.: Predicting Future States With-Dimensional Markov Chains for Fault Diagnosis. In: *Industrial Electronics, IEEE Transactions on* Bd. 56 (2009), Nr. 5, S. 1774–1781.
- [215] MORGAN, I. ; LIU, H. ; TURNBULL, G. ; BROWN, D.: Time discretisation applied to anomaly detection in a marine engine. In: *Knowledge-Based Intelligent Information and Engineering Systems* : Springer, 2007. S. 405–412
- [216] MÜLLER, G. ; PONICK, B.: *Theorie elektrischer Maschinen* : John Wiley & Sons, Limited, 2009.
- [217] MURTHY, A. C. ; VAVILAPALLI, V. K. ; EADLINE, D.: *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2*. Upper Saddle River, NJ : Addison Wesley Pub Co Inc, 2014.
- [218] NARAYANAN, S.: *Securing Hadoop*. Birmingham, UK : Packt Publishing, 2013.

- [219] NIELSEN, L.: *Hadoop: The Engine That Drives Big Data* : New Street Communications, LLC, 2013.
- [220] NORDBERG, H. ; BHATIA, K. ; WANG, K. ; WANG, Z.: BioPig: a Hadoop-based analytic toolkit for large-scale sequence data. In: *Bioinformatics* Bd. 29 (2013), Nr. 23, S. 3014–3019.
- [221] OHSAKI, M. ; SATO, Y. ; YOKOI, H. ; YAMAGUCHI, T.: A rule discovery support system for sequential medical data, in the case study of a chronic hepatitis dataset. In: *Proceedings of International Workshop on Active Mining*, 2002. S. 97–102
- [222] OKABE, M. ; MIWA, T. ; UMEMURA, K.: Anomaly Detection in Network Traffic based on String Analysis. In: *Proceedings of IC2006*. Bd. 2006, 2006. S. 67–74
- [223] OOI, B. Y.: CPU Usage Pattern Discovery Using Suffix Tree For Computational Resource Advisory System. In: *Distributed Frameworks for Multimedia Applications, 2006. The 2nd International Conference on*: IEEE, 2006. S. 1-8
- [224] ORDÓÑEZ, P. ; DES JARDINS ; M. FELTES, C. LEHMANN C. U. ; FACKLER, J.: Visualizing multivariate time series data to detect specific medical conditions. In: *AMIA Annual Symposium Proceedings*. Bd. 2008 : American Medical Informatics Association, 2008. S. 530-534
- [225] ORIANI, A. ; GARCIA, I. C.: From Backup to Hot Standby: High Availability for HDFS. In: *Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems, SRDS '12*. Washington, DC, USA : IEEE Computer Society, 2012. S. 131–140
- [226] OSSENDRIJVER, M.: *Babylonian Mathematical Astronomy: Procedure Texts: Procedure Texts* : Springer Science & Business Media, 2012.
- [227] OWENS, J. R ; LENTZ, J. ; FEMIANO, B.: *Hadoop real-world solutions cookbook*. Birmingham : Packt Pub., 2013.
- [228] PALENBERG, A.: *Informationsbroschüre zum Aufbau des Stromnetzes*. URL [http://www.forum-netzintegration.de/uploads/media/DUH\\_Kurzinfo\\_Stromnetzaufbau.pdf](http://www.forum-netzintegration.de/uploads/media/DUH_Kurzinfo_Stromnetzaufbau.pdf). - abgerufen am 2014-11-24.
- [229] PANG, K. F. ; EL GAMAL, A.: Communication complexity of computing the Hamming distance. In: *SIAM Journal on Computing* Bd. 15 (1986), Nr. 4, S. 932–947
- [230] PARSIAN, M.: *Data Algorithms: Recipes for Scaling Up with Hadoop and Spark*. 1. Aufl. : O'Reilly & Associates, 2015.
- [231] PATEL, P. ; KEOGH, E. ; LIN, J. ; LONARDI, S.: Mining motifs in massive time series databases. In: *2002 IEEE International Conference on Data Mining, 2002. ICDM 2003. Proceedings*, 2002. S. 370–377
- [232] PAVLO, A. ; PAULSON, E. ; RASIN, A. ; ABADI, D. J. ; DEWITT, D. J. ; MADDEN, S. ; STONEBRAKER, M.: A comparison of approaches to large-scale data analysis. In: *Proceedings of the 35th SIGMOD international conference on Management of data*, 2009, S. 165–178
- [233] PETERSOHN, H.: Kapitel 11: Einführung in die Zeitreihenanalyse. In: *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur* : Oldenbourg Verlag, 2005.

- [234] PHADKE, A.G. ; THORP, J.S.: History and applications of phasor measurements. In: *Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES*, Atlanta, Georgia, USA, 2006, S. 331–335
- [235] PHADKE, A. G. ; THORP, J. S.: *Synchronized phasor measurements and their applications* : Springer US, 2008.
- [236] *PiggyBank - Apache Pig - Apache Software Foundation*. URL <https://cwiki.apache.org/confluence/display/PIG/PiggyBank>. - abgerufen am 2015-04-21.
- [237] PLUNKETT, T. ; NELSON, B. ; MACDONALD, B.: *Hadoop and Oracle: The Why, When & How Guide* : Osborne, 2015.
- [238] POUGET, F.: *Distributed system of honeypot sensors: discrimination and correlative analysis of attack processes*, PhD thesis, Paris Institute of Technology, Institut Eurecom, Paris, Frankreich, 2006.
- [239] POUGET, F. ; URVOY-KELLER, GUILLAUME ; DACIER, MARC: Time signatures to detect multi-headed stealthy attack tools. In: *Proceedings of the 18th annual first conference, Baltimore, MD, June, 2006*. S. 25–30
- [240] *PoweredBy - Hadoop Wiki*. URL <https://wiki.apache.org/hadoop/PoweredBy>. - abgerufen am 2015-04-16.
- [241] PRAJAPATI, V.: *Big Data Analytics with R and Hadoop*. Birmingham, UK : Packt Publishing, 2013.
- [242] *Processing.org*. URL <https://processing.org/>. - abgerufen am 2015-05-13.
- [243] RAKTHANMANON, T. ; CAMPANA, B. ; MUEEN, A. ; BATISTA, G. ; WESTOVER, B. ; ZHU, Q. ; ZAKARIA, J. ; KEOGH, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. (Beijing, China) : ACM New York, NY, USA, 2012, S. 262–270
- [244] RATANAMAHATANA, C. A. ; LIN, J. ; GUNOPULOS, D. ; KEOGH, E. ; VLACHOS, M. ; DAS, G.: Mining Time Series Data. In: MAIMON, O. ; ROKACH, L. (Hrsg.): *Data Mining and Knowledge Discovery Handbook*. Boston, MA : Springer US, 2009. S. 1049–1077
- [245] REINSEL, D. ; CHUTE, C. ; SCHLICHTING, W. ; MCARTHUR, J. ; MINTON, S. ; XHENETI, I. ; TONCHEVA, A. ; MANFREDIZ, A.: *The Expanding Digital Universe - A Forecast of Worldwide Information Growth Through 2010*. White paper, IDC (Proj. Dir. GANTZ, J. F.), Framingham, Massachusetts, USA (2007). URL [https://www.tobb.org.tr/BilgiHizmetleri/Documents/Raporlar/Expanding\\_Digital\\_Universe\\_IDC\\_WhitePaper\\_022507.pdf](https://www.tobb.org.tr/BilgiHizmetleri/Documents/Raporlar/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf). - abgerufen am 2015-03-20
- [246] RENNERT, I. ; BUNDSCHUH, B.: *Signale und Systeme: Einführung in die Systemtheorie* : Carl Hanser Verlag GmbH & Co. KG, 2013.
- [247] RUDNIK, S. ; PELTA, R.: *Der Lotse durch die DIN VDE 0100 Eine Navigation durch die 22 Hauptteile der DIN VDE 0100 „Errichten von Niederspannungsanlagen“*. Berlin : VDE VERLAG, 2014.
- [248] SADOWSKI, M.: *Aktive Spannungsregelung im elektrischen Verteilnetz*, ABB-APR/AVPE, Produktbereich Power Electronics, VDE-Arbeitskreis Energietechnik, Schweiz (2012).

- [249] SALVADOR, S. ; CHAN, P. ; BRODIE, J.: Learning States and Rules for Time Series Anomaly Detection. In: *FLAIRS Conference*, Miami Beach, Florida, 2004. S. 306–311
- [250] SAMMER, E.: *Hadoop Operations* : O'Reilly Media, 2012.
- [251] SANCHEZ, J. R. ; ARIZMENDI, C. M.: Fractal Analysis of Electrical Power Time Series. In: *Behavioral, Economic and Socio-cultural Computing (BESC), 2016 International Conference on*. Durham, NC, USA: IEEE, (arXiv e-print Nr. cond-mat/9912112), 1999.
- [252] SATAPATHY, S. K. ; MISHRA, S. ; MISHRA, D.: Search Technique Using Wildcards or Truncation: A Tolerance Rough Set Clustering Approach. In: *International Journal of Advanced Computer Sciences and Applications* Bd. 1 (2010), Nr. 4.
- [253] SAX. URL <http://www.cs.ucr.edu/~eamonn/SAX.htm>. - abgerufen am 2012-04-25
- [254] SCHEFF, J. D. ; ALMON, R. R. ; DUBOIS, D. C. ; JUSKO, W. J. ; ANDROULAKIS, I. P.: A New Symbolic Representation for the Identification of Informative Genes in Replicated Microarray Experiments. In: *OmicS: a journal of integrative biology* Bd. 14 (2010), Nr. 3, S. 239–248.
- [255] SCHIPMAN, K. ; DELINCÉ, F.: *The Importance of good Power Quality* (technischer Bericht). Belgien : ABB Power Quality Products, 2003.
- [256] *Schlechte Stromqualität birgt Sicherheitsrisiken*. URL <http://www.ingenieur.de/Themen/IT-Sicherheit/Schlechte-Stromqualitaet-birgt-Sicherheitsrisiken>. - abgerufen am 2014-11-27.
- [257] SENIN, P. ; LIN, J. ; WANG, X. ; OATES, T. ; GANDHI, S. ; BOEDIHARDJO, A. P. ; CHEN, C. ; FRANKENSTEIN, S. ; LERNER, M.: GrammarViz 2.0: A Tool for Grammar-Based Pattern Discovery in Time Series. In: *Machine Learning and Knowledge Discovery in Databases* : Nancy, France. Springer, 2014. S. 468–472
- [258] SENIN, P. V.: Literature Review on Time Series Indexing. In: *CSDL Technical Report 09-08*. Honolulu, HI, 2009.
- [259] SHAFER, J. ; RIXNER, S. ; COX, A. L.: The Hadoop distributed filesystem: Balancing portability and performance. In: *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on* : White Plains, New York, USA. IEEE, 2010. S. 122–133
- [260] SHATKAY, H. ; ZDONIK, S.B.: Approximate queries and representations for large data sequences. In: *Proceedings of the Twelfth International Conference on Data Engineering, 1996*, New Orleans, Louisiana, IEEE, 1996, S. 536–545
- [261] SHAW, S. ; GROSS, C. ; GUPTA, A.: *Practical Hive: A Guide to Hadoop's Data Warehouse System*. 2015. Aufl. : Apress, 2015.
- [262] SHEN, Q. ; YANG, Y. ; WU, Z. ; YANG, X. ; ZHANG, L. ; YU, X. ; LAO, Z. ; WANG, D. ; LONG, M.: SAPSC: Security Architecture of Private Storage Cloud Based on HDFS. In: *2012 26th International Conference on Advanced Information Networking and Applications Workshops (AINA)*, Fukuoka, Japan, 2012. S. 1292–1297
- [263] SHENOY, A.: *Hadoop Explained* : Packt Publishing, 2014.
- [264] SHNEIDERMAN, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: *Visual Languages, 1996. Proceedings, IEEE Symposium on*. Boulder, Colorado : IEEE Computer Society Washington, DC, USA, 1996.



- [265] SHVACHKO, K. ; KUANG, H. ; RADIA, S. ; CHANSLER, R.: The Hadoop Distributed File System. In: *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Lake Tahoe, Nevada, USA, 2010. S. 1–10
- [266] SHVACHKO, K. V.: HDFS scalability: the limits to growth. In: *"login:" - the usenix magazine*. Bd. 2 (2010), Nr. April 2010, Volume 35, Number 2.
- [267] DA SILVA, J. C. ; KLUSCH, M.: Privacy-preserving discovery of frequent patterns in time series. In: *Advances in Data Mining. Theoretical Aspects and Applications* : Springer, 2007. S. 318–328
- [268] SILVA, P. M. ; BECERRA, V. M. ; CALADO, J. M. F. ; U. A.: Clustering techniques applied to multiple-models structures. In: *Proceedings of the 7th Workshop on Advanced Control and Diagnosis*, Zielona Góra, Polen, 2009. S. 1–6
- [269] SILVENT, A.-S. ; CARBAY, C. ; CARRY, P.-Y. ; DOJAT, M.: Data, Information and Knowledge for Medical Scenario Construction. In: *Proceedings of the intelligent data analysis in medicine and pharmacology workshop, Protaras, Cyprus, October, 2003*. S. 19–22
- [270] SILVENT, A.-S. ; DOJAT, M. ; GARBAY, C.: Multi-level temporal abstraction for medical scenario construction. In: *International Journal of Adaptive Control and Signal Processing* Bd. 19 (2005), Nr. 5, S. 377–394
- [271] SINGLA, M. ; PODDAR, S.: *Hadoop Interview Guide (Kindle Edition)* : Amazon Digital Services, 2015.
- [272] SITTO, K. ; PRESSER, M.: *Field Guide to Hadoop: An Introduction to Hadoop, Its Ecosystem, and Aligned Technologies*. 1. Aufl. : Sebastopol, CA, USA, O'Reilly Media, 2015.
- [273] SKILLICORN, D. B. ; TALIA, D.: Models and Languages for Parallel Computation. In: *ACM Comput. Surv.* Bd. 30 (1998), Nr. 2, S. 123–169
- [274] SOUZA, L. R.: Why Aggregate Long Memory Time Series? In: *Econometric Reviews* Bd. 27 (2008), Nr. 1–3, S. 298–316
- [275] SPIVEY, B. ; ECHEVERRIA, J.: *Hadoop Security: Protecting Your Big Data Platform*. 1. Aufl. : O'Reilly, 2015.
- [276] STIEFMEIER, T. ; ROGGEN, D. ; TRÖSTER, G.: Gestures Are Strings: Efficient Online Gesture Spotting and Classification Using String Matching. In: *Proceedings of the ICST 2Nd International Conference on Body Area Networks, BodyNets '07*. ICST, Brussels, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007. Nr. 16, S. 1–8
- [277] STOTZKA, R. ; HARTMANN, V. ; JEJKAL, T. ; SUTTER, M. ; VAN WEZEL, J. ; HARDT, M. ; GARCIA, A. ; KUPSCH, R. ; BOUROV, S.: Perspective of the Large Scale Data Facility (LSDF) supporting nuclear fusion applications. In: *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, Ayia Napa, Cyprus, 2011. S. 373–379
- [278] STOTZKA, R. ; MEXNER, W. ; DOS SANTOS ROLO, T. ; PASIC, H. ; VAN WEZEL, J. ; HARTMANN, V. ; JEJKAL, T. ; GARCIA, A. ; HAAS, D. ; U. A.: Large Scale Data Facility for Data Intensive Synchrotron Beamlines. In: *13th International Conference on Accelerator and Large Experimental Physics Control Systems*, Grenoble, France, 2011.

- [279] STOYAN, D.: *Stochastik für Ingenieure und Naturwissenschaftler: Eine Einführung in die Wahrscheinlichkeitstheorie und die Mathematische Statistik*. Auflage: 1. Auflage. Aufl. : Wiley-VCH Verlag GmbH & Co. KGaA, 1993.
- [280] *Stromnetz der Zukunft: Forum Netzintegration Erneuerbare Energien*. URL <http://www.forum-netzintegration.de/54/>. - abgerufen am 2014-11-24.
- [281] STRZELECKI, R. ; BENYSEK, G. (Hrsg.): *Power electronics in smart electrical energy networks, Power systems*. London : Springer, 2008.
- [282] SUTTER, M. ; HARTMANN, V. ; GÖTTER, M. ; VAN WEZEL, J. ; TRUNOV, A. ; JEJKAL, T. ; STOTZKA, R.: File Systems and Access Technologies for the Large Scale Data Facility. In: DAVOLI, F. ; LAWENDA, M. ; MEYER, N. ; PUGLIESE, R. ; WEGLARZ, J. ; ZAPPATORE, S. (Hrsg.): *Remote Instrumentation for eScience and Related Aspects*. New York, NY : Springer New York, 2012. S. 239–256
- [283] TAN, P.-N. ; STEINBACH, M. ; KUMAR, V.: *Introduction to Data Mining*. 1. Edition. Aufl. Boston : Pearson, 2005.
- [284] TANAKA, Y. ; UEHARA, K.: Discover Motifs in Multi-dimensional Time-Series Using the Principal Component Analysis and the MDL Principle. In: PERNER, P. ; ROSENFELD, A. (Hrsg.): *Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science*. Bd. 2734 : Springer Berlin / Heidelberg, 2003. S. 297–322
- [285] TANTISIRIROJ, W. ; SON, S. W. ; PATIL, S. ; LANG, S. J. ; GIBSON, G. ; ROSS, R. B.: On the Duality of Data-intensive File System Design: Reconciling HDFS and PVFS. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11*. New York, NY, USA : ACM, 2011. Nr. 67, S. 1–12
- [286] TEAM, O'REILLY RADAR: *Big Data Now: Current Perspectives from O'Reilly Radar: Current Perspectives from O'Reilly Radar* : O'Reilly Media, 2011.
- [287] *The Hadoop Ecosystem Table*. URL <https://hadoopecosystemtable.github.io/>. - abgerufen am 2015-04-16.
- [288] *The Large Hadron Collider | CERN*. URL <http://home.web.cern.ch/topics/large-hadron-collider>. - abgerufen am 2015-03-30.
- [289] *The Open Source Phasor Data Concentrator (OpenPDC)*. URL <http://openpdc.codeplex.com/>. - abgerufen am 2012-04-25
- [290] *The Smart Grid: Hadoop at the Tennessee Valley Authority (TVA) - cloudera, Josh Patterson*. URL <http://www.cloudera.com/blog/2009/06/smart-grid-hadoop-tennessee-valley-authority-tva/>. - abgerufen am 2012-08-15.
- [291] TIANYU LI ; FANG-YAN DONG ; KAORU HIROTA: Distance Measure for Symbolic Approximation Representation with Subsequence Direction for Time Series Data Mining. In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* Bd. Vol.17 No.2 (2013), S. 263–271.
- [292] *TimeseriesClassification - jmotif - Classification of timeseries - A time series data-mining toolkit based on SAX and TFIDF statistics. Implements SAX-VSM*. - Google Project Hosting. URL <http://code.google.com/p/jmotif/wiki/TimeseriesClassification>. - abgerufen am 2013-09-06.

- [293] TOMINSKI, C. ; SCHUMANN, H.: Enhanced interactive spiral display. In: *Proceedings of the Annual SIGRAD Conference, Special Theme: Interactivity*, 2008, S. 53–56
- [294] *Tree Visualization*. URL <http://www.randelshofer.ch/treeviz/>. - abgerufen am 2015-04-24.
- [295] TRUJILLO, G. ; KIM, C. ; JONES, S.: *Virtualizing Hadoop: How to Install, Deploy, and Optimize Hadoop in a Virtualized Architecture* : Financial Times Prentice Hall, 2015.
- [296] TUFTE, E. R.: *Visual Display of Quantitative Information*. Auflage: 2. Aufl. Cheshire, Conn : Bertrams, 2001.
- [297] TUFTE, E. R.: *Beautiful Evidence*. 1st edition edition. Aufl. Cheshire, Conn : Graphics Press, 2006.
- [298] TUKEY, J. W.: *Exploratory Data Analysis*. 1. Edition. Aufl. Reading, Mass : Pearson, 1977.
- [299] TURKINGTON, G. ; MODENA, G.: *Learning Hadoop 2* : Packt Publishing, 2015.
- [300] *TVA: Home Page*. URL <http://www.tva.gov/>. - abgerufen am 2015-01-23.
- [301] UEHARA, K. ; TANAKA, Y. ; MAKIO, K.: Motif discovery algorithm from motion data. In: *Annual conference on JSAI*. Bd. 18, 2004.
- [302] UKKONEN, E.: On-line construction of suffix trees. In: *Algorithmica* Bd. 14 (1995), Nr. 3, S. 249–260.
- [303] VAN WIJK, J. J. ; VAN SELOW, E. R.: Cluster and calendar based visualization of time series data. In: *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, 1999. S. 4–9
- [304] VAN DANIKER, M.: Leverage of Spiral Graph for Transportation System Data Visualization. In: *Transportation Research Record: Journal of the Transportation Research Board* Bd. 2165 (2010), Nr. 1, S. 79–88
- [305] VENDRUSCULO, L. G. ; DE MEDEIROS OLIVEIRA, S. R. ; ESQUERDO, J. C. D. M. ; ANTUNES, J. F. G.: Análise de padrões seqüenciais em série histórica do rio Paraguai. In: *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)* : In: SIMPÓSIO DE GEOTECNOLOGIAS NO PANTANAL, 2., 2009, Corumbá, MS. Anais... Campinas: Embrapa Informática Agropecuária; São José dos Campos: INPE, 2009. 2010.
- [306] VENNER, J.: *Pro Hadoop*. Berkeley, CA : Apress, 2009. S. 323-332
- [307] VENNER, J.: *Pro hadoop 2<sup>nd</sup> edition*. Berkeley, CA : Apress, 2013.
- [308] VERLEYSSEN, M. ; FRANÇOIS, D.: The Curse of Dimensionality in Data Mining and Time Series Prediction. In: CABESTANY, J. ; PRIETO, A. ; SANDOVAL, F. (Hrsg.): *Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science* : Springer Berlin Heidelberg, 2005. S. 758–770
- [309] VITANYI, P.M.B. ; LI, M.: Minimum description length induction, Bayesianism, and Kolmogorov complexity. In: *IEEE Transactions on Information Theory* Bd. 46 (2000), Nr. 2, S. 446–464.
- [310] VLACHOS, M. ; KOLLIOS, G. ; GUNOPULOS, DIMITRIOS: Discovering similar multidimensional trajectories. In: *Data Engineering, 2002. Proceedings. 18th International Conference on* : IEEE, 2002, S. 673–684
- [311] WADKAR, S. ; VENNER, J.: *Pro Apache Hadoop*. 2. Aufl. : Apress, 2014.

- [312] WAGMOB: *Big Data and Hadoop* : WAGmob; 1.5 edition, 2013.
- [313] WALSH, J. E.: Some Nonparametric Tests of Whether the Largest Observations of a Set are too Large or too Small. In: *The Annals of Mathematical Statistics* Bd. 21 (1950), Nr. 4, S. 583–592
- [314] WANG, X. ; CANDAN, K. S.: Relevant shape contour snippet extraction with metadata supported hidden Markov models. In: *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR'10, Xi'an, China)* : ACM New York, NY, USA, 2010. S. 430–437
- [315] WARASUP, K. ; NUKOOLKIT, C.: Discovery association rules in time series data. In: *KMUTT Research and Development Journal*. Bd. 29 (2012), Nr. 4 : King Mongkut's University of Technology Thonburi (KMUTT), Thailand, S. 447–462
- [316] WARDEN, P.: *Big Data Glossary, 1<sup>st</sup> edition* : O'Reilly Media, 2011.
- [317] WARTALA, R.: *Hadoop: Zuverlässige, verteilte und skalierbare Big-Data-Anwendungen*. München : Open Source Press, 2012.
- [318] WEBER, M. ; ALEXA, M. ; MÜLLER, W.: Visualizing time-series on spirals. In: *proceedings of the IEEE Symposium on Information Visualization (Infovis 2001)*, San Diego, California, USA, 2001, S. 7 ff.
- [319] WEI, L. ; KUMAR, N. ; LOLLA, V. N. ; KEOGH, E. ; LONARDI, S. ; RATANAMAHATANA, C.: Assumption-Free Anomaly Detection in Time Series. In: *SSDBM'2005 Proceedings of the 17th international conference on Scientific and statistical database management*. Bd. 5, Santa Barbara, CA, USA, 2005. S. 237–242
- [320] WEINER, P.: Linear pattern matching algorithms. In: *SWAT '73 Proceedings of the 14th Annual Symposium on Switching and Automata Theory (swat 1973)* : IEEE, Washington, DC, USA, 1973. S. 1–11
- [321] *Welcome to Apache Pig!* URL <http://pig.apache.org/>. - abgerufen am 2013-07-18.
- [322] *Welcome to Apache™ Hadoop®!* URL <http://hadoop.apache.org/>. - abgerufen am 2015-03-12.
- [323] *What Is Big Data?* - Gartner IT Glossary. URL <http://www.gartner.com/it-glossary/big-data/>. - abgerufen am 2015-03-18.
- [324] WHITE, T.: *Hadoop: The Definitive Guide (2<sup>nd</sup> edition)* : O'Reilly Media / Yahoo Press, Sebastopol, CA, USA, 2009.
- [325] WHITE, T.: *Hadoop: the definitive guide (1<sup>st</sup> edition)* : O'Reilly Media / Yahoo Press, Sebastopol, CA, USA, 2010.
- [326] WHITE, T.: *Hadoop: The Definitive Guide (3<sup>rd</sup> edition)*. : O'Reilly Media / Yahoo Press, Sebastopol, CA, USA, 2012.
- [327] WHITE, TOM: *Hadoop: The Definitive Guide (3<sup>rd</sup> edition)*. 4. Aufl. : O'Reilly & Associates, Sebastopol, CA, USA, 2015.
- [328] WIJAYA, T. K. ; EBERLE, J. ; ABERER, K.: Symbolic representation of smart meter data. In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, Genoa, Italy : ACM, 2013. S. 242–248
- [329] WILBER, L.: *Search Technology in the Era of Big Data* : Morgan Claypool, 2012.

- [330] WILSON, W. ; BIRKIN, P. ; AICKELIN, U.: Motif detection inspired by immune memory. In: *Artificial Immune Systems* : Springer, 2007. S. 276–287
- [331] WILSON, W. O. ; FEYEREISL, J. ; AICKELIN, U.: Detecting motifs in system call sequences. In: *Information Security Applications* : Springer, 2007. S. 157–172
- [332] WU, H.-W. ; LEE, A. J.: Mining closed flexible patterns in time-series databases. In: *Expert Systems with Applications* Bd. 37 (2010), Nr. 3, S. 2098–2107
- [333] YAIK, O. B. ; YONG, C. H. ; HARON, F.: CPU Usage Pattern Discovery Using Suffix Tree. In: *Distributed Frameworks for Multimedia Applications, 2006. The 2nd International Conference on* : IEEE, 2006. S. 1–8
- [334] YANAI, K. ; YANASE, T.: Analysis and Learning Frameworks for Large-Scale Data Mining. In: KARAHOCA, A. (Hrsg.): *Advances in Data Mining Knowledge Discovery and Applications* : InTech, 2012.
- [335] YANG, E., F. BERTHIAMUME, M. L. YARMUSH, AND I. P. ANDROULAKIS: Selection of Informative Genes via Symbolic Hashing Of Time Series. In: *Proceedings of the Joint 9th International Symposium on Processing Systems Engineering and 16th European Symposium, 2006.*
- [336] YANG, E. ; MISRA, A. ; MAGUIRE, T. J. ; ANDROULAKIS, I. P.: Analysis of Regulatory Interaction Networks from Clusters of Co-expressed Genes. In: *Clustering Challenges in Biological Networks* (2009), S. 53
- [337] YANG, E. ; ANDROULAKIS, I.: Selection of Informative Genes in Time-Course Gene Expression Data. In: *The 2006 AIChE Annual Meeting*, San Francisco, CA, USA, 2006, Nr. 124d
- [338] YANKOV, D. ; KEOGH, E. ; REBBAPRAGADA, U.: Disk aware discord discovery: finding unusual time series in terabyte sized datasets. In: *Knowledge and Information Systems* Bd. 17 (2008), Nr. 2, S. 241–262
- [339] YI, B.-K. ; FALOUTSOS, C.: Fast Time Sequence Indexing for Arbitrary Lp Norms. In: *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB '00.* San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000, S. 385–394
- [340] ZBURIVSKY, D.: *Hadoop Cluster Deployment* : Packt Publishing, 2013.
- [341] ZHANG, X. ; WU, J. ; YANG, X. ; OU, H. ; LV, T.: A novel pattern extraction method for time series classification. In: *Optimization and Engineering* Bd. 10 (2009), Nr. 2, S. 253–271
- [342] ZIKOPOULOS, P. C. ; EATON, C. ; ZIKOPOULOS, P.: *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data* : Mcgraw-Hill Professional, 2012.
- [343] ZOUMBOULAKIS, M. ; ROUSSOS, G.: Escalation: Complex event detection in wireless sensor networks. In: *Smart Sensing and Context* : Springer, 2007, S. 270–285

# Formelverzeichnis

*Formel (2.1)*..... 25  
*Formel (2.2)*..... 25  
*Formel (3.1)*..... 60  
*Formel (A.1)*..... A-2  
*Formel (A.2)*..... A-3  
*Formel (A.3)*..... A-10  
*Formel (A.4)*..... A-10  
*Formel (A.5)*..... A-10  
*Formel (A.6)*..... A-20  
*Formel (A.7)*..... A-20  
*Formel (A.8)*..... A-21  
*Formel (A.9)*..... A-21  
*Formel (A.10)*..... A-21  
*Formel (A.11)*..... A-21  
*Formel (A.12)*..... A-23  
*Formel (A.13)*..... A-23  
*Formel (A.14)*..... A-25  
*Formel (A.15)*..... A-25  
*Formel (A.16)*..... A-31  
*Formel (A.17)*..... A-33  
*Formel (B.1)*..... B-12

# Anhang A - Bewertung und Auswahl von Methoden zur skalierbaren Analyse umfangreicher Zeitreihendaten

In diesem Kapitel werden geeignete Methoden, die zum Aufbau des in Kapitel 2 beschriebenen Konzepts benötigt werden, identifiziert, diskutiert und hinsichtlich ihrer Skalierbarkeit bewertet. Zunächst werden relevante Aufgaben und Methoden der Zeitreihenanalyse ermittelt und gegenübergestellt, um eine Zeitreihenrepräsentation auszuwählen, die sich für die benötigten neuen Analyseaufgaben eignet und gut mit großen Datenmengen skaliert. Als nächstes werden Ansätze und Frameworks für eine interaktive Visualisierung von Zeitreihen untersucht und es wird ermittelt, welche fehlende Funktionalität im Hinblick auf die visuelle Analyse und Exploration sehr großer Zeitreihendatenbanken neu zu entwickeln ist. Im Anschluss werden Verfahren und Technologien des Data-Intensive Computings vorgestellt und eine geeignete Infrastruktur und Software-Umgebung gewählt, auf welcher das neue Konzept zur skalierbaren Organisation und Auswertung großer Zeitreihendaten der Arbeit umsetzbar ist.

## A.1 Skalierbarkeitsbewertung bestehender Zeitreihenanalyseverfahren

Die Analyse großer Zeitreihendaten wie der in dieser Arbeit untersuchten Energie-Messdaten des KIT, ist ein komplexes Themenfeld, das sich der Methoden vieler verschiedener Forschungsgebiete bedient. Neben den anwendungsspezifischen Aspekten zur Energiedatenanalyse sind dies naturgemäß die Gebiete der klassischen Signalverarbeitung, der Zeitreihenanalyse und der Systemtheorie, welche sich etwa mit der Trendbereinigung, Vorhersage und Modellbildung befassen und z. B. in [164], [246] und [233] ausführlich behandelt werden.

Aktuell existieren für die diversen Anwendungsgebiete unterschiedliche Ansätze, welche, je nach Größe der typischen Datensätze innerhalb des jeweiligen Bereichs, mehr oder weniger von der Skalierungsproblematik bei großen Datenmengen betroffen sind. Im Fokus dieser Arbeit werden vor allem gut mit der Datengröße skalierende Ansätze betrachtet, welche Analysen von Zeitreihendaten prinzipiell unbegrenzter Größe erlauben. Effiziente Analysen sollen auch bei einem schnellen Zuwachs der Datengröße möglich sein. Ansätze, welche einerseits auf dimensionsredu-

zierten Datenrepräsentationen für Zeitreihen basieren und sich andererseits gut für eine skalierbare datenparallele und verteilte Verarbeitung im Sinne des DIC eignen, zeigen sich hierfür als besonders gut geeignet. Die folgenden Abschnitte führen die hierzu benötigten Grundlagen ein und motivieren aktuelle Aufgaben der Analyse großer Zeitreihensammlungen und die Wahl einer hierfür geeigneten neuartigen Zeitreihenrepräsentation, aus der in Abs. 3.6 und 3.7 neue Zeitreihenanalyseverfahren abgeleitet werden. A.1.1 Klassische Zeitreihenanalyse

Die klassische Zeitreihenanalyse beschäftigt sich als Form der Regressionsanalyse mit der statistischen Auswertung und Vorhersage von Zeitreihen (ZR). Die hierbei eingesetzten Methoden operieren meist direkt auf reellwertigen Zeitreihen.

#### A.1.1.1 Reellwertige Zeitreihen

Eine Zeitreihe  $X$  beschreibt die Veränderung in der Ausprägung eines quantitativ beobachteten<sup>45</sup> Merkmals  $M$  über die Zeit, bzw. einen festgelegten Zeitraum als Sequenz  $n$  aufeinanderfolgender reeller Werte:

$$x_i = (x_1, x_2, \dots, x_n), \quad x_i \in \mathbb{R}. \quad (\text{A.1})$$

Diese Werte können aus Messungen stammen oder durch Abtastung bereits vorhandener Aufzeichnungen gewonnen werden. Häufig wird für jeden Datenpunkt zusätzlich der Messzeitpunkt (*Zeitstempel*, engl. *Timestamp*) oder -zeitraum gespeichert, v. a. dann, wenn die Messungen zeitlich nicht exakt äquidistant sind. Die meisten Verfahren der Zeitreihenanalyse gehen implizit jedoch von äquidistanten Zeitschritten aus, weshalb abweichende Zeitreihendaten zunächst entsprechend vorverarbeitet werden müssen.

#### A.1.1.2 Zeitreihendatenbanken

Eine Sammlung von Zeitreihen kann als *Zeitreihendatenbank* (kurz ZR-DB; engl. *Time Series Database*, kurz TSDB) bezeichnet werden. Zeitreihendatenbanken dienen meistens dazu, Gemeinsamkeiten und Unterschiede zwischen den einzelnen ZR herauszustellen, bzw. Ähnlichkeitsbeziehungen zwischen den einzelnen ZR zu ermitteln und darauf aufbauende Abfragen und Analysen zu ermöglichen. In der Literatur wird meist von einer Sammlung gleich langer Zeitreihen ausgegangen, was jedoch in der Praxis häufig nicht der Fall ist. Zur Ermittlung von Ähnlichkeits-

---

<sup>45</sup> Oftmals werden auch qualitativ beobachtete Merkmale mit ordinalskalierten Werten im übertragenen Sinne als Zeitreihen dargestellt, um sie den Methoden klassischer Zeitreihenanalyse zugänglich zu machen, was jedoch strenggenommen nicht korrekt ist. Umgekehrt können metrische Werte jedoch diskretisiert und damit ordinal skaliert werden, um sie entsprechenden Methoden zugänglich zu machen (siehe z. B. die in Anhang A.1.8 beschriebene Symbolische Aggregat-Approximation).



beziehungen zwischen ZR unterschiedlicher Länge existieren nur einige wenige spezielle Verfahren (z. B. das in Anh. A.1.5 dargestellte *Dynamic Time Warping*, sowie das Verfahren der *Längsten Gemeinsamen Sequenz*).

Eine Datenbank von  $m$  ZR mit gleicher Länge  $n$  kann gemäß [51] repräsentiert werden durch eine  $m \times n$  Matrix der Form

$$ZRDB = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,j} & \cdots & x_{m,n} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_m \end{pmatrix}. \quad (\text{A.2})$$

Liegen hingegen  $m$  ZR unterschiedlicher Länge oder sogar unterschiedlicher Abtastraten vor, können die Operationen einer als Matrix dargestellten ZR-DB nicht durchgeführt werden und die einzelnen ZR müssen vorverarbeitet werden, bzw. es werden spezielle Methoden und Datenrepräsentationen zur Ermittlung von Ähnlichkeitsbeziehungen benötigt.

#### A.1.1.3 Skalierungsprobleme der klassischen Zeitreihenanalyse

Viele der klassischen Verfahren skalieren unzureichend mit der Größe der Zeitreihe(n) und eignen sich daher nur bedingt für Analysen besonders großer Datensätze. Dies liegt vor allem in der hohen Dimensionalität großer reellwertiger Zeitreihendaten begründet, die sich für multivariate Zeitreihen nochmals vergrößert. Aber auch die meist exponentielle Laufzeitkomplexität zum Einsatz kommender Algorithmen führt zu Problemen, sobald eine kritische Anzahl an Datenpunkten verarbeitet werden soll [176].

Einige Ansätze versuchen dieser Problematik so zu begegnen, dass eine Vorverarbeitung großer Zeitreihendaten im Sinne einer zeitlichen Vergrößerung vorgesehen wird, sodass die Datenmenge ausreichend reduziert wird, um die vorhandenen mathematischen Methoden der Zeitreihenanalyse auf klassischen Rechnerarchitekturen wieder anwenden zu können. Hierzu findet sich in der Literatur die Unterabtastung, am häufigsten jedoch die Aggregation durch Mittelwert- oder Medianbildung über eine feste Anzahl aufeinanderfolgender Messwerte, beispielsweise durch Zusammenfassung stündlicher Messwerte zu Tagesdurchschnittswerten. Dieses Vorgehen eignet sich für einige Anwendungen und wird häufig eingesetzt, obgleich in vielen Fällen ebenso von vornherein eine niedrigere Abtastrate gewählt werden könnte (vergl. z. B. [274]). Das Vorgehen hat den Vorteil, dass wohlbekanntere Verfahren und gebräuchliche Werkzeuge weiterverwendet werden können und keine neuen komplexen Verfahren, bzw. spezielle Architekturen und Software eingeführt und verstanden werden müssen.

Trotzdem wird ein solcher Einsatz klassischer Verfahren hier nicht näher untersucht, da zahlreiche Anwendungen, u. a. auch die Analyse der hochfrequenten Energiemessungen des KIT, eine

flexible zeitliche Auflösung voraussetzen, um Analysen sowohl auf hohen als auch niedrigeren Detailstufen zu ermöglichen. Die Aggregation der Messwerte über die Zeit zum Zwecke der Nutzbarmachung klassischer Analysemethoden ist v. a. deshalb hier inakzeptabel, da mit zunehmender Datengröße die Grenzen der Verfahren, bzw. deren Implementierung auf klassischen Rechnersystemen, erreicht werden und dann die Aggregate zeitlich weiter vergrößert werden müssen. Die Skalierungsproblematik wird damit also nicht gelöst, sondern nur soweit wie möglich umgangen – zu Lasten der Genauigkeit. Unter diesen Bedingungen wären die angestrebten Analysen hochfrequenter Messdaten niemals gleichzeitig für große Zeitbereiche und hohe Detailgrade möglich. Des Weiteren können aggregierte ZR streng genommen nicht mehr als kausal kontinuierlich betrachtet werden, weshalb einige typische Operationen, wie z. B. die Kompensation ungleich verteilter Messzeitpunkte durch Interpolation, nicht länger zulässig sind.

Die zeitliche Aggregation spielt jedoch eine wichtige Rolle bei Längenskalierungen und der Indizierung von Zeitreihen, die wiederum eine Grundlage für die Realisierung zahlreicher Analyseaufgaben darstellt.

### **A.1.2 Vorverarbeitung**

Viele Zeitreihen sind das Resultat von Messungen oder Abtastungen, die fehlerbehaftet sein können, weshalb auftretende Artefakte wie sensorbedingtes Rauschen und durch Messfehler bedingte Ausreißer in einem Vorverarbeitungsschritt kompensiert werden müssen. Für manche Analyseaufgaben, v. a. zur Ähnlichkeitsbewertung von ZR, müssen im Vorfeld zusätzliche Normierungen und Skalierungen durchgeführt werden. Die einzelnen Schritte weisen Abhängigkeiten auf, weshalb die Durchführungsreihenfolge Einfluss auf die Ergebniszeitreihe haben kann. Einige Zeitreihenrepräsentationen enthalten implizit bereits Vorverarbeitungsschritte, weshalb in diesen Fällen oft keine zusätzliche Vorverarbeitung notwendig ist und sogar unerwünschte Effekte haben kann.

Die wichtigsten Vorverarbeitungsschritte, auf die im Folgenden näher eingegangen wird, sind:

- Kompensation von Rauschen
- Zeitliche Skalierung und Interpolation
- Filterung fehlerbedingter Ausreißer und
- Normalisierung des Wertebereichs.

#### *A.1.2.1 Kompensation von Rauschen*

Zur Aufarbeitung verrauschter Signale können Filter eingesetzt werden, die das Signal auf ein für die durchzuführende Aufgabe akzeptables, verkleinertes Frequenzspektrum beschränken. Meist werden hier Tiefpassfilter eingesetzt, um hochfrequentes Rauschen, etwa durch rauschbehaftete Sensoren, und Ausreißer zu entfernen. Eine weitere Möglichkeit zur Kompensation von Rauschen

und Ausreißern stellt die zeitliche Aggregation dar, etwa durch Mittelwert- oder Medianbildung über zusammenhängende, meist gleich lange Zeitbereiche. Als Verfahren, das in der Zeitreihenanalyse häufig eingesetzt wird, ist hier v. a. der *Gleitende Mittelwert* (engl. *Moving Average*) zu nennen, bei welchem einem Datenpunkt der Mittelwert über ein gleitendes Fenster – eine aufeinanderfolgende Sequenz von Werten – zugewiesen wird. Je nach relativer Lage des Datenpunkts zum Fenster und der Gewichtung der Einzelwerte spricht man von *zentrierten* oder *nicht zentrierten*, *gewichteten* oder *ungewichteten*, bzw. *linear* oder *exponentiell geglätteten Mittelwerten*.

#### A.1.2.2 Zeitliche Skalierung und Interpolation

Zeitliche Aggregation kann auch zur Längenskalierung von Zeitreihen eingesetzt werden, was beispielsweise Ähnlichkeitsvergleiche unterschiedlich langer ZR ermöglicht, indem die längere ZR in eine Anzahl Aggregate überführt wird, die der Länge der gegenüberzustellenden ZR entspricht. Andererseits können durch Interpolation auch zusätzliche Datenpunkte erzeugt werden, um eine Zeitreihe künstlich zu verlängern, was jedoch je nach Art der Daten zu ungültigen Ergebnissen führen kann. Eine Interpolation kann auch dann sinnvoll sein, wenn nicht äquidistante Messdaten mit äquidistanten ZR verglichen werden sollen. Hierzu muss jedoch stets genau geprüft werden, inwieweit die zugrundeliegenden Daten interpolierbar sind und welche neuen Artefakte dabei entstehen können. Insbesondere kann es hierdurch unter bestimmten Bedingungen zur Filterung hochfrequenter Signalkomponenten kommen, nämlich dann, wenn die Signalfrequenz sich der Nyquist-Frequenz, also der halben Abtastfrequenz, annähert. In eine solche Interpolation sollten, falls vorhanden, auch die Zeitstempel der Datenpunkte zur Gewichtung miteinbezogen werden.

#### A.1.2.3 Filterung fehlerbedingter Ausreißer

Messmethoden und Sensoren können fehlerbehaftet sein und somit Zeitreihen hervorbringen, die fehlerbedingte Ausreißer enthalten. Die nachträgliche Identifikation dieser Ausreißer in den Daten ist jedoch ziemlich schwierig. Eine weit verbreitete Methode ist es, den Bereich als „Erwartungsbereich“ anzusehen, in dem die Mehrheit der Messwerte liegt. Werte weit außerhalb dieses Erwartungsbereichs werden dann als Ausreißer betrachtet. Dieses Vorgehen ist jedoch insofern problematisch, da der Erwartungsbereich willkürlich definiert werden muss. Z. B. werden für die verbreiteten *Box Plots* (erstmal in [102] beschrieben) Stichprobenwerte innerhalb des Intervalls des Quartilabstands<sup>46</sup> ( $Q_{75} - Q_{25}$ ) als Erwartungswerte und solche, die mehr als das 1,5-fache des

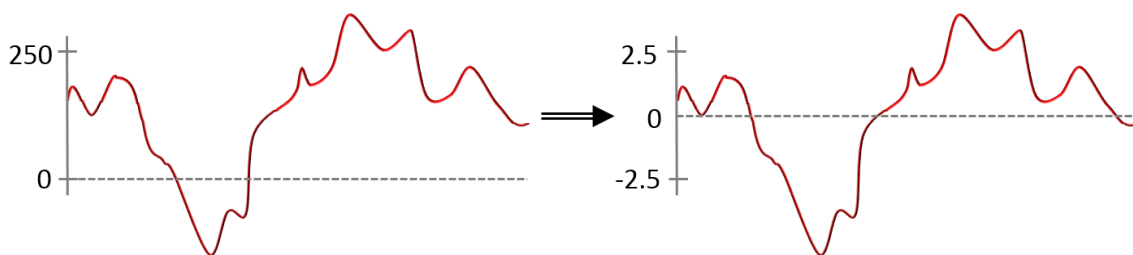
---

<sup>46</sup> Quartile sind spezielle Quantile, also Schwellwerte, welche das Intervall aufgetretener Messwerte in vier Bereiche aufteilen, so dass in jedem Bereich ein Viertel der Messwerte enthalten sind. 25% aller aufgetretenen Werte sind also kleiner als das 25%-Quartil (kurz  $Q_{0,25}$ ), 50% sind kleiner als  $Q_{0,50}$  usw. Als Interquartilbereich (in der Literatur häufig fälschlicherweise als Interquartilabstand) bezeichnet man das Intervall zwischen  $Q_{0,75}$  und  $Q_{0,25}$ , in dem die Hälfte aller Messwerte liegen. Der Interquartilabstand dient hingegen als ein Maß für die Streuung einer Messung.

Quartilabstands außerhalb dieses Intervalls liegen, als Ausreißer gedeutet. Auch für Verteilungen fern einer Normalverteilung – etwa multimodale Verteilungen – ist das Vorgehen problematisch. Für die nachträgliche Unterscheidung zwischen fehlerbedingten Ausreißern und originär auftretenden Extremwerten ohne Vorwissen ist bisher kein Verfahren bekannt. Liegt jedoch ein Modell vor, das Auskunft über die Auftrittswahrscheinlichkeit von Ausreißern gibt, so kann diese Information genutzt werden, wie z. B. im Ausreißertest nach Walsh [313]. Aufgrund der beschriebenen Problematik ist es sinnvoll, das Messverfahren sorgfältig dahingehend zu optimieren, möglichst wenige bzw. keine fehlerbedingten Ausreißer zu erzeugen.

#### A.1.2.4 Normalisierung des Wertebereichs

Einen weiteren Vorverarbeitungsschritt, der Voraussetzung vieler Analyseaufgaben, insbesondere bei Ähnlichkeitsvergleichen, aber auch bei der Visualisierung von ZR ist, stellt die Normalisierung bzw. statistische Standardisierung des Wertebereichs dar. Eine häufig verwendete aber nur für bestimmte Aufgaben geeignete Normalisierung ist die sog. *Minimum-Maximum Skalierung*, bei welcher der Wertebereich zwischen Minimum und Maximum innerhalb einer konkreten Zeitreihe auf einen Zielbereich – z. B. das Intervall  $[0,1]$  reeller Zahlen – abgebildet wird. Sie kann beispielsweise nützlich sein, wenn Zeitreihen mit unterschiedlichen Wertebereichen visualisiert werden sollen, da ansonsten eine ZR mit vergleichsweise kleinerem Wertebereich (z. B.  $[-1, 1]$ ) auf derselben Skala mit einer ZR mit größerem Wertebereich (z. B.  $[0, 5000]$ ) visuell lediglich als flache Linie dargestellt würde. Nicht zuvor entfernte Ausreißer können diese Skalierung jedoch stark verfälschen. Beispielsweise ändert sich der Maximalwert für die Skalierung enorm, wenn in einer ZR, deren Werte beispielsweise ansonsten im relativ kleinen Intervall  $[0, 1]$  enthalten sind, ein Messfehler zu einem Wert von 100 führt. Das neu skalierte Intervall wäre dann  $[0, 100]$ , während alle außer einem einzigen Wert im Intervall  $[0, 1]$  liegen. Die Visualisierung der skalierten ZR würde wiederum eine flache Linie ergeben. Auch der Mittelwert der Zeitreihe läge nicht mehr in der Mitte des neuen Intervalls, was die Minimum-Maximum Skalierung ungeeignet für einige Aufgaben macht.



**Abbildung A.1:** Darstellung der Z-Normalisierung (von engl. *zero mean and unit energy*) anhand eines Beispiels.

Eine weitere Normalisierung bzw. Standardisierung, die dieses Problem umgeht und sich für zahlreiche Analyseaufgaben – insbesondere Zeitreihenvergleiche – bewährt hat, stellt die sog. *Z-Normalisierung*<sup>47</sup> dar (z. B. in [145], eingeführt in [92] als *zero mean and unit energy*). Dabei werden ZR so skaliert, dass sich ein Mittelwert  $\mu$  von Null und eine Standardabweichung  $\sigma$  von Eins ergeben. Derartig skalierte Zeitreihen können dann alleine aufgrund ihres zeitlichen Verlaufs verglichen werden, ohne dass ihre Wertebereiche und Mittelwerte einen Einfluss auf die Distanz der Zeitreihen haben. Dabei werden automatisch Verschiebungen im Wertebereich (*Shift*) und unterschiedliche Skalierungen ausgeglichen. Differenzen zwischen zwei Zeitreihen können damit weitgehend unabhängig von der jeweiligen Lage im Werteraum angegeben werden. In Abbildung A.1 soll das Verfahren anhand eines Beispiels verdeutlicht werden.

### A.1.3 Aktuelle Analyseaufgaben

Zu den typischen Aufgaben der klassischen Zeitreihenanalyse gehören die Vorverarbeitung im Sinne einer Bereinigung von Rauschen und Ausreißern, eine kompakte Verlaufsbeschreibung, die Feststellung und Bereinigung von Trends und saisonalen Komponenten, die Detektion periodischen Verhaltens sowie Modellbildung und Prognose.

Für viele neuere wissenschaftliche Fragestellungen sollen jedoch zunehmend auch weitergehende Aufgaben durchgeführt werden, welche vor allem für die Analyse größerer Bestände an Zeitreihendaten bedeutsam sind. Basis vieler dieser Aufgaben ist der inhaltliche Vergleich von Zeitreihen  $X_i$  in einer Zeitreihendatenbank mit einer Anfrage  $Q$  durch eine Distanzfunktion  $D(Q, X_i)$  und Aussagen über deren Ähnlichkeit.

Diese Analyseaufgaben beinhalten unter anderem die

- inhaltsbasierte Anfrage (engl. query by content)
- Kategorisierung bzw. Klassifikation
- Gruppierung (engl. Clustering) sowie die
- *Indizierung* von Zeitreihen für schnelle *Suchen* und die
- semantische Untergliederung (Segmentierung)

---

<sup>47</sup> Der Begriff ist abgeleitet von engl. *zero mean, unit variance normalization*. In der deutschsprachigen Literatur wird auch oft von einer *Z-Transformation*, bzw. von *Statistischer Standardisierung* gesprochen. Streng genommen darf jedoch nur dann von *Standardisierung* gesprochen werden, wenn der wahre *Erwartungswert* als Null-Wert und die wahre *Varianz* der zugrundeliegenden Zufallsvariablen verwendet werden. Da aber meist das arithmetische Mittel und die Wurzel der Standardabweichung gegebener Stichproben Verwendung finden, sollte eigentlich korrekterweise von einer sog. *Studentisierung* [279] gesprochen werden, worauf in dieser Arbeit aus Gründen der besseren Verständlichkeit verzichtet wird.

- Detektion häufiger Muster (engl. Motifs)
- Detektion seltener Muster (engl. Discords)
- Entdeckung von Regeln
- Interaktive Visuelle Exploration.

Die genannten Analyseaufgaben haben die Gemeinsamkeit, dass bei ihrer Anwendung auf reellwertige ZR alle relevanten Daten für jede Anfrage aufs Neue ausgewertet werden müssen, weshalb zur Beschleunigung meistens vorberechnete Indexstrukturen bzw. approximative Verfahren zum Einsatz kommen. Außerdem hat die Definition des verwendeten Ähnlichkeitsmaßes einen entscheidenden Einfluss auf die Umsetzung jener Aufgaben.

Für eine Motivation und detaillierte Betrachtung der meisten genannten Analyseaufgaben sei auf [79] verwiesen, wo auch auf die Bedeutung von Distanzmaßen eingegangen wird. Auf die einzelnen Aufgaben wird in den folgenden Abschnitten auf der Grundlage von [79], [86], [103], [139] und [244] genauer eingegangen.

#### A.1.3.1 *Inhaltsbasierte Anfragen (query by content)*

Bei *inhaltsbasierten Anfragen* sollen zu einem vorgegebenen Datensatz alle anderen Datensätze innerhalb der zu durchsuchenden Datenbasis ermittelt werden, die der Vorgabe ähnlich sind. Für Zeitreihen kann diese Aufgabe als Teilsequenz-Vergleich (engl. Subsequence Matching) aufgefasst werden. Teilsequenz-Vergleiche stellen bereits seit zwei Jahrzehnten einen Forschungsschwerpunkt im Bereich der Zeitreihenanalyse dar (z. B. [120], [81], [149]).

Es können dabei je nach Datenbasis, Vorgabe- und Ergebnisdaten fünf Fälle unterschieden werden, die in Tabelle A.1 aufgeführt sind. Fall *a*) gleicht eine Datenbasis mehrerer vollständiger Zeitreihendatensätze mit einem Vorgabedatensatz ab. Dieser stellt bei b), c) und d) lediglich eine Teilsequenz, ein sog. Suchmuster dar. Fall *a*) kann als vollständiger Sequenzabgleich (gemäß engl. Whole Series Matching [177]) bezeichnet werden. Die Fälle b), c), d) und e) stellen hingegen einen Teilsequenzabgleich (engl. Subsequence Matching [177]) dar, wobei dieser Vorgang für b) und c) für alle zu durchsuchenden Datensätze wiederholt wird; in d) und e) wird in einem einzigen Datensatz gesucht.

Teilsequenzabgleiche können beispielsweise durch einen sog. *Fensterverschiebungs-* (engl. *Sliding Window-*) Algorithmus realisiert werden, bei welchem die Anfragezeitreihe über der größeren Basiszeitreihe um je einen Zeitschritt verschoben wird und dort mit der lokalen Teilsequenz des Fensterinhalts der Basiszeitreihe abgeglichen wird.

Gemäß [177] wurden bereits unzählige Methoden für vollständige Sequenzabgleiche vorgestellt (z. B. die in [152] getestet). Da hierfür jedoch Vorwissen über die Datenbasis vorliegen muss (z. B. die Längen der Zeitreihen, bzw. Muster) sind die Anwendungsmöglichkeiten stark begrenzt.

Auch arbeiten viele Ansätze nur mit Listen von Zeitreihendatensätzen gleicher Länge als Datenbasis, was in realen komplexen Analyseszenarien eher selten der Fall sein dürfte. Teilsequenzabgleiche können zu vollständigen Sequenzabgleichen generalisiert werden, indem die Basiszeitreihe in nichtüberlappende Segmente unterteilt wird (siehe die Beschreibung von Chunking in Abs. A.1.3.4), welche dieselbe Länge aufweisen wie die gesuchte Sequenz, was in den erwähnten Fensterverschiebungs-Algorithmen zum Einsatz kommt.

**Tabelle A.1:** Fälle von Zeitreihen-Ähnlichkeitsanfragen.

	<i>Vorgabedaten</i>	<i>Datenbasis</i>	<i>Ergebnisdaten</i>
a)	<b>kompletter Zeitreihendatensatz</b>	<b>Liste</b> kompletter Zeitreihendaten- sätze	<b>Liste</b> ähnlicher Zeitreihendaten- sätze
b)	<b>Teilsequenz</b>	<b>Liste</b> kompletter Zeitreihendaten- sätze	<b>Liste</b> von Zeitreihen mit ähnlichen Teilsequenzen und jeweils <b>erste</b> Fundstelle
c)	<b>Teilsequenz</b>	<b>Liste</b> kompletter Zeitreihendaten- sätze	<b>Liste</b> von Zeitreihen mit ähnlichen Teilsequenzen und jeweils <b>alle</b> Fundstellen
d)	<b>Teilsequenz</b>	<b>einzelner</b> großer Zeitreihendatensatz ZR	<b>erste</b> Fundstelle einer ähnli- chen Teilsequenz in ZR
e)	<b>Teilsequenz</b>	<b>einzelner</b> großer Zeitreihendatensatz ZR	<b>alle</b> Fundstellen ähnlicher Teilsequenzen in ZR

Ein Distanzmaß für Zeitreihen ermöglicht in Kombination mit einer dimensionsreduzierenden Zeitreihenrepräsentation auch die effiziente Realisierung einer *inhaltsbasierten Ähnlichkeitssuche*, also einer unscharfen Suche, die in vielen Anwendungsbereichen benötigt wird. Eine solche kann bei besonders großen Datenbanken reellwertiger Zeitreihen sehr viel mehr Zeit und Ressourcen in Anspruch nehmen, da hierzu Differenzen für alle Datenpunkte aller Zeitreihen berechnet werden müssen, so dass alle Datensätze mindestens einmal sequentiell durchlaufen werden müssen (engl. *Sequential Scanning*). Methoden zur Zeitreihenindizierung bieten hierfür elegante Lösungen an, wodurch schnelle Suchen ermöglicht werden sollen. Dabei stellt sich auch die Frage, wie Zeitreihen verglichen werden können, die unterschiedliche Wertebereiche und Längen aufweisen.

Da diese und ähnliche Aufgaben häufig durchgeführt werden müssen und kurze Reaktionszeiten erwartet werden, ist ein möglicher Ansatz, in dimensionsreduzierten, also komprimierten, Abbildern der Zeitreihen zu suchen, bzw. diese zu indizieren, wie in Abbildung A.2 dargestellt. Durch die approximative Natur derartiger Zeitreihenrepräsentationen verwandelt sich die Aufgabe in eine vergleichsweise triviale Mustererkennung. Gesucht wird dann in einer verhältnismäßig kleinen Datenbank, welche auf die Merkmale reduziert sein sollte, die für die Unterscheidung der Zeitreihen relevant sind. Für eine Such-Anwendung auf einem Client-Rechner kann die Index-Datenbank dann oft soweit reduziert werden, dass sie im schnellen lokalen Hauptspeicher gehalten werden und der Zugriff auf langsamere, entfernte Datenquellen vermieden werden kann.

Bei der *inhaltsbasierten Suchanfrage* soll eine geordnete Liste  $L$  aus einer Grundmenge von Zeitreihen  $DB$  ermittelt werden, welche der Anfrage-Zeitreihe  $Q = (q_1, \dots, q_n)$  am ähnlichsten ist, also laut [79]:

$$\forall X_k, X_j \in L, k > j \stackrel{!}{\Leftrightarrow} D(Q, X_k) > D(Q, X_j) \quad (\text{A.3})$$

Zusätzlich können je nach Realisierung auch die erste oder alle Fundstellen ermittelt werden. Es können mehrere Vorgehensweisen unterschieden werden: Eine *abstands-basierte Suche* (engl. *epsilon-range query*) führt einen Maximalabstand  $\epsilon$  ein und lässt nur Ergebnisse zu, für welche  $\epsilon$  unterschritten wird.

Laut [79] kann die abstands-basierte Suche formal definiert werden als

$$L_\epsilon = \{X_i \in DB \mid D(Q, X_i) \leq \epsilon\} \quad (\text{A.4})$$

Da jedoch die Wahl des Parameters  $\epsilon$  stark von Art und Ausprägung der Eingangsdaten abhängt, ist es für Benutzer leichter, stattdessen die Anzahl der erwarteten Ergebnisse als einen Parameter  $K$  anzugeben. Man spricht in diesem Fall von einer *K-Nächste-Nachbarn* (KNN) Anfrage (engl. *K-Nearest-Neighbors*).

Gesucht ist wiederum eine geordnete Liste

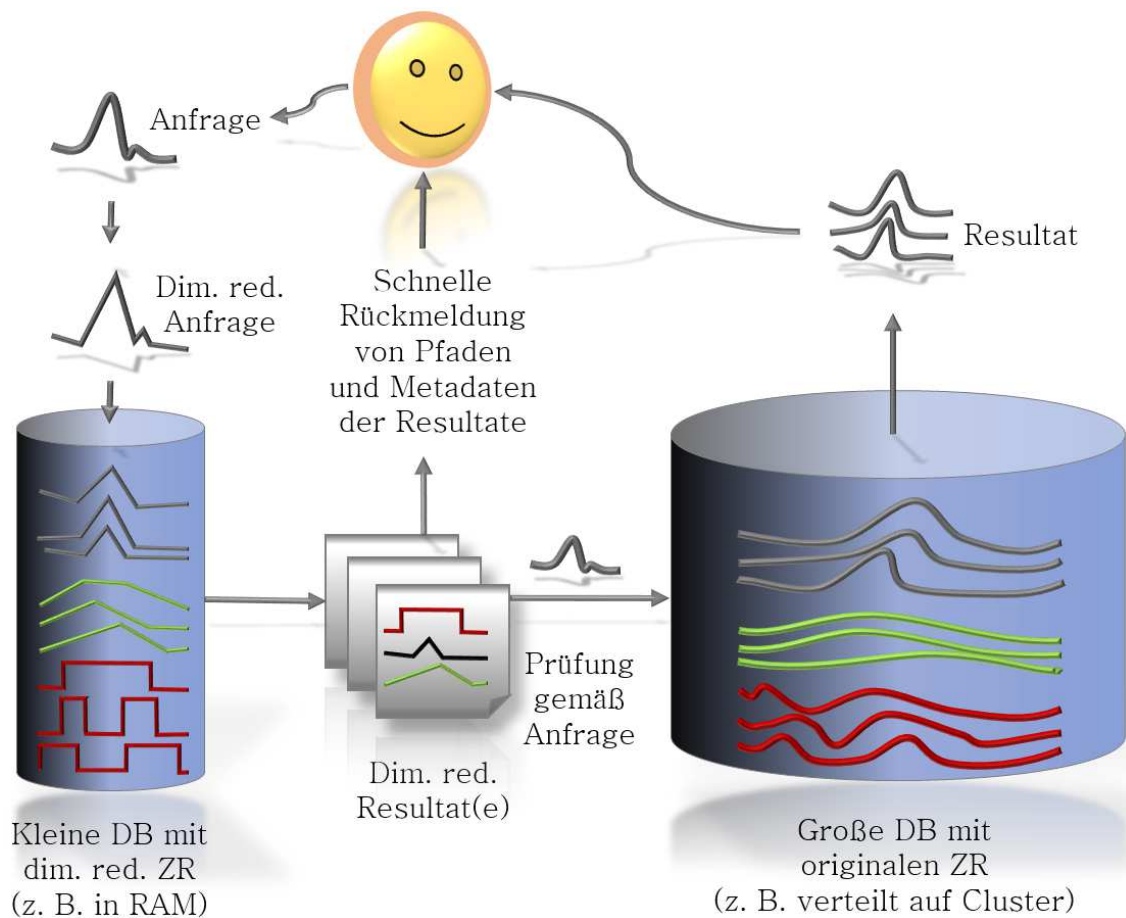
$$L_{KNN} = \{X_i \mid X_i \in DB\},$$

so dass (A.5)

$$|L_{KNN}| = K \text{ und } \forall X_j \notin L_{KNN} : D(Q, X_i) \leq D(Q, X_j).$$

Ein möglicher Ablauf einer solchen Suche ist in Abbildung A.2 schematisch dargestellt. Man beachte dabei, dass die Rückmeldung der Ergebnisse an den Benutzer zweistufig ist: Eine schnelle Rückmeldung der approximativen Ergebnisse ermöglicht z. B. interaktive Echtzeitanwendungen und eine langsamere Rückmeldung der exakten Prüfung der Original-Zeitreihe.





**Abbildung A.2:** Ablauf unscharfer Ähnlichkeitssuchen in Zeitreihen.  
Quelle: Eigene Darstellung, in Anlehnung an [244].

### A.1.3.2 Klassifikation

Bei der *Klassifikation* handelt es sich hingegen um eine Einordnung in vorgegebene Klassen  $C = \{c_i\}$  aufgrund von Merkmalsausprägungen, wie z. B. bei der Zuordnung einer Pflanze zu einer Gattung aufgrund von Blatt- oder Blütenformen. Meist sind bei dieser Aufgabe viele Beispiele für jede Klasse – sog. Trainingsdaten  $T_i$  – im Voraus bekannt und gegeben. Nach einer Trainingsphase sollen dann neue Zeitreihen  $X_j$  mit unbekannter Klassenzugehörigkeit effizient mit einem Etikett (engl. *Label*) für die zugeordnete Klasse  $c_k$  versehen werden. Prinzipiell kann unterschieden werden zwischen einer Klassifikation vollständiger Zeitreihen und einer Klassifikation von Teilsequenzen.

Ein typisches Problem bei der Klassifizierung ist das sog. *Über-* bzw. *Untertrainieren*, im englischen Sprachraum meist *over-*, bzw. *under-fitting* genannt, ein zu genaues bzw. zu ungenaues Mo-

dell. Hierzu kann es leicht kommen, wenn die Trainingsdatensätze nicht repräsentativ für die gesamte Klasse sind. Dies ist während der Trainingsphase schwer zu beurteilen, da zukünftig einzuordnende Daten stark von den Trainingsdaten abweichen könnten. Außerdem ist es meist nach der Trainingsphase nicht mehr möglich, neue Klassen einzufügen, zu entfernen oder abzuändern.

#### A.1.3.3 Clustering - Automatische Gruppierung

Im Gegensatz zur Klassifizierung stehen beim Clustering die möglichen Klassen nicht im Voraus fest. Es sollen vielmehr automatisch „natürliche“ Häufungen (engl. *Clusters*) im Sinne von Gemeinsamkeiten und Ähnlichkeiten in der gegebenen Datenmenge gefunden werden, welche dann wie Klassen behandelt werden. Diese sollen in sich möglichst homogen sein und sich untereinander möglichst signifikant unterscheiden. Deshalb kann Clustering als Optimierungsaufgabe betrachtet werden, bei der die Varianz innerhalb von Clustern minimiert und zwischen Clustern maximiert wird. Von entscheidender Bedeutung ist dabei natürlich das Distanzmaß und die Auswahl geeigneter Merkmale, anhand derer Datensätze unterschieden werden.

Es können sowohl vollständige ZR als auch Teilsequenzen gruppiert werden<sup>48</sup>. Es besteht ebenfalls die Möglichkeit, vollständige ZR aufgrund der Eigenschaften ihrer Teilsequenzen zu gruppieren, z. B. bei einem Clustering nach der *längsten gemeinsamen Teilsequenz* [112].

Gemäß [103] können die zahlreichen verschiedenen Ansätze in fünf Kategorien eingeordnet werden: *Partitionierung, hierarchisches Clustering, dichtebasiertes Clustering, Netzpartitionierung und modellbasiertes Clustering*. Problematisch ist die ungewisse und nicht spezifizierbare Anzahl resultierender Cluster. Deshalb wird häufig auf ein hierarchisches Clustering zurückgegriffen, da hierdurch alle möglichen Aufteilungen ermittelt werden, so dass nachträglich eine Aufteilung auf geeignetem Verzweigungsniveau ausgewählt werden kann. Hierzu – und außerdem zur besseren Interpretierbarkeit – empfiehlt sich die Darstellung der hierarchischen Cluster als sog. *Dendrogramm*, also als Baumstruktur.

#### A.1.3.4 Semantische Segmentierung und Detektion von Zustandsübergängen

Häufig soll eine Zeitreihe in mehrere Abschnitte mit inhärent konsistenter Charakteristik unterteilt werden. Das Ziel ist die kompakte Beschreibung des zeitlichen Verlaufs und die Detektion von Änderung. Die Segmentierung einer Zeitreihe kann als ein Spezialfall des Clusterings nicht überlappender Teilsequenzen betrachtet werden. Gemäß [79] können rekursive Top-Down-Ansätze von iterativen Bottom-Up-Ansätzen unterschieden werden. Manche Zeitreihenrepräsentationen beinhalten eine implizite Segmentierung, so z. B. die *Abschnittsweise Lineare Approxima-*

---

<sup>48</sup> Ein Clustering von Teilsequenzen kann unter bestimmten Umständen bedeutungslos sein [153],[58], weshalb die Eignung von Daten und Methoden hierfür genau geprüft werden muss.

tion (engl. *piecewise linear approximation*, kurz PLA), welche eine Zeitreihe als Sequenz aneinandergesetzter Geradenabschnitte unterschiedlicher Länge approximiert und damit bereits sog. *Wendepunkte*<sup>49</sup> liefert [147]. Solche Wendepunkte sollten stets eine Änderung der Charakteristika einer Zeitreihe einleiten, um einen semantischen Bezug zu erhalten.

Andere Ansätze definieren Grenzbereiche, die angeben, wie weit sich eine Merkmalsausprägung innerhalb einer gewissen Zeitspanne ändern darf. Bei Überschreitung wird dann an dieser Stelle ein Wendepunkt definiert. Geeignete Merkmale sind beispielsweise Durchschnitt, Korrelation, Varianz, Frequenzspektrum [1], [151], Wavelets [79] oder fraktale Dimension [13], [23], [251]. Problematisch ist hierbei, dass es nicht trivial ist, automatisch einen geeigneten Detailgrad der Segmentierung auszumachen, so dass Rauschen und kleine lokale Änderungen zugunsten einer besseren Beschreibung des Gesamtverlaufs ignoriert werden können, ohne dabei zu grob vorzugehen und Änderungspunkte zu übersehen.

Eine spezielle Form der Segmentierung wird in [177] als sog. *Chunking* vorgestellt. Hierbei wird eine Zeitreihe entweder aufgrund einer fixen Periode oder aufgrund von Formaspekten in nicht-überlappende Segmente zerteilt. Der Ansatz ist beispielsweise hilfreich, um aus einer ECG-ZR einzelne Herzschläge zu extrahieren o. ä. [147] gibt einen Überblick über zahlreiche Segmentierungsansätze, welche getestet und verglichen werden, und stellt ein interessantes Online-Verfahren vor, das linear mit der Datenmenge skaliert und qualitativ hochwertige Approximationen erzeugt.

#### A.1.3.5 Kompakte Zusammenfassung von Zeitreihen

Für die Beurteilung besonders großer Zeitreihen ist es häufig hilfreich, deren charakteristische Merkmale in kompakter Form zusammenzufassen. Es besteht dabei ein direkter Zusammenhang zur semantischen Segmentierung aus A.1.3.4, da die Aufteilung der ZR in konsekutive Segmente direkt eine Zusammenfassung ermöglicht, indem signifikante Merkmale der Segmente extrahiert werden. Auch kann eine geeignete dimensionsreduzierte Repräsentation der ZR (siehe Abs. A.1.5) eine Zusammenfassung liefern. Diese beiden Vorgehensweisen der Zusammenfassung haben den Vorteil, dass sie abstrakte Anfragen auf Basis von Eigenschaften der zusammengefassten ZR ermöglichen, wie z. B. „finde alle ZR, deren Mittelwert größer ist als drei und die genau zwei Gipfel aufweisen“. Die Beschreibung der Zusammenfassung kann hierfür etwa in strukturierter textueller Form, z. B. in Form deskriptiver XML-basierter Metadaten erfolgen. Derartige Beschreibungen sind jedoch für menschliche Betrachter oft nicht leicht verständlich. Des-

---

<sup>49</sup> Der Begriff Wendepunkt bezeichnet im Sinne der Kurvendiskussion eine Krümmungsänderung, auch Bogenwechsel genannt. Hier wird der Begriff aber in einem erweiterten Sinn gebraucht, nämlich als derjenige Zeitpunkt, an dem eine Kurve ihr bisheriges charakteristisches Verhalten ändert.

halb werden auch häufig zusammenfassende visuelle Darstellungen einer Zeitreihe erzeugt, welche einem Betrachter einen schnellen Überblick über große Zeitreihen verschaffen sollen. Hier besteht die hauptsächliche Herausforderung darin, eine möglichst detailreiche Darstellungsform zu finden, die auf beschränktem Darstellungsraum (etwa in einem Bericht auf DIN-A4 Papier oder auf einem Bildschirm) Platz findet.

In allen Fällen soll für eine gegebene Zeitreihe  $X$  mit  $n$  Datenpunkten eine Approximation  $A(X)$  gefunden werden, die kleiner als der Vektor der Datenpunkte ist und die wesentlichen Merkmale wie Struktur und Form der ZR enthält (vergl. [303]).

#### A.1.3.6 Indizierung von Zeitreihen

Eine weitere wichtige Analyseaufgabe, die teilweise eng mit den bereits vorgestellten Aufgaben zusammenhängt, ist die Indizierung von Zeitreihen für schnelle Zugriffs- und Suchverfahren in großen ZR-DB (vergl. [139], [244], [86] und [79]). Sie wird insbesondere für die in A.1.3.1 beschriebenen inhaltsbasierten Anfragen benötigt. Das Ziel dabei ist es, eine sequentielle Suche über alle in Frage kommenden ZR zu vermeiden, etwa durch eine Aufteilung des Suchraums, so dass dieser während der Suche zunehmend verkleinert werden kann, um schnelle Anfragen zu ermöglichen. Nahezu alle bekannten Verfahren bilden hierzu einen Index über eine dimensionsreduzierte Repräsentation der Zeitreihen. Der erste bekannte Ansatz wurde 1993 von Agrawal u. a. vorgestellt [1]. 1994 stellten Faloutsos u. a. [81] ihr oft eingesetztes GEMINI-Framework vor. Gemäß [81] soll eine Indizierungsmethode

- schneller sein als eine sequentielle Suche
- einen geringen zusätzlichen Speicherbedarf aufweisen
- Anfragen unterschiedlicher Länge erlauben
- keine Neuberechnung des Index nach Änderungen in der TS-DB erfordern
- korrekte Ergebnisse liefern, in denen keine ähnlichen Zeitreihen fälschlicherweise ausgelassen werden (engl. *false dismissals* oder *false negatives*)

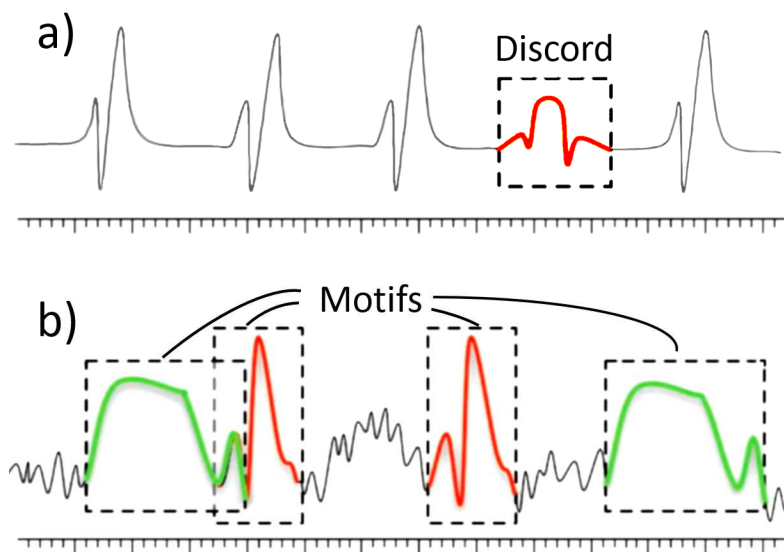
„Fälschlicherweise“ als ähnlich eingestufte ZR sind als Ergebnis zwar zulässig, ihre Differenz zum gesuchten Muster und ihre Anzahl sollte jedoch möglichst gering sein (vergl. [170]). In [146] nennen Keogh zwei weitere wünschenswerte Eigenschaften einer ZR-Indizierung:

- Die Laufzeit des Indizierungsvorgangs sollte hinreichend kurz sein
- Der Index sollte unterschiedliche Distanzmaße unterstützen

In Hinblick auf eine gute Skalierbarkeit der Indizierungsmethode sollte sie außerdem auch mit großen Datenmengen zurechtkommen und auch verteilt gespeicherte Datensätze indizieren können.

## A.1.3.7 Detektion seltener Muster – Discords

Das Auffinden seltener Muster innerhalb einer Zeitreihe ist eine weitere wichtige Aufgabe, die in vielen Anwendungsbereichen, v. a. in Bio- und Medizinischer Informatik, zunehmend an Bedeutung gewinnt. Abbildung A.3 a) zeigt ein solches ungewöhnliches Muster innerhalb einer Zeitreihe mit ansonsten periodischer Wiederholung eines charakteristischen Musters. Man denke als Beispiel etwa an die massiven Datenmengen, die im Gesundheitsbereich durch Biomonitoring aufgenommen werden. Bislang werden diese Zeitreihendaten von Fachpersonal hauptsächlich manuell überprüft, automatisch kann meist lediglich Alarm ausgelöst werden, sobald Werte außerhalb eines definierten Normbereichs liegen. Letzteres stellt jedoch keine Form der Detektion ungewöhnlicher Muster, sondern der vergleichsweise trivialen Detektion von Ausreißern dar.



**Abbildung A.3:** a) Idealisiertes ungewöhnliches Muster (*Discord*) in periodischer Zeitreihe und b) wiederkehrende Muster (*Motifs*). Identisch zu Abbildung 2.5.  
Quelle: [79], modifiziert.

Nun möchte man jedoch weitreichendere Analysen ermöglichen, wie etwa „ungewöhnliche“ strukturelle Muster in den Messungen zu detektieren und – auf das obige Beispiel bezogen – in Patientendaten, wie etwa EKG-Messungen, ungewöhnliche Herzschläge erkennen. Wird diese Aufgabe von einem Fachmann durchgeführt, fließt all seine Erfahrung und sein Wissen in Form eines krankheits- bzw. patientenspezifischen Modells in die Analyse ein. Bei einer automatischen Analyse, die notwendig wird, sobald die Anzahl und Länge auszuwertender Zeitreihen zunimmt oder keine Referenzwerte vorliegen, sind solche Modelle jedoch in der Praxis oft nicht gegeben.

Aufbauend auf [79] kann die Aufgabe formal folgendermaßen definiert werden:

*Finde für eine gegebene Zeitreihe  $Z$  und ein Modell  $M$  ihres Normalverhaltens alle  
Teilabschnitte  $Z'$ , die nicht zu  $M$  passen sowie deren Positionen in  $Z$ .*

Eine besondere Schwierigkeit ist der Umgang mit überlappenden Teilsequenzen, da man eindeutige Fundstellen eines jeden Discords als Resultate erhalten möchte. Außerdem dürfen kleine lokale Unterschiede sowie Rauschen und Ausreißer nicht überbewertet werden. Es besteht ferner ein Zusammenhang zur modellbasierten *Vorhersage von Zeitreihen*, da die Abweichung von vorhergesagten Werten als Maß für deren Ungewöhnlichkeit betrachtet werden kann [79]. Steht jedoch kein Modell bereit, kann die Aufgabe reduziert werden zu:

*Finde für alle Teilsequenzen  $Z'$  der Länge  $l$  einer Zeitreihe  $Z$  der Länge  $n$  mit  $n \geq l$  diejenigen  $m$  Teilsequenzen  $Z'_d$  und deren Positionen in  $Z$ , die sich am meisten von allen anderen Teilsequenzen unterscheiden.*

In [143] wird hierfür der Begriff *Discords* eingeführt. Für die Berechnung von Discords wird hier lediglich die Teilsequenzlänge  $l$  als Parameter benötigt. Aus der resultierenden nach Unterschiedlichkeit geordneten Liste der Teilsequenzen kann dann die gewünschte Anzahl  $m$  entnommen werden. Darauf aufbauend wurde in [338] ein exakter Algorithmus zur Detektion von Discords fester Länge vorgestellt, welcher die Zeitreihe lediglich zweimal sequentiell durchlaufen muss und sich daher gut für große Datensätze eignet.

Ist jedoch wenig Vorwissen über die gesuchten ungewöhnlichen Muster vorhanden und es sollen Discords beliebiger Länge detektiert werden, steigt der Aufwand enorm, da für eine Zeitreihe der Länge  $n$  für alle möglichen Längen im Bereich  $[1..(n-1)]$  geprüft werden muss. In [257] wurde jüngst ein effizienter hierarchischer Ansatz vorgestellt, welcher basierend auf einer symbolischen Zeitreihendarstellung eine kontextfreie Grammatik einer Zeitreihe aufstellt und damit Discords beliebiger Länge detektiert.

#### A.1.3.8 Detektion häufiger Muster – Motifs

Eine der Detektion von Discords ähnliche Aufgabe ist das Auffinden häufig wiederkehrender Muster – sog. *Motifs* – innerhalb einer Zeitreihe. Diese Analyseaufgabe wurde durch Gensequenz-Analysen im Bereich der Bioinformatik motiviert, wo es u. a. darum geht, unbekannte Verwandtschaftsbeziehungen aufgrund von Gemeinsamkeiten im Erbgut aufzudecken. Der Vorgang kann wiederum aufbauend auf [79] formal definiert werden als:

*Finde für eine gegebene Zeitreihe  $Z$  alle Teilabschnitte  $Z'$ , die mehrmals in  $Z$  vorkommen und deren Positionen in  $Z$ .*

Ein Ansatz für die Suche nach Motifs wurde 2002 erstmalig in [180], [231] (derselben Autoren) als Suche nichtüberlappender „typischer“ Teilsequenzen beschrieben. Durch die Entdeckung der fehlenden Aussagekraft eines Clusterings reellwertiger Teilsequenzen 2003 durch Keogh u. a.

(spätere Publikation als Zeitschriftenartikel in [153]) erfuhr das *Motif-Mining* besondere Aufmerksamkeit, da die Autoren darauf hinwiesen, dass Motifs zu einem sinnvollen Clustering beitragen können [79].

Außerdem können Motifs gefunden werden, die das „Normalverhalten“ vieler Zeitreihen beschreiben, wodurch eine vereinfachte Modellierung und eine semantische Segmentierung ermöglicht werden kann. Die Aufgabe der Zeitreihenklassifikation kann zudem gemäß [341] beschleunigt werden, indem klassenspezifische Motifs ermittelt und verwendet werden [79].

#### A.1.3.9 Entdeckung von Regeln

Die Extraktion von Regeln ist eine Analyseaufgabe, der in den letzten Jahren zunehmend Beachtung geschenkt wurde. Hierbei geht es darum, Zustände zu identifizieren, in denen sich eine Zeitreihe in einzelnen Zeitabschnitten befindet und anschließend die Zustandsübergänge zu untersuchen. Um einem Zeitreihenabschnitt einen Zustand zuzuordnen zu können, ist eine zeitliche Aggregation notwendig, z. B. die Bildung von Mittelwerten, Standardabweichungen etc. Die Wahl der Aggregationsfunktion kann mehr oder weniger anwendungsabhängig sein und die Anzahl zusammengefasster Werte hängt von der Abtastrate der Zeitreihendaten und der Frequenz der Zeitreihe ab. Außerdem kann die Zeitreihe unterschiedlich segmentiert werden, etwa in gleich große Zeitabschnitte oder datenadaptiv, z. B. durch Ausnutzung detektierter Motifs und Discords. Auch der Wertebereich wird in diskrete Intervalle aufgeteilt, so dass jedem zeitlichen Aggregat eindeutig ein Intervall zugeordnet werden kann. Dies entspricht einer Symbolisierung der Zeitreihe wie z. B. bei der *Symbolischen Aggregat-Approximation* (kurz SAX, siehe Abs. A.1.8), welche eine Zeitreihe in eine (ordinale) Symbolsequenz transformiert.

In diesen Symbolsequenzen können dann grammatikalische Regeln ermittelt werden. Das u. a. untersuchen in [64] adaptive Methoden zur Entdeckung lokaler Muster in multivariaten Zeitreihen basierend auf einem Fensterverschiebungsalgorithmus und Teilsequenz-Clustering. Das Verfahren muss jedoch gemäß [178] als ungültig betrachtet werden, da das Clustering der Teilsequenzen keine semantisch bedeutsamen Resultate erzeugt. In [178] werden Korrekturen vorgeschlagen, durch welche viele vor [178] veröffentlichte Verfahren, die als fehlerhaft identifiziert wurden, modifiziert werden können. Ohsaki u. a. [221] schlagen ein Verfahren basierend auf Musterextraktion und Entscheidungsbäumen vor, welches auf SAX-codierten Zeitreihen basiert. In [249] stellen Salvador und Chan einen Segmentierungsalgorithmus namens *Gecko* vor, welche in Kombination mit dem Klassifikationsalgorithmus *RIPPER* dazu verwendet werden, einen Zustandsautomaten aufzubauen. Lin und Li übertragen in [182] das aus der Textanalyse bekannte *Bag-of-words* Verfahren auf die Zeitreihenanalyse (*Bag-of-patterns*), welches sich gemäß der Autoren besser für lange Zeitreihen eignet als *formbasierte* Ansätze. In [174] schlagen Li und Lin ein weiteres Verfahren vor, welches zwei Probleme zu lösen versucht, die sich bei der Regelextraktion

ergeben: Die meist nicht a priori bekannte Länge häufiger Muster und die Möglichkeit der Koexistenz von Mustern unterschiedlicher Länge. Das Verfahren basiert auf der SAX-Codierung von Zeitreihen und nutzt den rekursiven *Sequit*or-Algorithmus zur Extraktion einer *kontextfreien Grammatik* als hierarchische Struktur bei linearer Komplexität. Der Ansatz wird in [257] von Senin u. a. aufgegriffen und als interaktives Softwarewerkzeug umgesetzt.

#### A.1.3.10 Interaktive Visuelle Exploration

Eine der grundlegendsten und wichtigsten Aufgaben bei der Analyse von Zeitreihen ist die Visualisierung. Sie stellt eine kompakte Repräsentation von Zeitreihen bereit, welche dem menschlichen Betrachter einen schnellen Überblick über den charakteristischen zeitlichen Verlauf bietet. Der visuelle Vergleich mehrerer Kurvendarstellungen ermöglicht dem menschlichen Betrachter eine sofortige intuitive Ähnlichkeitsbewertung. Auch Ausreißer, Unregelmäßigkeiten, Periodizität und Muster werden auf eine intuitive Weise erkannt.

Es existieren unzählige Darstellungsoptionen für Zeitreihen, die alle ihre speziellen Vorzüge und Einschränkungen aufweisen, weshalb die Entscheidung für eine bestimmte Darstellung oft aufgaben- daten- oder anwendungsspezifisch ist. Am häufigsten kommt jedoch nach wie vor die verbreitete und tradierte Darstellung von Zeitreihen als Kurvendiagramm (engl. *Curve Plot*) zum Einsatz.

Gerade bei der Erstsichtung großer Sammlungen langer Zeitreihen ist es außerdem wichtig, schnell in den Daten navigieren zu können und schnell und unkompliziert zwischen verschiedenen Detailstufen zu wechseln. Daher haben sich v. a. interaktive Visualisierungen von Zeitreihen durchgesetzt, bei welchen Benutzer Darstellungsparameter unter sofortiger visueller Wirkung ändern können. Insbesondere gilt es, die bereits 1996 in [264] veröffentlichte Forderung von Ben Shneiderman zu erfüllen: „*Overview first, zoom and filter, then details-on-demand*“. Derartige Visualisierungen sind bisher jedoch nur beschränkt für große Datensätze geeignet. Die meistbenötigten interaktiven Aktionen wie das *Zoomen*, *Scrollen* und *Filtern* können effizient mit Hilfe von Computermaus und Tastatur durchgeführt werden. Da nur bei kurzen Reaktionszeiten eine effektive Exploration großer Zeitreihen möglich ist, stellen diese einen kritischen Faktor dar, besonders bei der ressourcenintensiven Darstellung langer Zeitreihen und großer Datenmengen.



#### A.1.4 Ähnlichkeitsbewertung von Zeitreihen

Sollen mehrere Zeitreihen miteinander verglichen werden, wird ein Maß für deren Ähnlichkeit bzw. Unterschiedlichkeit benötigt. Üblicherweise werden sog. *Distanzmaße*<sup>50</sup> für Sequenzen metrisch skalierteter Variablen – auf ZR bezogen also reellwertige ZR – verwendet, während man beim Vergleich von Sequenzen nominaler bzw. ordinaler Variablen eher von *Ähnlichkeitsmaßen* spricht. In diesem Abs. werden die bekannten Distanzmaße untersucht und beurteilt. Ein Beispiel eines Ähnlichkeitsmaßes für Sequenzen ordinalskalierteter bzw. symbolischer Variablen ist das Distanzmaß der *Symbolischen Aggregat-Approximation* (kurz SAX), namens *MINDIST*.

##### A.1.4.1 Menschliche Ähnlichkeitswahrnehmung und Robustheit von Distanzmaßen

Im Bereich des Data-Mining und des Maschinellen Lernens ist man bemüht, Distanzen zu berechnen, welche der Beurteilung der Unterschiedlichkeit durch einen Menschen entsprechen, der die visuellen Kurvendarstellungen von Zeitreihen vergleicht. Für einen menschlichen Betrachter ist es nämlich ohne weiteres möglich, die Ähnlichkeit zweier Kurven losgelöst von deren Wertebereichen, Skalierungen, Verschiebungen und Abtastraten, allein aufgrund *ihrer Form* bzw. *ihrer Struktur* zu beurteilen.

Hieraus lassen sich Anforderungen an die Robustheit eines optimalen Distanzmaßes ableiten, deren Bedeutung jedoch stets anwendungsabhängig gewichtet werden muss. Ein Distanzmaß soll robust sein gegenüber folgenden Beeinträchtigungen:

- im Wertebereich:
  - Skalierungen (engl. *Scaling*)
  - Verschiebungen (engl. *Shift*)
  - Verzerrungen (engl. *Warping*)
  - lokale Unregelmäßigkeiten (Rauschen, Ausreißer).
  
- im Zeitbereich:
  - Skalierungen (z. B. durch unterschiedliche Abtastraten)
  - Verschiebungen
  - Verzerrungen (engl. *Time Warping* oder *Dynamic Time Scaling*)

Die Problematik wird in Abbildung A.4 verdeutlicht.

---

<sup>50</sup> Die Begriffe Distanzmaß und Ähnlichkeitsmaß werden nachfolgend in einem laxen Sinne verwendet (wie in [170]) und als invers betrachtet, ohne notwendigerweise den Bedingungen eines metrischen Maßes genügen zu müssen.

Ein Distanzmaß sollte möglichst so flexibel sein, dass sowohl kleine als auch sehr große ZR sowie ZR unterschiedlicher Länge oder überlappende Bereiche zuverlässig verglichen werden können ohne z. B. kleine lokale Unterschiede über- oder unterzubewerten.

Des Weiteren ergeben sich beim Vergleich großer Mengen von Zeitreihendaten Anforderungen an die Skalierbarkeit und damit an die Komplexität der Distanzberechnung. Wünschenswert für eine optimale Skalierung mit großen Datenmengen ist eine linear von der Datengröße abhängige Komplexität ( $O(n)$ ).

Es können Distanzmaße, die direkt auf den Originaldaten operieren von solchen unterschieden werden, die auf abgeleiteten Repräsentationen (siehe Abs. A.1.5) operieren. Während erstere bei wesentlich höherer Laufzeit zu exakten Ergebnissen führen, eignen sich die auf dimensionsreduzierten Zeitreihenrepräsentationen definierten Distanzen gut für effiziente, unscharfe Ähnlichkeitsvergleiche. Die Genauigkeit des Abstandsmaßes sollte dabei bestimmbar sein. Berechnete Distanzen im Merkmalsraum sollen möglichst gleich oder kleiner als die „tatsächlichen Distanzen“<sup>51</sup> sein (siehe „*Bounding Lemma*“ in [81]), um zu vermeiden, dass eine ähnliche ZR beim Vergleich fälschlicherweise nicht als ähnlich erkannt wird. Die Wahl eines Ähnlichkeits- bzw. Distanz- oder Pseudodistanzmaßes ist meist anwendungsabhängig und steht in engem Zusammenhang mit der Wahl einer Zeitreihenrepräsentation, die in Abs. A.1.6 beschrieben wird.

#### A.1.4.2 Metrische Distanzmaße

Es existieren metrische und nichtmetrische Distanzmaße, sog. *Pseudodistanzmaße*<sup>52</sup>. Eine Metrik  $d$  für Zeitreihen ist definiert als:

$$d : M_{ZR} \times M_{ZR} \rightarrow \mathbb{R}, \tag{A.6}$$

mit  $M_{ZR}$  als ZR-DB

so dass für die Zeitreihen  $x, y, z \in M_{ZR}$  die folgenden Bedingungen erfüllt sind:

$$\text{Nicht-Negativität}^{53}: \quad d(x, y) \geq 0 \tag{A.7}$$

---

<sup>51</sup> Leider ist der „tatsächliche Abstand“ nicht eindeutig bestimmbar und hängt von der Art der Daten und der zu realisierenden Anwendung stark ab. Trotzdem dient in der Literatur häufig der Euklidische Abstand als Referenz, obwohl dessen Einsatz aufgrund seiner fehlenden Robustheit (siehe oben und Abbildung A.4) ohne Vorverarbeitung problematisch sein kann.

<sup>52</sup> Pseudodistanzmaße erfüllen nicht alle Bedingungen für eine Metrik. Häufig sind jedoch für bestimmte Aufgaben auch nicht alle, sondern nur bestimmte Bedingungen relevant.

<sup>53</sup> Die Eigenschaften *Nicht-Negativität* und *Identität* werden in der Literatur manchmal als *Positive Definitheit* zusammengefasst.

$$\text{Identität}^{53}: \quad d(x, y) = 0 \stackrel{!}{\Leftrightarrow} x = y \quad (\text{A.8})$$

$$\text{Symmetrie}: \quad d(x, y) = d(y, x) \quad (\text{A.9})$$

$$\text{Dreiecksungleichung}: \quad d(x, y) \leq d(x, z) + d(z, y) \quad (\text{A.10})$$

In der Literatur finden sich zahlreiche Vorschläge für Ähnlichkeitsbewertungen von Zeitreihen. Eine ausführliche Beschreibung und ein Vergleich der meisten bekannten Distanzmaße für Zeitreihen findet sich in [51]. Am häufigsten werden jedoch der intuitiv verständliche und effizient berechenbare Euklidische Abstand und das komplexere aber leistungsfähige Dynamic Time Warping (DTW) eingesetzt. Im Folgenden werden einige relevante Distanzmaße für Zeitreihen gemäß [51], [103], [258] und [79] vorgestellt.

#### A.1.4.3 Punktweise Euklidische Distanz

Ein häufig verwendetes und leicht verständliches Abstandsmaß, das direkt auf reellwertigen Zeitreihen operiert, ist der sog. Euklidische Abstand<sup>54</sup> [81]. Er ist für zwei Zeitreihen  $X, Y$  der Länge  $n$  definiert als:

$$D_{\text{Euklid}}(X, Y) = \sum_{i=0}^{n-1} \sqrt{(x_i - y_i)^2} \quad (\text{A.11})$$

Es werden hier also die Differenzen der Werte zweier ZR summiert, die denselben Zeitbezug aufweisen (in Abbildung A.4 sind diese Werte durch Pfeile verbunden), weshalb auch von *Punktweiser Euklidischer Distanz* gesprochen wird.

Die Punktweise Euklidische Distanz kann bei linearer Komplexität ( $O(n)$ ) effizient berechnet werden und kann sich dabei mit vergleichsweise komplexen Distanzmaßen messen – insbesondere für sehr lange Zeitreihen [51] – weshalb sie als Basis für zahlreiche ähnlichkeitsbasierte Anwendungen fungiert. Sie kann als Metrik betrachtet werden, da sie die oben genannten Eigenschaften (siehe Formel (A.7), (A.9) und (A.10)) erfüllt.

Ein Nachteil des Euklidischen Abstands ist jedoch die Tatsache, dass er ohne eine entsprechende Vorverarbeitung nicht robust ist gegenüber den Transformationen

- zeitliche Skalierung

---

<sup>54</sup> Der euklidische Abstand stellt einen Spezialfall des gewichteten Euklidischen Abstands dar. Sein Quadrat entspricht außerdem einem Spezialfall der Mahalanobis-Distanz. Er stellt außerdem den zweiten Grad der allgemeinen  $L^p$ -Normen (also  $L^2$ ) dar (vergl. [339]).

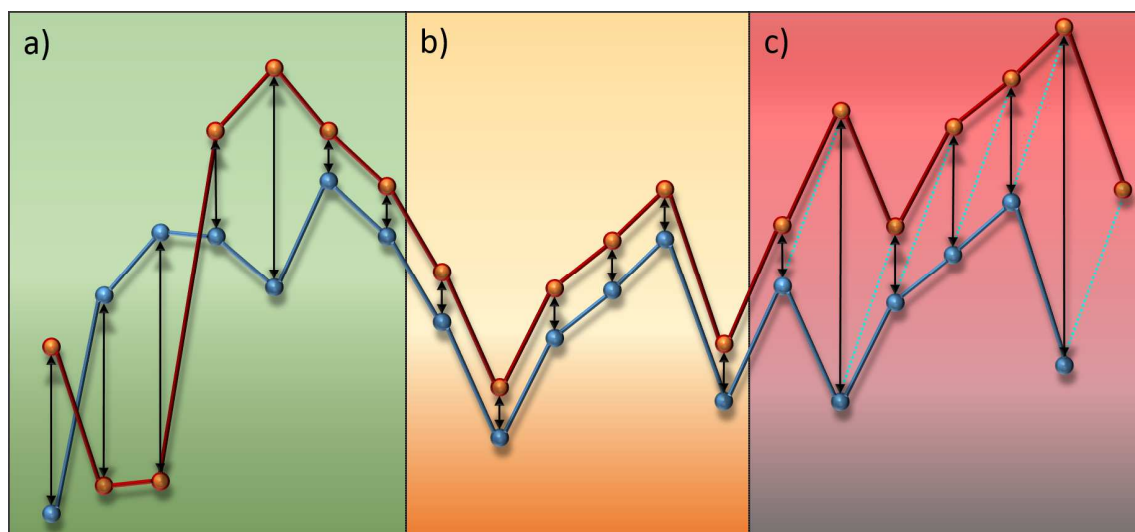
## Anhang A - Bewertung und Auswahl von Methoden zur skalierbaren Analyse umfangreicher Zeitreihendaten

- zeitliche Verschiebungen
- zeitliche Verzerrungen (engl. *Time Warping*)
- Skalierung des Wertebereichs
- Verschiebung des Wertebereichs
- Verzerrung des Wertebereichs

und außerdem nur für ZR gleicher Länge definiert ist.

Der letzte Punkt kann gelöst werden, indem nur überlappende Bereiche verglichen werden und das Ergebnis entsprechend gewichtet wird. Dann ist jedoch zu ermitteln, an welcher Position die ZR überlappen sollen oder es müssen alle Möglichkeiten getestet werden. Außerdem muss die Distanz dann entsprechend der Überlappung gewichtet werden.

Die Nachteile werden in Abbildung A.4 verdeutlicht: In Abschnitt *a)* der Abbildung scheint  $D_{Euklid}$  die Unterschiedlichkeit der beiden Kurven noch recht gut zu beschreiben. In Abschnitt *b)* jedoch verlaufen beide Kurven identisch, weisen aber einen Versatz im Wertebereich auf, welcher als zusätzliche Distanz in  $D_{Euklid}$  eingeht. Eine solche Verschiebung im Wertebereich kann beispielsweise durch eine Vorverarbeitung wie die in A.1.8.1 beschriebene Z-Normalisierung kompensiert werden, die einen solchen Versatz von ansonsten identischen ZR entfernt.



**Abbildung A.4:** Probleme bei der Distanzberechnung zweier Kurven - a) punktweise Differenz entspricht der Wahrnehmung, b) Versatz im Wertebereich, c) zusätzlicher Zeitversatz.

In Abschnitt *c)* weisen die beiden Kurven zusätzlich einen zeitlichen Versatz – hier um einen Zeitschritt – auf, welcher ebenfalls dazu führt, dass  $D_{Euklid}$  sich vergrößert, obwohl sich die Form des Verlaufs beider Kurven nicht unterscheidet (zusammengehörige Punkte sind in *c)* durch ge-

strichelte Linien verbunden). Für die Kompensierung zeitlicher Verschiebungen und Verzerrungen stehen *Sliding Window* Ansätze oder komplexe Ähnlichkeitsberechnungen wie z. B. das sog. *Dynamic Time Warping* zur Verfügung, das im nachfolgenden Abs. dargestellt wird.

#### A.1.4.4 *Dynamic Time Warping*

Das sog. *Dynamic Time Warping* (DTW)<sup>55</sup> eignet sich auch für komplexe Anwendungen, für die der Euklidische Abstand unzureichende Ergebnisse liefert, da dieser nicht alle Aspekte der Ähnlichkeit berücksichtigt, die ein menschlicher Betrachter bei der Ähnlichkeitsbewertung einzusetzen vermag. Das Verfahren, das grundlegend bereits 1959 in [28] beschrieben wurde, ermöglicht Vergleiche von ZR, die zeitlicher Skalierung, Verschiebung oder Verzerrung – wie etwa Stauchung und Dehnung – unterliegen. Außerdem kann auch die Distanz zwischen ZR unterschiedlicher Länge ermittelt werden.

Dies wird dadurch realisiert, dass nicht wie beim Euklidischen Abstand die Differenz gegenübergestellter Wertepaare aufsummiert wird, sondern auch mehrere Datenpunkte der einen ZR auf einen Datenpunkt der anderen bezogen werden können und umgekehrt. Mit diesem Trick lässt sich die Abstandsermittlung in ein Optimierungsproblem überführen: Zwei ZR werden so ausgerichtet, skaliert und lokal gestaucht bzw. gestreckt, dass sich eine minimale Distanz ergibt, sich also optimal überdecken. Natürlich muss dabei die zeitliche Reihenfolge der Werte erhalten werden. Als Maß für die Unterschiedlichkeit zweier ZR wird also die Distanz bei optimaler Überdeckung genutzt.

Für zwei Zeitreihen  $X = \{x_1, x_2, \dots, x_n\}$  der Länge  $n$  und  $Y = \{y_1, y_2, \dots, y_m\}$  der Länge  $m$  wird folgende  $n \times m$  Distanzmatrix gebildet, welche die Distanzen zwischen jedem Punkt aus  $X$  zu jedem Punkt aus  $Y$  enthält:

$$M_{Dist} = \begin{pmatrix} d(x_1, y_1) & d(x_1, y_2) & \cdots & d(x_1, y_m) \\ d(x_2, y_1) & d(x_2, y_2) & & \\ \vdots & & \ddots & \\ d(x_n, y_1) & & & d(x_n, y_m) \end{pmatrix} \quad (\text{A.12})$$

Die Distanz des  $i$ -ten Punktes in  $X$  zum  $j$ -ten Punkt in  $Y$  wird dann mit  $M_{Dist}(i, j)$  referenziert [51]. Aus der Distanzmatrix kann nun ein minimaler *Verzerrungspfad* (von engl. *Warping Path*, kurz WP), bestehend aus benachbarten Elementen der Distanzmatrix als ermittelt werden:

$$WP = \{w_1, w_2, \dots, w_k, \dots, w_K\}, \text{ mit} \quad (\text{A.13})$$

$$\max(n, m) < K < (m + n - 1) \text{ und } w_k = M_{Dist}(i, j)$$

---

<sup>55</sup> Eine deutsche Bezeichnung taucht in der Literatur nicht auf.

Da es sich um eine Optimierungsaufgabe handelt, die gut in Teilschritte zerlegbar ist, deren Zwischenergebnisse wiederverwendet werden können, wird zur Berechnung typischerweise die Methode der Dynamischen Programmierung [27] eingesetzt, woher sich auch der Name *Dynamic Time Warping* herleitet.

Außerdem können die Möglichkeiten weiter eingeschränkt werden, indem ein Maximalwert für die Anzahl zusammenfassbarer Werte eingeführt wird, oder die Möglichkeiten der Verzerrungspfade durch Ausschluss von Randbereichen der Distanzmatrix mit verschiedener Form eingeschränkt werden. Eine detaillierte Beschreibung dieser Möglichkeitseinschränkungen (engl. *Constraints*) findet sich etwa in [49] und [243]. Eine Variante mit geringerer Komplexität ( $O(n)$ ) und damit schnellerer Laufzeit wurde von Keogh u. a. unter dem Namen UCR-DTW präsentiert [243], [209].

Es gilt jedoch zu beachten, dass bei einer Ähnlichkeitssuche mittels DTW für jeden Suchlauf stets alle Zeitreihen erneut sequentiell durchlaufen werden müssen, während für Suchen in indizierten ZR-DB dieser Aufwand nur einmalig bei der Indizierung aufgebracht werden muss. Ist die Indizierung abgeschlossen, kann sich die Komplexität der Suche dramatisch verringern auf ( $O(k)$ , mit  $k < n$ ), wobei  $k$  von der eingesetzten Datenrepräsentation und Indizierungsmethode abhängt.

#### A.1.4.5 Edit-Distanzen

Die Familie der sog. *Edit-Distanzen* basiert auf dem Vergleich zweier Zeichenketten  $a$  der Länge  $n$  und  $b$  der Länge  $m$  anhand der minimalen Anzahl notwendiger Editieroperationen, um eine der beiden Sequenzen in die andere zu überführen. Sie geht zurück auf den russischen Wissenschaftler Vladimir Iosifovic Levenshtein<sup>56</sup>, der dieses Konzept bereits 1965 einführte ([173] wurde erst 1966 publiziert). Das von ihm vorgeschlagene metrische Distanzmaß für den Raum symbolischer Sequenzen wird deshalb heute als Levenshtein-Distanz bezeichnet. Edit-Distanzen werden häufig bei der Text-Analyse eingesetzt, um z. B. Worte als unterschiedliche Formen desselben Wortstamms zusammenzufassen oder um durch Rechtschreibfehler veränderte Worte zu erkennen.

Für die ursprüngliche *Levenshtein-Distanz* werden nur einzelne *Einfüge-, Lösch-, und Ersetzungsoperationen* einzelner Zeichen zugelassen und gezählt. Die Berechnung weist eine Komplexität von  $O(nm)$  auf.

---

<sup>56</sup> Um Verwirrung zu vermeiden, wird hier die Schreibweise verwendet, die bei der Verleihung der Hamming-Medaille an Levenshtein gebraucht wurde. Weitere in der Literatur vorkommende Transliterationen sind „Wladimir Iossifowitsch Lewenstein“ und „Vladimir Iosifovič Levenštejn“.

Die Levenshtein-Distanz kann als Erweiterung der Hamming-Distanz [229] betrachtet werden. Letztere verzichtet lediglich auf Einfüge-Operationen und ist damit nur für Zeichenketten gleicher Länge definiert. Gewichtet man die verschiedenen Operationen unterschiedlich, ergibt sich die Gewichtete Levenshtein-Distanz [201].

Eine Variante, die eine zusätzliche Operation – die Verschiebung eines Zeichens innerhalb einer Zeichenkette – einführt, ist die sog. *Damerau-Levenshtein-Distanz*. Die Levenshtein Distanz stellt außerdem in gewisser Weise eine auf Zeichenketten spezialisierte Form des DTW dar, da durch Einfüge- und Löschoptionen zeitliche Verzerrungen ausgeglichen werden können.

#### A.1.4.6 Längste gemeinsame Teilsequenz

Das Verfahren der Längsten Gemeinsamen Teilsequenz (kurz LCSS von engl. Longest Common Sub-Sequence) [310] basiert auf der Ermittlung der längsten zusammenhängenden Zeichenfolge, die in zwei Zeichenketten  $a$  und  $b$  der Längen  $n$  und  $m$  enthalten ist. Es nutzt die Länge dieser längsten gemeinsamen Sequenz als Maß für die Ähnlichkeit  $\ddot{A}$  von  $a$  und  $b$ . Da sich hierdurch große Teile der Zeitreihen unterscheiden können, ohne dass dies in die Berechnung eingeht, zeigt sich dieses Ähnlichkeitsmaß als sehr robust gegenüber Rauschen und Ausreißern, sofern diese nur gelegentlich vorkommen. Außerdem kann – ähnlich wie bei DTW – eine optimale Ausrichtung der beiden Zeitreihen ermittelt werden, in der Lücken enthalten sein dürfen. Die Berechnung kann formal folgendermaßen [51] beschrieben werden:

$$\ddot{A}_{LCSS}(i, j) = \begin{cases} 0 & \text{für } i = 0, \\ 0 & \text{für } j = 0, \\ 1 + LCSS[i - 1, j - 1] & \text{für } a_i = b_j, \\ \max(LCSS[i - 1, j], LCSS[i, j - 1]) & \text{sonst.} \end{cases} \quad (\text{A.14})$$

mit  $1 \leq i \leq n$  und  $1 \leq j \leq m$

Um direkt auf reellwertigen Zeitreihen operieren zu können, kann die Bedingung ( $a_i = b_j$ ) aus obiger Gleichung zu ( $|a_i - b_j| < \epsilon$ ) umgeformt werden, so dass ein Schwellwert  $\epsilon$  für die Distanz zweier reellwertiger Elemente angegeben werden kann, statt exakt zu vergleichen.

$\ddot{A}_{LCSS}(n, m)$  ergibt einen Ähnlichkeitswert für  $a$  und  $b$ , welcher die Länge  $l$  der längsten gemeinsamen Teilsequenz darstellt und folgendermaßen [244] in eine Distanz umgerechnet werden kann:

$$D_{LCSS}(a, b) = \frac{n + m + 2l}{n + m} \quad (\text{A.15})$$

Die Komplexität des Verfahrens beträgt  $O(nm)$ , kann jedoch zu  $O((n + m)\delta)$  reduziert werden, falls eine Beschränkung wie z. B.  $|i - j| < \delta$  eingeführt wird.

#### A.1.4.7 Verteilungsbasierte Distanzen

Ein weiterer Ansatz zur Bestimmung der Ähnlichkeit zwischen Zeitreihen besteht darin, die jeweiligen Werteverteilungen zu vergleichen, statt die Zeitreihen selbst gegenüberzustellen. Hierdurch wird jegliche zeitliche Information bei der Ähnlichkeitsbewertung außer Acht gelassen. Form und Struktur der ZR bleiben also unberücksichtigt, was zunächst einen Nachteil darstellt. Andererseits ist ein verteilungsbasiertes Verfahren robust gegenüber zeitlichen Transformationen wie Verschiebungen, Skalierungen und sogar lokalen Verzerrungen.

In [61] wird ein Verfahren zur Ähnlichkeitsbewertung vorgeschlagen, das auf normalisierten Histogrammen basiert. Es eignet sich gemäß der Autoren für Anfragen zur Prüfung der Existenz eines Musters und zur Ermittlung von zu einem Muster passenden ZR.

#### A.1.4.8 Strukturbasierte Ähnlichkeit

Ähnlich der verteilungsbasierten Distanzen lassen sich weitere formunabhängige Distanzmaße über strukturelle Eigenschaften höherer Ebene definieren. Diese sind v. a. nützlich für Ähnlichkeitsvergleiche langer Zeitreihen, da formbasierte Verfahren hier meist versagen [182]. Geeignete strukturelle Eigenschaften von Zeitreihen lassen sich beispielsweise über Segmentierungsverfahren oder anhand seltener und häufiger Muster ermitteln. Ein Beispiel stellt das von Lin und Li vorgestellte Bag-of-patterns-Verfahren [182] dar, welches Ähnlichkeit anhand charakteristischer häufiger Teilsequenzen ermittelt.

### A.1.5 Dimensionsreduzierende Zeitreihenrepräsentationen

Nach der in Abs. A.1.1 beschriebenen klassischen Zeitreihenrepräsentation als Vektor reeller Zahlen wurden in den vergangenen Jahrzehnten nach und nach alternative Datenrepräsentationen für Zeitreihen vorgeschlagen. Viele davon besitzen die Eigenschaft, durch unterschiedliche Arten der Approximation die „Dimensionalität“ bzw. die „Mächtigkeit“ der Ursprungsdaten teilweise deutlich zu reduzieren<sup>57</sup>. Sie basieren auf der Beschreibung von Zeitreihen durch kompakte Merkmale und ermöglichen dadurch die Extraktion von Information auf einem erhöhten Abstraktionsniveau. Viele der in Abs. A.1.3 dargestellten Analyseaufgaben werden so effizienter durchführbar und besser skalierbar für große Datenmengen.

---

<sup>57</sup> Im Zusammenhang mit der hohen Dimensionalität von ZR wurde in der englischsprachigen Literatur oft von einem Fluch („*the curse of dimensionality*“ geprägt durch R. Bellman [26], [154], [308]) gesprochen. Hiermit ist gemeint, dass die Performanz einer Ähnlichkeitssuche für Zeitreihen einbricht und sich auf die einer sequentiellen Suche reduziert, sobald die Dimensionalität der ZR einen Wert von ca. 16 überschreitet. Durch dimensionsreduzierende Repräsentationen kann dieses Problem in vielen Fällen überwunden werden.



Zu den ersten dimensionsreduzierenden Repräsentationen bzw. Transformationen, welche im Bereich des Data-Mining genutzt wurden (vergl. [181],[258]), gehören die auf spektraler Zerlegung basierenden zu den ältesten. V. a. die Diskrete Fourier Transformation (DFT) [1],[41], welche ein zeitdiskretes Signal von der Zeit- in die Frequenzdomäne transformiert und als nach Frequenz geordnete Liste der Koeffizienten komplexer Sinuskurven darstellt, ist nach wie vor ein häufig eingesetzter, leistungsfähiger Ansatz. Weitere Vertreter sind die Diskrete Cosinus Transformation (DCT) [2], welche eine ZR als Summe von Cosinus-Funktionen darstellt und dabei nur wenige Koeffizienten benötigt [188].

Zu den etablierten Repräsentationen gehören auch die Singulärwertzerlegung (kurz SVD von engl. Singular Value Decomposition) [162], die Wavelet-basierten Verfahren (kurz DWT von engl. Discrete Wavelet Transform) [56] – insbesondere die Haar-Wavelet-Transformation (HWT) – und die rekursiven Tschebyscheff-Polynome (CHEB) [46].

Des Weiteren wurden neuerdings sog. abschnittsweise Approximationen vorgeschlagen, welche die Dimensionalität reduzieren, indem die aufeinanderfolgenden Teilsequenzen bzw. Segmente von ZR approximiert werden. Prominente Vertreter sind die Abschnittsweise Lineare Approximation (engl. Piecewise Linear Approximation, kurz PLA) [260], welche ein Signal durch Geradenabschnitte annähert, die Abschnittsweise Aggregat-Approximation (kurz PAA) [339],[151] (siehe A.1.7) und die Adaptive Abschnittsweise Konstante Approximation (engl. Adaptive Piecewise Constant Approximation, kurz APCA) [55]. Daneben existieren zahlreiche weitere Repräsentationen, die sich oft besonders für bestimmte Aufgaben eignen.

Abbildung A.5 zeigt eine hierarchisch gruppierte Übersicht. Hier wird gemäß [176] unterschieden zwischen datenadaptiven und nicht datenadaptiven Transformationen. Letztere zeichnen sich dadurch aus, dass die zu wählenden Parameter für alle Eingangszeitreihen gleich bleiben, während bei datenadaptiven Transformationen die Parameter jeweils an die Eingangszeitreihen angepasst werden müssen. Die Funktionsweise der Approximationen ist oft relativ ähnlich. Zur Veranschaulichung sind in Abbildung A.6 die Resultate einiger Transformationen dargestellt. Bei allen findet eine Zusammenfassung der Ursprungswerte zur Dimensionsreduktion statt, indem die ursprüngliche Zeitreihe durch wenige Aggregate oder Koeffizienten beschrieben wird. In

Tabelle A.2 werden zentrale Eigenschaften der unterschiedlichen Transformationen gegenübergestellt. Zusätzlich zur Datenkompression versprechen viele Repräsentationen die Umsetzbarkeit neuer Analyseaufgaben und -verfahren, die sich besonders für Analysen großer Zeitreihenbestände als nützlich erweisen, v. a. schnelle unscharfe Suchen und effiziente Indizierung, wie in Abbildung A.2 dargestellt.

Einige der benötigten Operationen sind nur für diskrete Wertebereiche definiert und nicht auf reellwertige Zeitreihen anwendbar. Diese beinhalten u. a. das sog. *Hashing*, *Markov Modelle*, *Suffix-Bäume* und Entscheidungs-Bäume (vergl. [176]). Deshalb werden für derartige Aufgaben vermehrt auch diskrete Repräsentationen wie etwa die *Symbolische Aggregat-Approximation (SAX)* eingesetzt, welche auf PAA aufbaut und zusätzlich den Wertebereich diskretisiert.

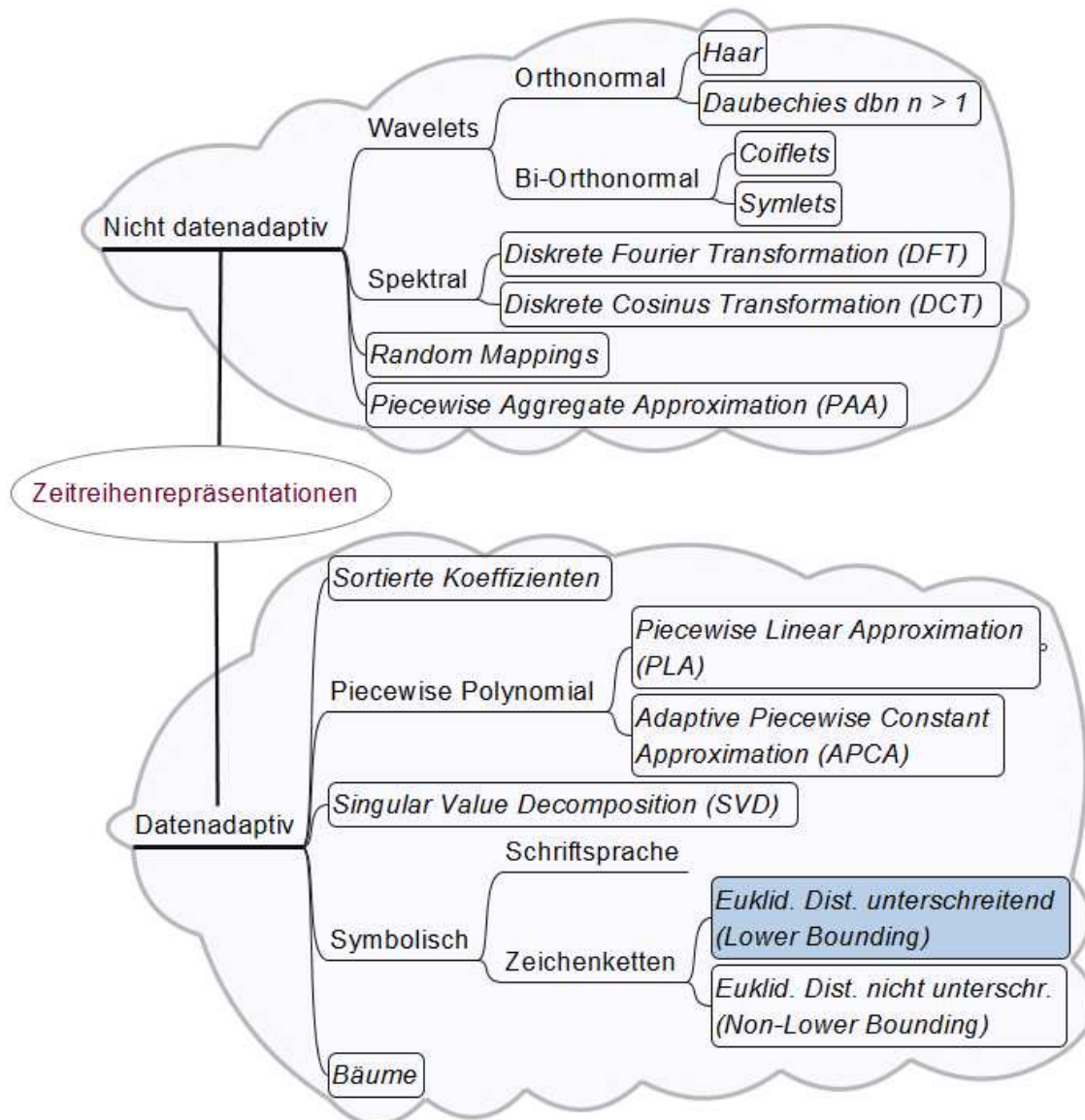


Abbildung A.5: Zeitreihenrepräsentationen. Quelle: [176], modifiziert.

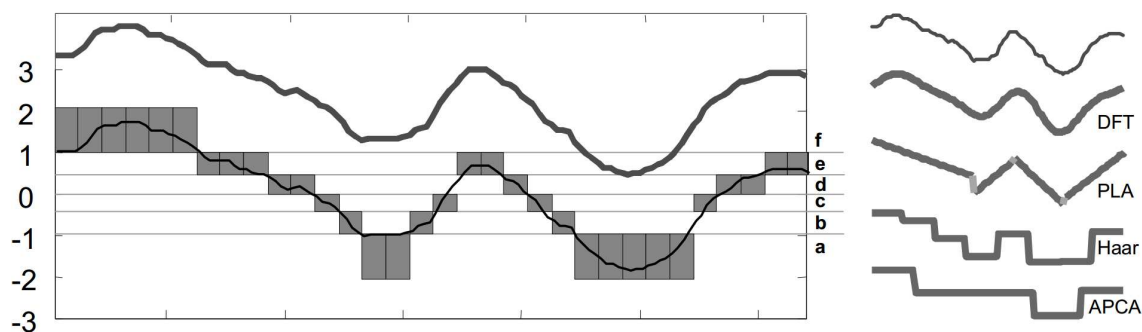
**Tabelle A.2:** Relevante Eigenschaften im Data-Mining gebräuchlicher  
Zeitreihenrepräsentationen, Angaben teilw. aus [258] (identisch zu Tabelle 2.1).

	Komplexität	Anwendung	Approximation	Besonderheiten
APCA	?	vielseitig einsetzbar, robust	je nach Parameterwahl und Signal	geeignet für energiereiche Signale, zwei Koeffizienten pro Segment, insgesamt aber wenige Koeffizienten
DFT	$O(n \log(n))$	Kompression, Periodizitäts- analyse, Signalbereinigung	sehr gut für viele oder energiereichste Koeffizienten, hinreichend für die ersten $n$ . Nicht auf alle Signale anwendbar (stark unperiodische)	gute Genauigkeit, manche Signale erfordern viele Koeffizienten für gute Approximation, <b>Indizierung</b> <b>komplex</b>
HWT	$O(n)$	Kompression, Periodizitäts- analyse	unter Einschränkungen einstellbar	einfachste Wavelet-Transformation, hierarchisch
PAA	$O(Nm)$ bei $N$ PAA- Segmenten und $m$ Sliding Windows	vielseitig einsetzbar, robust	Zeitlich beliebig einstellbar	einfach, wenige Parameter
PLA	$O(n \log(n))$	Signal-bereinigung, Trend-analyse	einstellbar	Inkrementeller Algorithmus, Fehler berechenbar
SAX	$O(PAA)$ + $O(n)$	vielseitig einsetzbar, robust	flexibel einstellbar	diskret (symbolisch), wenige Parameter, analytische Verfahren aus Textanalyse & Bioinformatik nutzbar (Markov Modelle, Suffix- Bäume, Regelextraktion, ...)
SVD	$\min \left\{ \begin{array}{l} O(M^2 n), \\ O(Mn^2) \end{array} \right\}$ bei $M$ Sequenzen der Länge $n$	vielseitig, Suchen (z. B. PageRank)	optimal	nicht für einzelne Sequenzen, Neuberechnung nach Hinzufügen neuer Sequenzen notwendig

### A.1.6 Wahl einer Zeitreihenrepräsentation für skalierbare Analysen

Die Wahl der Datenrepräsentation ist stets abhängig von den durchzuführenden Analyseaufgaben einerseits und den Daten andererseits, was Evaluationsstudien (z. B. [152]) bestätigen. Deshalb werden im Folgenden Kriterien für die Wahl einer Repräsentation aufgestellt, mit welcher das im Rahmen dieser Arbeit entwickelte Konzept zur skalierbaren Analyse großer Zeitreihendaten verwirklicht werden kann.

Ein erstes wichtiges Kriterium für die Auswahl einer geeigneten Datenrepräsentation ist das Vorhandensein eines definierten Distanz- bzw. Ähnlichkeitsmaßes und dessen Eigenschaften. Ein solches wird benötigt, um verschiedene Zeitreihen auch in der alternativen Repräsentation miteinander vergleichen zu können. Dabei will man sichergehen, dass die Ordnung der Distanzen zwischen allen verglichenen Zeitreihen im ursprünglichen Wertebereich auch für die Distanzen im Merkmalsraum erhalten bleibt, dass also z. B. die beiden ähnlichsten Sequenzen auch im Merkmalsraum den kleinsten Abstand haben. Hierzu muss das Distanzmaß die sog. Kontrahierende Eigenschaft (engl. *Contractive Property* oder auch oft *Lower-Bounding Condition*) besitzen (siehe Anh. ). Dies bedeutet, dass der Abstand zwischen zwei Zeitreihen im Merkmalsraum nie größer sein darf als der „tatsächliche“ in den Originaldaten. Mit dem tatsächlichen oder realen Abstand ist in der Praxis meist der weit verbreitete Euklidische Abstand gemeint (siehe Abs. 0). Ist ein Distanzmaß kontrahierend, bedeutet das also, dass eine darauf basierende Ähnlichkeitssuche zwar falsche Treffer, aber keine falschen Ausschlüsse zum Ergebnis haben kann.



**Abbildung A.6:** Vergleich der Ableitung von Zeitreihenrepräsentationen. Quelle: [181].

Links oben: Originalzeitreihe, links unten: Vertikale und horizontale Quantisierung, rechts Resultate unterschiedlicher Approximationen.

Ein weiteres wichtiges Kriterium stellen die Analyseaufgaben dar, die realisiert werden sollen. Im Rahmen dieser Arbeit gehören dazu v. a. die Aufgaben *inhaltsbasierte Anfrage* und *unscharfe Suche*, sowie die Detektion häufiger und seltener Muster. Für eine tiefergehende Betrachtung der einzelnen Repräsentationen im Kontext von Indizierung und ähnlichkeitsbasierten Suchen sei auf [258], [79] und [151] verwiesen. Dort wird auch auf die Eignung der Repräsentationen für Dis-

tanzmaße eingegangen und die Komplexität der Indizierung verglichen. Allgemeine Anforderungen an eine Zeitreihenrepräsentation, können basierend auf [79] folgendermaßen zusammengefasst werden:

- signifikante Reduktion der Dimensionalität
- Konzentration auf grundlegende Charakteristik
  - sowohl auf lokaler Ebene (Form)
  - als auch auf globaler Ebene (Struktur)
- effiziente Berechenbarkeit
  - Geringe Zeitkomplexität
  - niedrige Ressourcenanforderungen
- gute Rekonstruierbarkeit des Originalsignals
- Unempfindlichkeit für Rauschen bzw. implizite Glättung

Die meisten Zeitreihenrepräsentationen stellen – teilweise anwendungsspezifische – Kompromisse bei der Gewichtung dieser Anforderungen dar.

### A.1.7 Abschnittsweise Aggregat-Approximation - PAA

Die Abschnittsweise Aggregat-Approximation (engl. Piecewise Aggregate Approximation, kurz PAA) [151] repräsentiert eine ZR als Mittelwerte über gleich große Zeitintervalle. Die Anzahl der Zeitintervalle muss kleiner oder gleich 58 der Länge der Ursprungszeitreihe sein, kann aber ansonsten frei gewählt werden<sup>59</sup>. Hierdurch kann die Dimensionalität der ZR beliebig reduziert werden. Der Vorgang kann als irreversible Kompression betrachtet werden.

Formal wird eine Zeitreihe  $X$  der Länge  $n$  in einen Vektor  $\bar{X} = (\bar{x}_1, \dots, \bar{x}_M)$  der Länge  $M$  überführt:

$$\bar{x}_i = \frac{M}{n} \sum_{j=\frac{n}{M}(i-1)+1}^{\frac{n}{M}i} x_j \quad (\text{A.16})$$

mit  $1 \leq i \leq n$ ,  $1 \leq j \leq M$  und  $M \leq n$ .

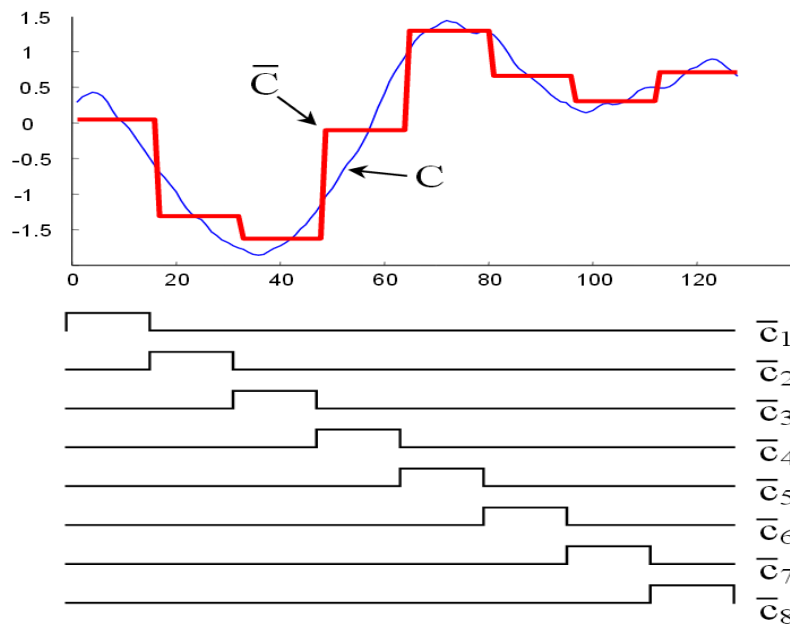
Um die Dimensionalität in zeitlicher Richtung von  $n$  auf  $M$  zu reduzieren, wird die Zeitreihe also in  $M$  gleich große Teilabschnitte aufgeteilt, deren Mittelwerte die neuen Datenpunkte darstellen.

---

<sup>58</sup> Für den trivialen Fall, in welchem die Anzahl zu mittelnder Zeitintervall gleich der Länge der Zeitreihe gewählt wird, bleibt die Ursprungszeitreihe unverändert erhalten.

<sup>59</sup> Ein weiterer trivialer Fall ist die Bildung eines Mittelwertes über die gesamte ZR.

Dies ist deshalb zulässig, da Nachbarwerte in Zeitreihen im Allgemeinen stark miteinander korrelieren<sup>60</sup>. Die Charakteristik von Intervallen benachbarter Werte kann so durch den Mittelwert innerhalb des Intervalls gut beschrieben werden. Der Wert  $M$  wird in der Literatur als *PAA-Größe* bezeichnet, die resultierende ZR als *PAA-Sequenz*. Durch die PAA-Codierung wird es außerdem möglich, Zeitreihen unterschiedlicher Größe miteinander zu vergleichen, indem beide in dieselbe Anzahl zeitlicher Aggregatwerte zerlegt werden.



**Abbildung A.7:** Reduktion einer Zeitreihe mit 128 Datenpunkten zu einer PAA-Sequenz mit acht Mittelwerten als lineare Kombination einfacher Basisfunktionen. Quelle: [176].

Die Konklusionen der Beiträge [151] und [258], in denen dimensionsreduzierende Zeitreihenrepräsentationen verglichen werden, kann so zusammengefasst werden, dass eine vergleichsweise einfache Repräsentation wie PAA sich gegenüber komplexeren Transformationen durchaus behaupten kann, um Indizierungen oder ähnlichkeitsbasierte Suchen durchzuführen. Während PAA ähnliche Funktionalität und Resultate wie Haar-Wavelets aufweist und die erzielte Genauigkeit für Ähnlichkeitsvergleiche der von DFT-basierten Ansätzen entspricht (vergl. [258]), ist PAA vergleichsweise einfach verständlich und berechenbar und besitzt nur einen einzigen einzu-

<sup>60</sup> Grundlage hierfür ist die starke Autokorrelation von Zeitreihen, welche die Eigenschaft besitzen, dass deren Werte nicht rein zufällig sind. Von dieser Eigenschaft kann dann ausgegangen werden, wenn es sich beispielsweise um Messdaten handelt und gemäß des Nyquist-Shannon-Abtasttheorems eine geeignete Abtastrate oberhalb der Nyquist-Rate gewählt wurde.

stellenden Parameter – die PAA-Segmentanzahl („PAA-Größe“). PAA zeigt sich robust gegenüber verrauschten Signalen und Ausreißern und besitzt ein leistungsfähiges Ähnlichkeitsmaß. Von den in [258] getesteten nicht-symbolischen Zeitreihenrepräsentationen war die Indizierung mit PAA am schnellsten möglich.

Insgesamt ergeben sich nach [151] folgende Vorteile von PAA gegenüber anderen Repräsentationen:

- Schnelle, effiziente Berechenbarkeit
- Anfragen beliebiger Länge möglich (im Gegensatz zu *DFT* & *DWT*).
- Einfügen und Löschen von ZR in konstanter Zeit (im Gegensatz zu *SVD*).
- Bearbeitbarkeit von Anfragen, die kürzer sind als die ursprünglich indizierte Zeitreihe
- Unterstützung des sog. *gewichteten Euklidischen Abstands*.

Der letzte Punkt wird als besonders nützlich angesehen, da der gewichtete Euklidische Abstand fortschrittliche Anfragetechniken unterstützt, wie z. B. das sog. Relevance Feedback [150]. Für PAA ist folgendes Ähnlichkeitsmaß  $D$  für zwei Zeitreihen in PAA Repräsentation  $\bar{X}$  und  $\bar{Y}$  definiert:

$$D(\bar{X}, \bar{Y}) \equiv \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^N (\bar{x}_i - \bar{y}_i)^2} \quad (\text{A.17})$$

Hierbei ist  $n$  die Anzahl der Ursprungswerte und  $N$  die gewählte Anzahl an PAA-Segmenten. Die Dimensionalität wird also von  $n$  auf  $N$  reduziert. Diese Distanz unterschreitet die Euklidische Distanz<sup>61</sup> derselben Zeitreihen im ursprünglichen Werterraum [151], was ansonsten nur für die Distanzmaße der Transformationen *DFT*, *DWT*, *SVD*, *APCA* und *PLA* gilt. Damit stellt die PAA eine optimale Basis für die Definition symbolischer Repräsentationen wie *SAX* dar. Die Komplexität der PAA Codierung kann durch Verwendung eines sog. Sliding Window Algorithmus von  $O(NM)$  auf  $O(Mm)$  reduziert werden, wobei  $m$  die Anzahl der Fenster bezeichnet.

---

<sup>61</sup> Die Unterschreitung der Euklidischen Distanz (engl. *Lower Bounding Condition*) ist für die Gültigkeit ähnlichkeitsbasierter Suchergebnisse oder Klassifikationen und Cluster-Bildungen von entscheidender Bedeutung. Vor der Einführung der Symbolischen Aggregat-Approximation, die auf PAA basiert, war keine *symbolische* Repräsentation für Zeitreihen bekannt, welche diese sog. kontrahierende Eigenschaft besaß.

### A.1.8 Symbolische Aggregat-Approximation (SAX)

Ein bekannter Vertreter der neuen Datenrepräsentationen für Zeitreihen ist die auf der PAA aufbauende *Symbolische Aggregat-Approximation* (SAX) [176], welche 2003 von Jessica Lin, Eammon Keogh u. a. eingeführt wurde und seither intensiv akademisch untersucht und erfolgreich eingesetzt wurde. Die Funktionsweise der SAX-Codierung wurde in Abs. 2.3.1 beschrieben und die SAX-basierte Ähnlichkeitsbewertung von Zeitreihen mit dem SAX-Distanzmaß MINDIST in Abs. 2.3.2. Die zahlreichen Verbesserungsvorschläge aus der Literatur sowie einige erfolgreiche Anwendungsbeispiele werden in den folgenden beiden Abschnitten besprochen.

#### A.1.8.3 Verbesserungen und Erweiterungen von SAX

Diverse Autoren haben sich mit den informationstheoretischen Eigenschaften und möglichen Verbesserungen und Erweiterungen von SAX beschäftigt. Muhammad u. a. stellen in [123] einen Vorschlag für ein abgewandeltes Distanzmaß vor, das den Euklidischen Abstand besser annähern soll, während gleichzeitig die Unterschreitung des Euklidischen Abstands erhalten bleibt. In [291] wird ein ähnlicher Vorschlag für die Verbesserung der Genauigkeit gemacht, wobei hier die Richtung des Verlaufs der Teil-Zeitreihe, welche in ein SAX-Symbol eingeht, in die Distanzberechnung eingehen soll. Der Gedanke ist zwar interessant, nach Meinung des Autors können die in [291] als problematisch dargestellten Fälle jedoch durch geeignete Wahl der SAX-Parameter (PAA- und Alphabets-Größe) korrekt codiert werden. Insbesondere sollte bei der Wahl der PAA-Größe stets die Nyquist-Frequenz durch die Anzahl an PAA-Segmenten überschritten werden, bzw. es sollte darauf geachtet werden, dass die kürzeste vorkommende Periode des Ursprungssignals zu mindestens zwei Segmenten zusammengefasst wird, da ansonsten ein Effekt ähnlich dem des Aliasing auftreten kann.

In [328] wird ein zu SAX bzw. iSAX weitestgehend identischer Ansatz vorgestellt, welcher keine Z-Normalisierung vorsieht und sich ansonsten lediglich durch die Berechnung der Nachschlagetabellen für die Umbruchpunkte unterscheidet. Hierdurch soll jedoch angeblich eine bessere „Online-“ Konvertierung möglich sein. Die Autoren entscheiden sich jedoch zugunsten einer besseren Vergleichbarkeit der resultierenden Symbole gegen dynamische und für vorberechnete Tabellen auf Basis historischer Daten, weshalb dieser Ansatz keine Verbesserung darstellt.

In weiteren Arbeiten wird versucht, den Funktionsumfang von SAX zu erweitern. So wird etwa in [221] die Extraktion von Regeln aus Zeitreihen beschrieben, und in [315] die Erkennung von Assoziationen. [330], [284] und [284] nutzen SAX zum Auffinden häufiger Muster. Dabei ist [330] vom Gedächtnis menschlicher Immunzellen inspiriert und der in [284] beschriebene Ansatz findet auch Motifs unbekannter Länge. [210] verallgemeinert SAX für den multidimensionalen Fall. In [14] und [15] wird SAX erweitert, um Zeitreihen zu segmentieren. [121] beschreibt eine alternative ähnlichkeitsbasierte Suche für SAX. In [37] wird ein neues Komplexitätsmaß für



nichtstationäre Zeitreihen eingeführt, das auf SAX beruht. In [214] werden SAX-Worte als Eingangsdaten für ein Markov-basiertes Vorhersagesystem genutzt. In [332] wird mit SAX nach sog. *closed flexible patterns* (CFPs) in Zeitreihen-Datenbanken gesucht. Das Paper [53] setzt SAX bzw. iSAX zur Analyse von Motifs ein. In [314] werden mittels SAX und Hidden Markov Modellen Umrissformen analysiert.

Durch die einfache und nachvollziehbare Funktionsweise von SAX bei gleichzeitig hoher Genauigkeit und effizienter Berechenbarkeit erfreut sich SAX besonders im Data-Mining Umfeld zunehmend großer Beliebtheit, was durch die zahlreichen Erweiterungsvorschläge und die auf SAX basierenden Anwendungen aus Abs. 0 belegt wird.

#### A.1.8.4 Anwendungen von SAX und Literatur

Obwohl die SAX-Repräsentation noch relativ jung ist, wurde mittlerweile in zahlreichen Anwendungsbereichen und Projekten ihr Einsatz erprobt. Des Weiteren gab es auch Verbesserungs- und Erweiterungsvorschläge. Die im Folgenden besprochene Literatur soll einen Überblick über den vielfältigen Einsatz in unterschiedlichsten Bereichen geben. Die Literatursammlung wurde teilweise von der SAX Homepage der Autoren [253] übernommen, erweitert und thematisch gruppiert.

In den Bereichen **Medizin und Gesundheit** gibt es zahlreiche veröffentlichte Verfahren, die auf SAX basieren. Etwa wird SAX in [7] zur Analyse komplexer kinetischer Abläufe eingesetzt. Der Einsatz zur Detektion häufiger Muster in der Telemedizin wird in [270], [269] und [74] untersucht. In [166] werden mittels SAX ECG Daten analysiert. Eine Visualisierung zur Unterscheidung medizinischer Zustände von Patienten findet sich in [224]. In [80] werden Bilder aus der Magnetresonanztomographie analysiert. In [60] und [59] werden Handabdrücke biometrisch erkannt. Ein medizinisches Unterstützungssystem für Experten wird in [161] beschrieben. [122] nutzt einen an SAX orientierten Ansatz, um das Wohlbefinden krankheitsanfälliger Menschen zu überwachen. In [113] wird SAX zur Identifikation von Hirnzuständen verwendet. In [301] wird SAX in Kombination mit Zufallsprojektionen (*random projections*) verwendet, um aufgenommene Bewegungsdaten zu analysieren, in [54], [198] und [12] werden häufig ausgeführte Bewegungen von Personen ermittelt und [175] nutzt SAX zur Gestenerkennung. In [22] werden die Kreuzungen von Trajektorien untersucht, in [159] die Atmung von Musikern, in [78] der menschliche Gang. In [239] wird SAX zur Erkennung bewaffneter Angriffe eingesetzt. In [267] werden unter Wahrung der Privatsphäre häufige Muster in personenbezogenen Daten ermittelt.

Auch für Analysen in den Bereichen **Bioinformatik und Genetik** eignet sich SAX durch seine Ähnlichkeit zum DNA-Code hervorragend. So verwenden Androulakis u. a. in [6] ein auf SAX basiertes Verfahren, um Gene mit hohem Informationsgehalt zu finden, ähnliche Anwendungen

werden in [337], [254] und [335] beschrieben. In [141] wird SAX modifiziert, um ähnliche Subsequenzen in Genen zu finden.

Im Bereich **Umwelt** werden z. B. in [202] Tornados untersucht und in [24] Messdaten über Hurrikans ausgewertet. [202] wendet SAX auch auf die Wettervorhersage an, [141] zur Klassifikation von Umgebungsgeräuschen. In [305] werden die Wasserstände von Flüssen mittels SAX ausgewertet.

Auch in den Bereichen **Industrie und Wirtschaft** wurde SAX bereits eingesetzt. So werden etwa in [215] Anomalien in Schiffsmotoren mittels SAX festgestellt. Eine Charakterisierung der Wirkmechanismen von Entzündungshemmern mithilfe von SAX wird in [336] beschrieben. In [82] wird mit SAX eine Bewegungssegmentierung für Roboter realisiert. Eine SAX-basierte Qualitätskontrolle bei der Halbleiterherstellung wird in [183] dargestellt. In [107] wird eine Fehleranalyse für Analogschaltkreise beschrieben und [268] nutzt SAX für ein Clustering telemetrischer Wärmeentwicklungs-Daten. In [187] werden mittels SAX Finanzdaten ausgewertet.

Es gibt unzählige weitere Bereiche, in denen SAX angewandt wird. Beispielsweise setzt [238] SAX in der Analyse von **Netzwerkdaten** ein, in [222] werden Anomalien in Netzwerkübertragungen ermittelt. In [106] kommt SAX bei der Analyse besonders großer Netzwerkverkehrsdaten zum Einsatz. In [343] werden wichtige Ereignisse in WLAN-Netzen ermittelt. [333] beschreibt die Nutzung der SAX-basierten Software *Viztree* [177], um Spuren in CPUs auszuwerten. In [5] wird böswilliger Internetverkehr über einen sog. Honeypot-Mechanismus mittels SAX erkannt. In [331] werden die Klassifizierung von Kontroll-Diagrammen nach SAX-basierten Mustern und die Auswertung von Betriebssystem-Aufrufen beschrieben. In [4] wird SAX zur Analyse der Farbverteilung in Bildern verwendet. Der Autor von [223] nutzt SAX für ein Beratungssystem zu Computing-Ressourcen.

Die Fülle an Anwendungen und das zunehmende wissenschaftliche Interesse legen nahe, dass SAX eine vielversprechende neue Datenrepräsentation für Zeitreihen ist, die Potenzial für neue skalierbare Analyseverfahren hat. Die Vorzüge von SAX gegenüber anderen Verfahren in Hinblick auf die Effizienz bei der Durchführung der vorgestellten Analyseaufgaben wird in [276] genauer untersucht. Es wird z. B. die Segmentierung von Zeitreihen mittels SAX mit zwei weiteren Verfahren verglichen: Einem Verfahren namens SWAB [148], welches solange Datenpunkte zusammenfasst, bis eine Kostenschwelle erreicht ist und einem sog. genetischen Algorithmus (GA), welcher durch evolutionäre Suche eine optimale Segmentierung ermittelt. Es zeigt sich, dass durch die Verwendung von SAX eine ungefähr 2,7-fache Beschleunigung dieser Aufgabe erzielt werden kann.

## A.2 Explorative Analyse und interaktive Visualisierung

Eine Analyse großer Mengen teilweise unstrukturierter Daten ist unmöglich, ohne sich im Vorfeld einen Überblick über diese Daten zu verschaffen. Die älteste bekannte Möglichkeit hierzu ist die Zeitreihenvisualisierung als Kurvendarstellung. Es existieren jedoch zahlreiche weitere Ansätze, die alle bei bestimmten Explorationsaufgaben ihre Vorzüge zeigen. In diesem Abs. werden deshalb die besonderen Anforderungen an Zeitreihenvisualisierungen in Kombination mit großen Datenmengen erörtert und die Funktionsweise sowie Vor- und Nachteile verschiedener geeigneter Visualisierungen aufgezeigt.

### A.2.1 Skalierbarkeitsanforderungen

Auch bei der Zeitreihenvisualisierung müssen besondere Skalierungsaspekte berücksichtigt werden, wenn größere Datenmengen dargestellt werden sollen. Dies umfasst einerseits die mit der Datenmenge steigenden benötigten Ressourcen eines Computers und andererseits Probleme bei der Visualisierung einer großen Anzahl an Elementen. Die verschiedenen Teilaspekte werden im Folgenden näher betrachtet.

- **Computer-Ressourcen**
  - **Festplattenspeicher:** Es wird ausreichend Festplattenspeicher benötigt, um große Zeitreihendaten dort zwischenspeichern zu können.
  - **Arbeitsspeicher:** Für eine flüssige Darstellung und Interaktion mit dieser ist es häufig notwendig, anzuzeigende Daten im Arbeitsspeicher zu halten, der hierfür ausreichend dimensioniert sein muss.
  - **Eingabemethodik:** Die Eingabegeräte stellen den Schlüssel für eine effektive und effiziente Interaktion mit der Visualisierung dar. Die Computermaus ist im Visualisierungsbereich noch immer das Standard-Eingabegerät und sollte optimal eingesetzt werden, für spezielle Aufgaben sind jedoch möglicherweise andere Eingabemethoden besser geeignet.
  - **Anzeigesystem:** Natürlich spielt das Anzeigesystem des Computers bei der Zeitreihenvisualisierung eine entscheidende Rolle. Um möglichst viele Details darstellen zu können, sollte das verwendete Display deshalb eine möglichst hohe Auflösung besitzen und auch nicht zu klein sein. Es empfiehlt sich außerdem der Einsatz mehrerer Monitore, so dass mehrere Visualisierungen gleichzeitig bildschirmfüllend angezeigt werden können.
- **Visualisierung vieler Elemente:** Die gängigen Visualisierungswerkzeuge sind nicht für die Darstellung großer Datenmengen gemacht und sind häufig beschränkt auf einige tausend Datenpunkte. Sollen so viele Elemente gleichzeitig dargestellt werden, dass dies auf der be-

reitstehenden Anzeigefläche nicht mehr möglich ist, müssen Informationen bei der Darstellung ausgelassen werden. Dies erfordert besondere Strategien und Interaktionsmöglichkeiten, um eine sinnvolle Exploration sehr großer multivariater Zeitreihen zu ermöglichen.

- **Visuelles Aliasing:** „Unterabtastung“ kann zu unerwünschten Artefakten führen. Fällt die visuelle Abtastrate unterhalb der Nyquistfrequenz, kommt es zu Darstellungsfehlern. Gerade bei stark periodischen Zeitreihen kommt es dann bei der Darstellung zur Anzeige neuer, verzerrter Kurven, die nicht länger der Charakteristik der ursprünglichen Zeitreihe entsprechen (siehe Abbildung A.8).
- **Detailgrad:** Bei der Auslassung zu vieler Einzelwerte geht schlichtweg Information bei der Kurvendarstellung verloren. Beispielsweise werden möglicherweise bedeutsame lokale Extremwerte nicht als Kurvenabschnitt dargestellt, was etwa den falschen Eindruck vermitteln könnte, die Kurve verlaufe stetig weiter. Dies muss durch entsprechende Maßnahmen kompensiert werden.
- **Reaktionszeiten:** Die Geschwindigkeit und Flüssigkeit von Darstellung und Interaktion ist entscheidend für eine effektive, effiziente Exploration. Es ist mit Geschwindigkeitseinbußen zu rechnen, sobald eine kritische Anzahl an Elementen dargestellt werden muss, welche die verfügbaren Ressourcen des Computers voll ausschöpft. Gerade die Interaktivität der Darstellung sollte jedoch unbedingt gewährleistet sein. Vor allem muss es bei kurzer Reaktionszeit möglich sein, flüssig in der Zeitreihendarstellung zu navigieren und den dargestellten Zeitbereich zu verändern (zeitlich zu „zoomen“). Jede Aktion sollte ein sofortiges visuelles Feedback zur Folge haben.

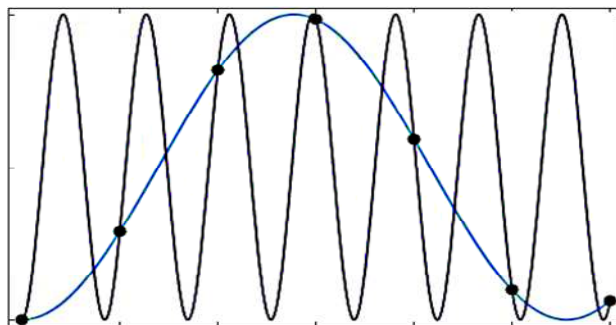


Abbildung A.8: Aliasing bei Unterabtastung sinusähnlicher Zeitreihen.

### A.2.2 Kurvendarstellung

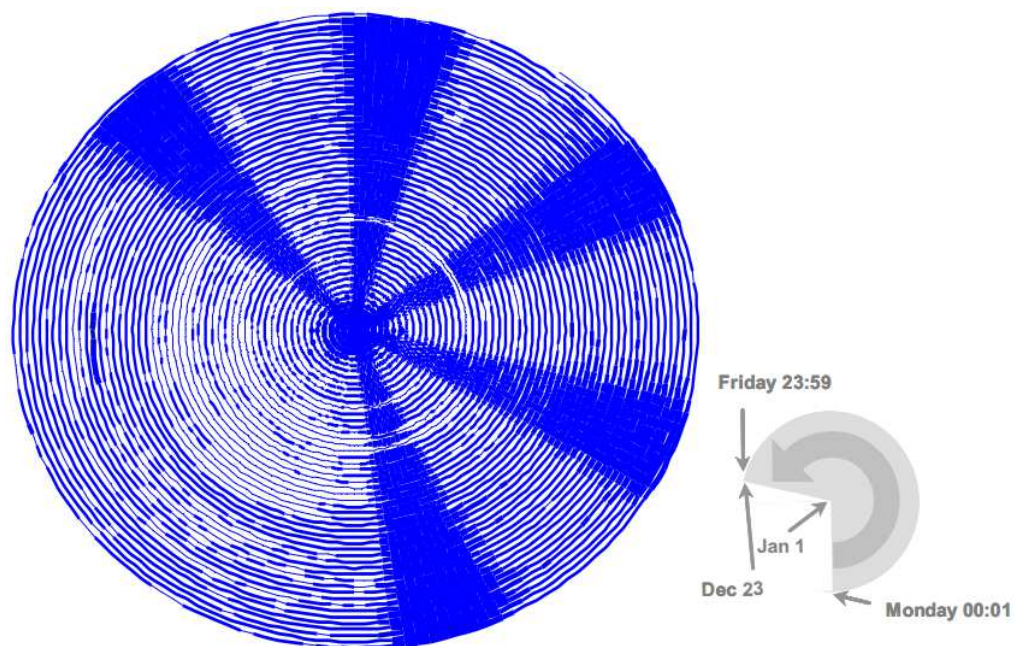
Eine Kurvendarstellung von Zeitreihendaten präsentiert die Datenpunkte als Punkte in einem zweidimensionalen Koordinatensystem. Dabei werden über eine Achse die Positionen der Datenpunkte in der Zeit codiert – hierzu wird meist die horizontale Achse gewählt. Eine weitere Achse codiert die Werte der Datenpunkte – meist ist dies die vertikale Achse. Es handelt sich gemäß [3]

also um eine statische Abbildung von zeitlichen auf räumliche Positionen. Beide Achsen werden für gewöhnlich linear skaliert. Diese Form der Zeitreihenvisualisierung über Positionen zeigt sich für die Darstellung quantitativer Variablen als effizienter als die Codierung durch Farbe oder weitere visuelle Eigenschaften wie Textur, Form oder Orientierung (vergl. [197]). Die Punkte können außerdem auf verschiedene Art verbunden werden, um unterbrechungsfreie Kurven zu erzeugen. Im einfachsten Fall sind die Verbindungen gerade Linien, es werden jedoch auch häufig nichtlineare Verbindungen unter Berücksichtigung benachbarter Werte verwendet. Dies führt zwar zu ansprechenderen Kurvendarstellungen, kann jedoch in vielen Fällen einen falschen Eindruck über die Daten vermitteln, weshalb zumindest optional die einzelnen Datenpunkte sichtbar darstellbar sein sollten. Auch die Skalierung des Darstellungsbereichs kann großen Einfluss auf die Erkennbarkeit der Charakteristik haben. Sollen sehr viele Datenpunkte auf kleinem Raum angezeigt werden, stehen möglicherweise nicht genügend Pixel zu Verfügung, um jeden Datenpunkt anzuzeigen, was dazu führen kann, dass lokale Details der Kurve nicht dargestellt werden.

Für die Darstellung multivariater Zeitreihen gibt es verschiedene Vorgehensweisen. Es kann ein gemeinsamer Wertebereich festgelegt werden oder die einzelnen Kurven werden in unterschiedlichen Wertebereichen angezeigt. Ersteres hat den Vorteil, dass die Kurven in einem absoluten Bezugsrahmen verglichen werden können. Allerdings dürfen sich die einzelnen Wertebereiche nicht zu stark unterscheiden, da ansonsten Kurven, welche einen Großteil des gemeinsamen Wertebereichs einnehmen extrem gestreckt – und solche, deren Wertebereich nur einen kleinen Teil des gemeinsamen Bereichs einnehmen stark gestaucht dargestellt werden, so dass ihre Charakteristik nicht mehr erkennbar ist. Eine Anzeige in separaten Wertebereichen hat den Vorteil, dass Zeitreihen völlig unabhängig von den jeweiligen Wertebereichen, jedoch unter Erhaltung des gemeinsamen Zeitbezugs dargestellt werden, so dass eher ein Vergleich der Charakteristik ermöglicht wird.

### A.2.3 Detektion periodischen Verhaltens

Häufig soll in Zeitreihen ein periodisches Verhalten wie etwa Saisonalitäten erkannt werden. Hierzu wurden spezielle Visualisierungen vorgeschlagen, u. a. Spiraldarstellungen ([50], [318]). Besonders interaktive Spiralkurven mit einstellbarer Periode (z. B. [293], [304]) wurden als effiziente Werkzeuge bei der Detektion von Periodizität betrachtet. Dabei werden eine oder mehrere Zeitreihen als archimedische Spirale dargestellt, wobei die Zeit von innen nach außen verläuft und die Ausprägung der Werte durch Farbe, Strichdicke oder analog zur Kurvendarstellung codiert wird. Ist die Periode gemäß eines vorliegenden Saisonalverhaltens eingestellt, wird dieses sichtbar, da Teile der Spirale mit ähnlichen Merkmalsausprägungen in den gleichen Winkelabschnitten fallen, wie z. B. in Abbildung A.9.



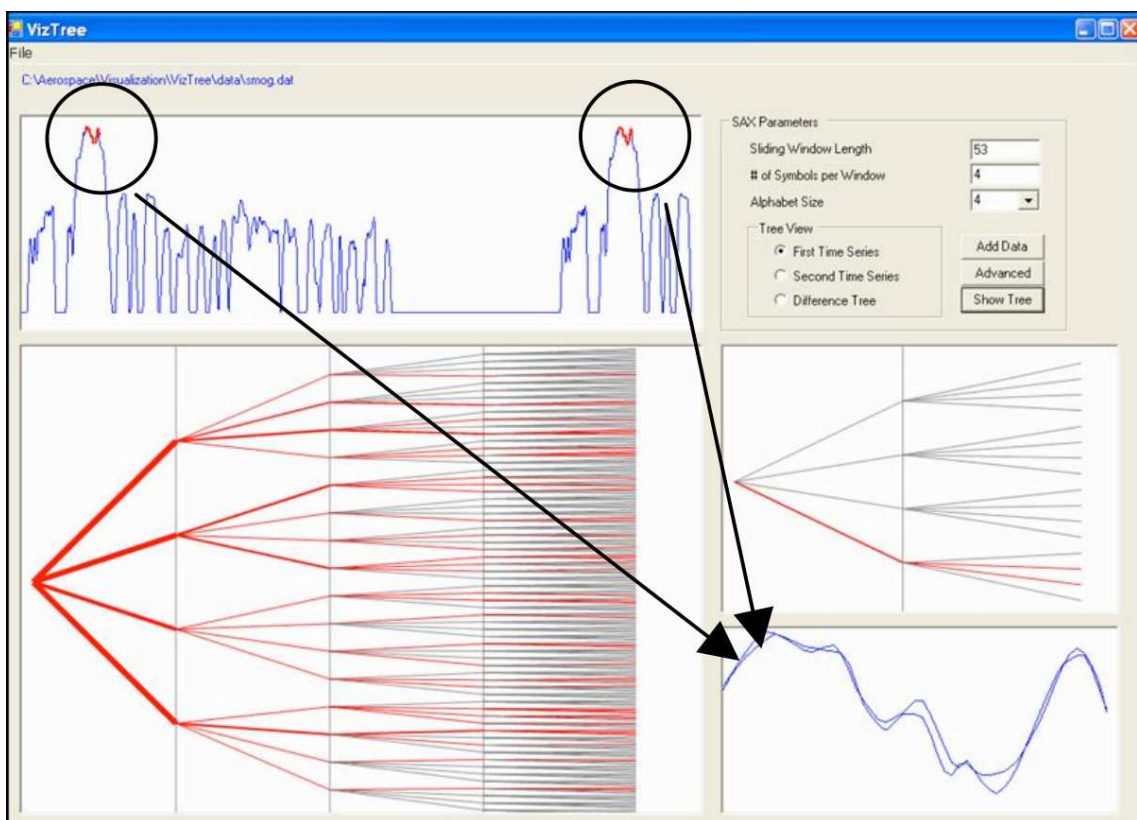
**Abbildung A.9:** SpiralGraph Ansatz nach Weber et al., angewandt auf einen Datensatz zum Stromverbrauch. Quelle: [171].

Die Darstellung benötigt viel Darstellungsplatz und ist daher nur beschränkt skalierbar. Als Variante kann jedoch eine rechteckige Darstellung verwendet werden, bei welcher die Zeit von links nach rechts und von oben nach unten verläuft, so dass ähnliche Werteausprägungen bei entsprechender Einstellung der Periode (hier die Zeilenlänge) direkt übereinander liegen.

#### A.2.4 Strukturanalyse mit VizTree

Auch eine Analyse der Struktur von Zeitreihen kann durch Visualisierungsverfahren unterstützt werden. Jedoch sind verhältnismäßig wenige Methoden bekannt, die über eine spezielle Anwendung hinausgehen und universell für beliebige Zeitreihen einsetzbar sind. Ein Beispiel stellt das 2004 durch Lin et. al. eingeführte Werkzeug *VizTree* [179] dar, dessen Benutzeroberfläche in Abbildung A.10 zu sehen ist. Es basiert auf der SAX-Codierung von Zeitreihen, auf Suffix-Bäumen [320] und auf Ukkonens Algorithmen zur Online-Berechnung von Suffix-Bäumen [302].

Die SAX-Repräsentation wird hier direkt aus einer Zeitreihe berechnet, die aus einer lokalen Datei im CSV<sup>62</sup>-Format geladen werden kann. Die SAX-Parameter für die Online-Codierung können über die Oberfläche rechts oben eingestellt werden. Die SAX-Codierung wird nicht für die gesamte Zeitreihe auf einmal durchgeführt, sondern es wird ein Fensterverschiebungsalgorithmus verwendet. Ein Fenster mit einstellbarer Größe wird sukzessive über der Zeitreihe verschoben und der jeweilige Fensterinhalt wird zu einer SAX-Sequenz codiert, die in einer Liste gespeichert wird, jedoch nur, wenn sie sich von der varangehenden SAX-Sequenz unterscheidet. Es ergibt sich eine Sammlung überlappender SAX-Sequenzen gleicher Länge, deren Häufigkeiten statistisch ausgewertet werden. Die ermittelten Häufigkeiten werden durch die maximal ermittelte Häufigkeit geteilt, wodurch sich Werte im Intervall  $[0,1]$  ergeben.



**Abbildung A.10:** Bildschirmabzug der Software VizTree. Quelle: [171].

Der Namensgebende Suffix-Baum wird in der Oberfläche dargestellt. Dabei repräsentiert jede Verzweigung eine Entscheidung für eines der SAX-Symbole und jeder Weg von der Wurzel zu einem Blatt repräsentiert eine der gesammelten SAX-Sequenzen. Die ermittelten Häufigkeiten

<sup>62</sup> CSV steht kurz für engl. *Comma-separated values* oder auch *character-separated values*. CSV ist ein einfaches textbasiertes Format zur Speicherung tabellarischer Daten, bei dem jede Textzeile einen Datensatz enthält, dessen Felder durch ein spezielles Zeichen getrennt werden, etwa durch Kommata, Semikolons, Tabulatoren etc.

codieren die Strichdicke und Farbe der Verzweigungen des Baums: Häufige Sequenzen werden mit dickeren, seltene mit dünneren roten Verbindungslinien dargestellt und Sequenzen, die überhaupt nicht vorkommen werden grau gefärbt. Durch einen Mausklick auf einen der Äste wird dieser ausgewählt und die entsprechenden Vorkommen in einer Kurvenansicht markiert. Der in Abbildung A.10 gezeigte Baum wurde für ein Alphabet mit vier Symbolen und einer Teilsequenzgröße von ebenfalls vier Symbolen erstellt, so dass 256 verschiedene SAX-Sequenzen dargestellt sind. Da eine Fenstergröße von 53 und eine PAA-Größe von vier gewählt wurden, werden je 53 Werte zu einer Sequenz von vier SAX-Symbolen aggregiert.

VizTree ermöglicht eine einfache visuelle Identifikation der häufigsten Teilsequenz-Muster und visuelle Vergleiche ähnlicher Muster. Jedoch verzweigt sich die visuelle Baumdarstellung mit zunehmender Teilsequenzlänge und Alphabetgröße immer mehr, so dass diese Parameter nur verhältnismäßig klein gewählt werden können.

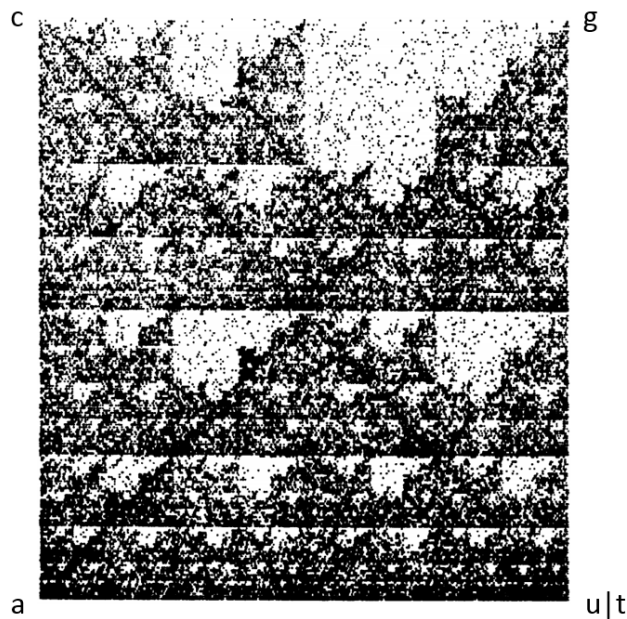
#### A.2.5 Zeitreihenpiktogramme zur Ähnlichkeitsvisualisierung

Die bisher betrachteten Methoden der Zeitreihenvisualisierung beleuchten die Charakteristik einzelner Zeitreihen, wobei jede Darstellung besondere Vorteile bei der Herausstellung bestimmter Merkmale besitzt. Natürlich können anhand solcher Merkmale auch mehrere Zeitreihen miteinander verglichen werden, um beispielsweise Ähnlichkeiten mithilfe der visuellen Darstellungen sichtbar zu machen, aufgrund welcher die Zeitreihen gruppiert werden können. Umso mehr Zeitreihen miteinander verglichen werden sollen, desto kompakter muss dabei die Darstellung jeder einzelnen Zeitreihe sein, was besondere Anforderungen an die Visualisierung stellt. Ein möglicher Ansatz hierzu ist die Repräsentation von Zeitreihen durch kleine Piktogramme (*Icons*), die z. B. aus statistischen Merkmalen abgeleitet werden.

##### A.2.5.1 *Chaos Game Representation*

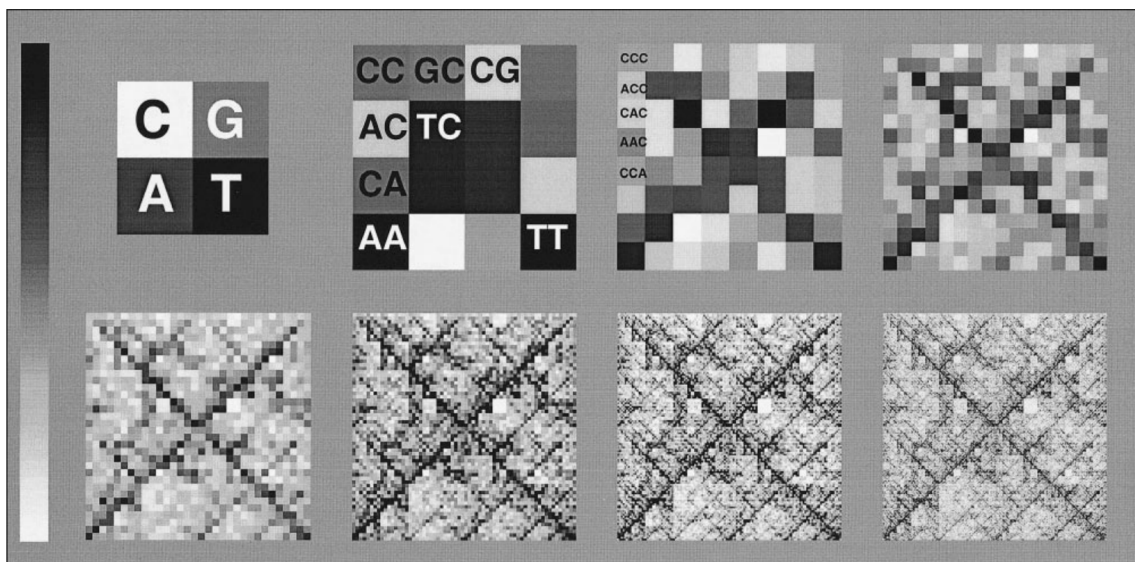
Piktogramme werden z. B. in der sog. *Chaos Game Representation* (CGR) [130] genutzt. Hier werden Symbolsequenzen (ursprünglich DNA-Sequenzen) zweidimensional visualisiert, indem jedem Symbol eines Alphabets eine Position zugewiesen wird und von einem beliebigen Startpunkt ausgehend beim Ablesen der Sequenz eine Linie in Richtung des entsprechenden Symbols gezogen wird, deren Länge dem halben Abstand zu diesem Symbol entspricht. So ergibt sich ein Bild bestehend aus Punkten, dessen komplexe Struktur Aufschluss über lokale und globale Sequenzstrukturen geben soll (siehe Abbildung A.11). Die Methode kann statt auf einzelne Symbole analog auch auf Symbolsequenzen angewandt werden, wobei dann meist die Häufigkeiten aller Teilsequenzen gleicher Länge (oder in der DNA-Sequenzierung z. B. Oligonucleotiden) durch Grauwerte repräsentiert werden (*Frequency Chaos Game Representation*, kurz FCGR, siehe Abbildung A.12).





**Abbildung A.11:** Beispiel einer CGR der menschlichen Beta-Globin Region im elften Chromosom mit 73 357 Basen. Quelle: [130], modifiziert.

CGR zeigte sich in zahlreichen Anwendungen (z. B. [69]) als gut geeignet, um den Grad der Zufälligkeit und die grundlegende Charakteristik langer Sequenzen zu visualisieren. FCGR ermöglicht beispielsweise darüber hinaus auch die Einschätzung kleinerer Sequenzausschnitte, da hieraus generierte FCGR-Bilder starke Ähnlichkeit mit den FCGR-Bildern der Gesamtsequenz aufweisen.



**Abbildung A.12:** Darstellung der Funktionsweise von FCGRs der Länge eins (l. o.) bis acht (r. u.): Die Häufigkeiten für Sequenzen werden als Grauwerte in einer tabellarischen Darstellung codiert. Quelle: [69].

Sollen jedoch für bestimmte Anwendungen bedeutsame Ähnlichkeiten und Unterschiede der Struktur mehrerer Sequenzen herausgestellt werden, zeigen sich auch einige Nachteile der CGR-Methode: Die Darstellung ist komplex und schwer verständlich und ermöglicht nur begrenzt semantische Rückschlüsse [93]. Außerdem wird viel Darstellungsfläche benötigt, wodurch sich CGR ungünstig für eine Gegenüberstellung vieler Zeitreihen zeigt.

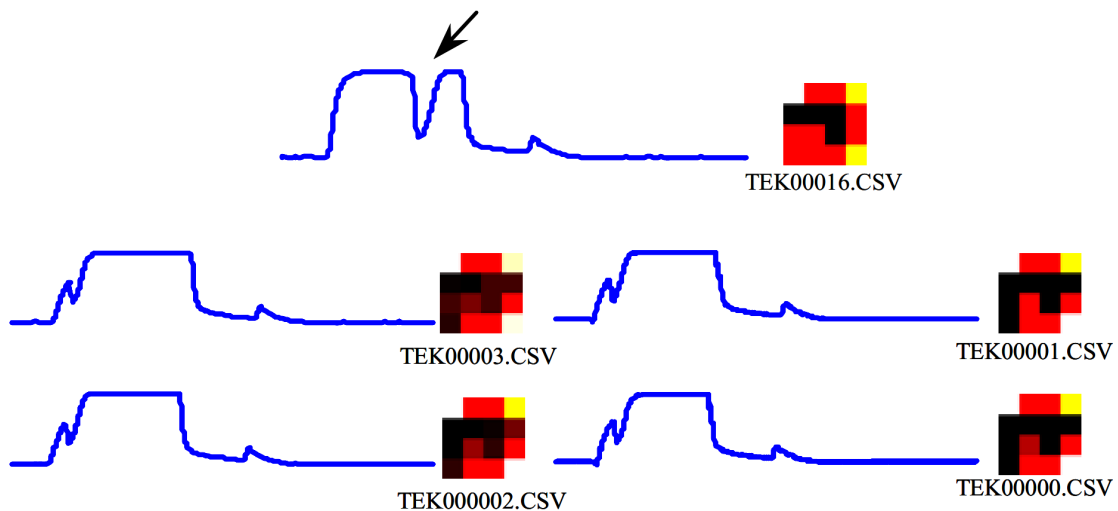
#### A.2.5.2 *Intelligente Icons*

Da Zeitreihendaten in Symbolsequenzen überführt werden können, z. B. durch die vorgestellte SAX-Transformation, wurde versucht, den FCGR-Ansatz auf Zeitreihen anzuwenden. Dies führte zur Idee der *Intelligenten Icons* [144], einer stark aggregierten Kompaktdarstellung von Zeitreihen beliebiger Länge in Form kleiner Bitmap-Bilder. Während in den verbreiteten Betriebssystemen Datei-Icons verwendet werden, um Dateien mit gleicher Dateiendung<sup>63</sup> visuell einheitlich darzustellen und unterscheidbar zu machen, sollen Intelligente Icons dies bezüglich des Dateiinhalts ermöglichen. Bezogen auf Zeitreihendateien sollen also ähnliche Zeitreihen auch ähnliche Datei-Icons erhalten. Die Methode setzt dabei auf die ausgeprägte Fähigkeit des Menschen zur visuellen Mustererkennung und macht eine Sicht auf den Inhalt von Dateien direkt in Dateieexplorern möglich.

Ein Intelligentes Icon stellt in Anlehnung an FCGRs ein Histogramm über die Vorkommenshäufigkeit aller möglichen Symbolsequenzen einer bestimmten Länge für ein bestimmtes Alphabet von Symbolen dar. Zunächst müssen alle Zeitreihen einheitlich in eine symbolische Darstellung überführt werden. Dann werden die Sequenz-Häufigkeiten ermittelt und durch die maximal auftretende Häufigkeit geteilt, um normierte Werte im Intervall  $[0,1]$  zu erhalten. Schließlich werden die Häufigkeiten farblich kodiert und analog zu FCGRs in eine quadratische tabellarische Darstellung aller möglichen Symbolsequenzen eingefügt. Abbildung A.13 zeigt Beispiele für fünf ähnliche Zeitreihen und ihre abgeleiteten Intelligenten Icons. Dabei weist eine der Sequenzen einen kurzen Werteeinbruch auf, der mit einem Pfeil markiert ist. Diese Besonderheit zeigt sich auch in der Icon-Darstellung dieser Kurve, wodurch eine einfache visuelle Gruppierung und die Erkennung von Ausreißern möglich werden. Durch Definition einer Distanzfunktion kann außerdem zu einem Icon automatisch dasjenige mit geringster oder größter Distanz ermittelt werden.

---

<sup>63</sup> Dateiendungen dienen in Betriebssystemen dazu, den Dateityp, bzw. das Datenformat zu repräsentieren, beispielsweise um festlegen zu können, welche Dateien mit welcher Anwendung geöffnet werden sollen. Bei der Darstellung von Dateien als Icons in einem Dateibrowser wird diesen Endungen ein kleines Bild (Icon) zugeordnet, das den Datentyp visuell erkennbar machen soll. Da es jedoch keinen einheitlichen Namensraum für Dateiendungen gibt, kommt es leider häufig dazu, dass für völlig unterschiedliche Datentypen dieselbe Dateiendung und damit dasselbe Icon verwendet wird.



**Abbildung A.13:** Beispiele für Zeitreihenabschnitte und abgeleitete Intelligente Icons. Ein Werteeinbruch unterscheidet eine der Zeitreihen (siehe Pfeil) von allen anderen, was sich in der Icon-Darstellung wiederfindet. Quelle: [170].

## A.3 Data-Intensive Computing und Architekturwahl

Eine Analyse von Zeitreihendaten großen Umfangs und/oder großer Anzahl erfordert zahlreiche Verarbeitungsschritte großer Datenmengen und unterscheidet sich dadurch im Ablauf grundsätzlich von der klassischen Zeitreihenanalyse. Zu besonders datenintensiven Verarbeitungsschritten gehören die Vorverarbeitung der Rohdaten, die Merkmalsextraktion und diverse Transformationen in andere Zeitreihenrepräsentationen aber auch komplexe Abfragen und Suchen. Viele dieser Teilaufgaben müssen typischerweise nur einmalig durchgeführt werden, es gibt jedoch auch Situationen, die eine vollständige Neuverarbeitung sämtlicher Daten erfordern – etwa für die Extraktion neuer als bedeutungsvoll erkannter Merkmale.

Herkömmliche, für sequentielle Verarbeitung ausgelegte Architekturen und Verfahren skalieren jedoch meist nicht linear mit der Datenmenge, so dass die Verarbeitungszeit sowie der Verbrauch an Ressourcen mit der Größe zu verarbeitender Daten oft exponentiell ansteigen. Für sehr große Datenmengen ist deshalb eine sequentielle Abarbeitung enorm zeitaufwendig oder aufgrund fehlender Ressourcen wie z. B. Datenspeicher schlichtweg unmöglich.

Aus diesem Grund ist es in zahlreichen Anwendungsbereichen gängige Praxis, besonders große Datenbestände, in denen prinzipiell neues Wissen verborgen liegen könnte, nicht vollständig, sondern nur ausschnittshaft zu analysieren. Dabei werden die „interessanten“ Ausschnitte meist rein heuristisch ausgewählt oder es werden gar zufällige Abschnitte repräsentativ untersucht. Das Aufstellen und Testen von Hypothesen und Modellen aufgrund ausschließlich exemplarischer Stichproben ist jedoch aus wissenschaftlicher Sicht fragwürdig. Trotzdem wird das komplette

Durchlaufen der Daten in vielen Bereichen weitgehend vermieden, da es – zumindest sequentiell – äußerst aufwendig ist, obwohl dies Voraussetzung zahlreicher Explorations- und Analyseanwendungen ist.

Liegt der Flaschenhals rechnergestützter Datenverarbeitung auf Seiten der Datenmanipulation bzw. der Lese- und Schreiboperationen und nicht – wie in der sog. *rechenintensiven Verarbeitung* – in den Rechenoperationen, so spricht man von *datenintensivem Rechnen* (engl. *Data-Intensive Computing*, kurz DIC) [273], [134]. Unter DIC werden die hierzu entwickelten Methoden und Architekturen für datenparallele Verarbeitung zusammengefasst.

### A.3.1 „Big Data“

Datenbestände, die aufgrund ihrer Größe oder der Komplexität ihrer Struktur nicht oder nur mit enormem Aufwand mit herkömmlichen Methoden und Architekturen verarbeitet – insbesondere analysiert – werden können, werden in aktueller Literatur als „*Big Data*“<sup>64</sup> bezeichnet (z. B. in [45], [342], [316], [286], [76], [84], [329], [65]). Dabei handelt es sich um besonders große Sammlungen von Daten unterschiedlicher und teilweise unbekannter Herkunft und Struktur, welche aufgrund ihrer Gesamtgröße, der Größe einzelner Dateien, der Komplexität der Daten- oder Dateistrukturen und deren Heterogenität nicht mit herkömmlichen Verfahren und Architekturen effizient verwaltet und ausgewertet werden können. Außerdem steht häufig im Voraus nur wenig Information über Inhalt, Struktur und Qualität der Daten zur Verfügung, die für deren Analyse genutzt werden kann. Es bedarf meist einer besonderen verteilten Datenorganisation und -Verarbeitung, um derartige Daten auszuwerten.

#### A.3.1.1 Definition von Big Data

Die populärste Definition von Big Data wurde erstmals 2012 von Gartner vorgeschlagen [323] und bezeichnet Daten als Big Data, welche drei zentrale Aspekte aufweisen – die sog. *3 V der Big Data*:

- *High volume*: engl. für hohes Datenvolumen
- *High velocity*: engl. für hohe Wachstumsgeschwindigkeit
- *High variety* engl. für große Vielfalt (der Datenrepräsentation)

In neuerer Literatur werden häufig zusätzliche Aspekte wie Unsicherheit und Relevanz der Daten mit weiteren Vs betont, so z. B. für engl. *veracity* – Wahrhaftigkeit oder für engl. *value* – Wert

---

<sup>64</sup> Da der Begriff *Big Data* nicht klar definiert ist und außerdem den Größenaspekt auf Dateiebene überbetont, wird neuerdings auch manchmal von *Complex Data* gesprochen.

bzw. Nutzen. Interessanterweise scheinen die verschiedenen Aspekte in unterschiedlichen Disziplinen als unterschiedlich wichtig bewertet zu werden: Im Bereich des Super Computing wird das Datenvolumen betont, im Bereich der Sensornetze und dem Internet der Dinge beispielsweise die Wachstumsgeschwindigkeit, in den Sozial- und Humanwissenschaften und ähnlichen Bereichen der Nutzen der Daten und für das Semantische Netz scheint die Vielfalt von Semantik und Struktur der Daten die größte Herausforderung darzustellen [114].

#### A.3.1.2 *Strukturierte und unstrukturierte Daten*

Eine weitere wichtige Dimension ist die Art der Strukturierung der Daten. Man spricht von *strukturierten Daten*, wenn diese leicht als einzelne Datensätze in einem sog. Datenbankschema – meist in Form von Tabellen – abgebildet werden können und von *unstrukturierten Daten*, wenn diese als Sammlungen von Einzeldateien mit unterschiedlichen Datenformaten vorliegen. Letztere würden eher dem Problemfeld der Big/Complex Data zugerechnet werden, da sie nicht mit herkömmlichen Verfahren verarbeitet werden können. Mischformen, beispielsweise Datenbankeinträge, welche große binäre Rohdaten beinhalten oder auf diese verweisen, werden als teilweise strukturiert bezeichnet.

#### A.3.1.3 *Entwicklung von Volumen und Inhalt*

Seit den ersten digitalen Datensammlungen in den 1970er Jahren wuchs die Gesamtmenge gespeicherter Daten in allen Bereichen kontinuierlich exponentiell an, wobei eine deutliche Beschleunigung des Wachstums seit ca. 2001 zu beobachten ist, welche mit der zunehmenden Verfügbarkeit digitaler Medien mit hoher Speicherkapazität in Zusammenhang gebracht werden kann (siehe Abbildung A.14). Eine detaillierte Untersuchung der weltweiten Entwicklung der Speicherung, Kommunikation und Verarbeitung von Daten, unter Einbeziehung von über 1100 Quellen, findet sich in [111] und im dazugehörigen Appendix. Gemäß Aussagen der International Data Corporation [87] wird ein Wachstum digitaler Daten zwischen 2005 und 2020 auf das dreihundertfache erwartet, von 130 Exabyte auf 40000 Exabyte (mehr als 5200 GB pro Person). Bis 2020 wird eine Verdopplung der Datenmenge alle zwei Jahre prognostiziert.

Handelte es sich bis ca. 1990 noch hauptsächlich um Unternehmensdaten, gewannen nachfolgend durch die rasant zunehmende Internetnutzung die Daten von Internetanwendungen, etwa Multimediadaten wie Videos, Fotos und Texte, an Bedeutung und überschritten ca. 2000 erstmals die Größe der Unternehmensdaten. Interessant ist dabei, dass es sich bei den Unternehmensdaten in der Vergangenheit hauptsächlich um strukturierte Daten handelte, worauf die im Unternehmensbereich eingesetzten Datenbanken und Werkzeuge für Wissensextraktion und Berichte optimiert waren. Aktuell wachsen die strukturierten Daten jährlich um ca. 22 %, unstrukturierte Daten jedoch jährlich um ca. 62 %. Bereits jetzt nehmen die unstrukturierten und teilstrukturierten Daten je nach Quelle in etwa 80 % bis 90 % aller Daten ein (siehe Abbildung A.15, vergl. z. B. [32], [71], [114], [136], [245]).

Die Autoren von [136] nennen einige Beispiele aus Bereichen, für welche aktuell enorme Datenmengen erzeugt werden. Diese sind zusammen mit weiteren Beispielen [109] in

Tabelle A.3 dargestellt.

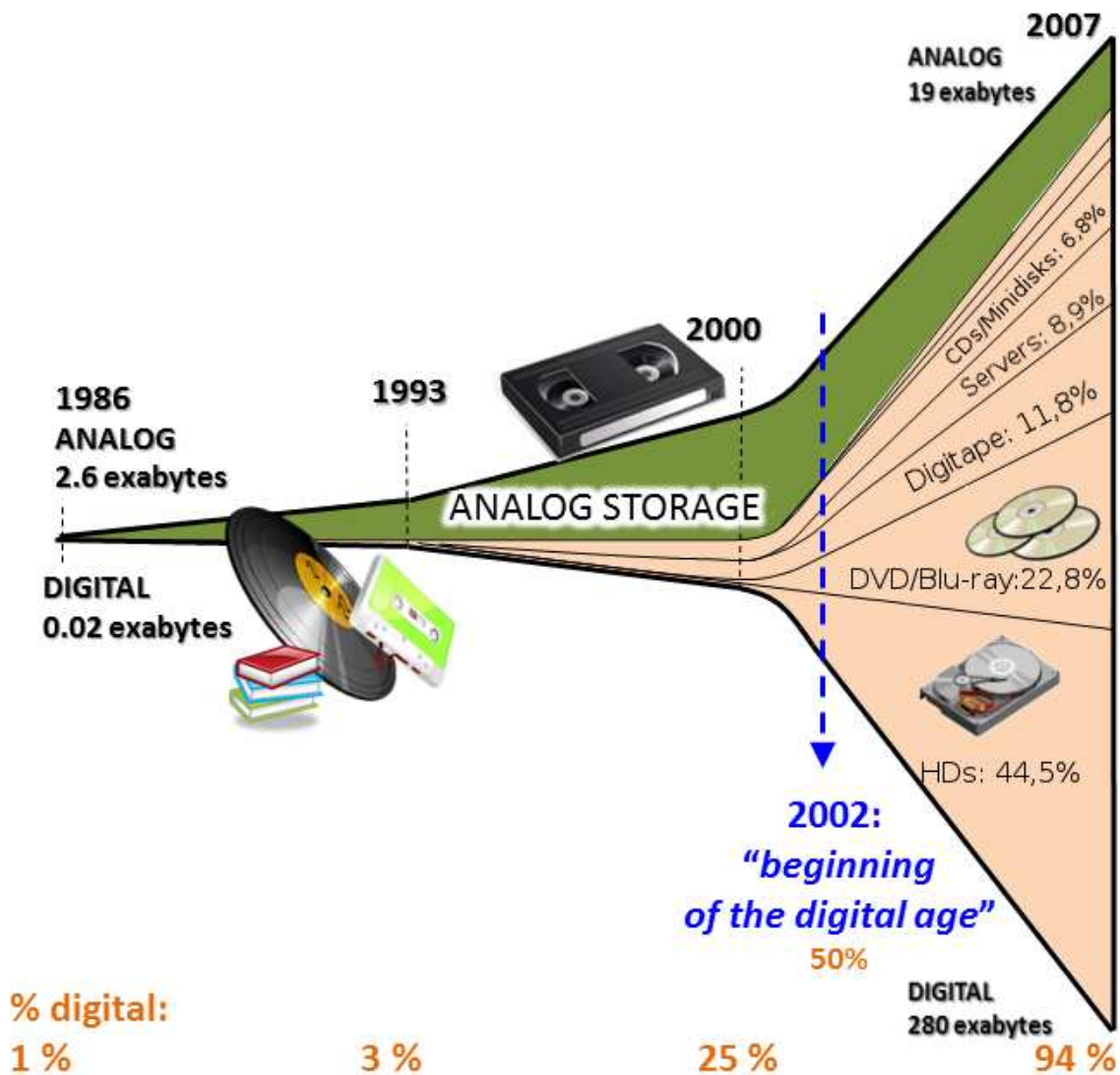
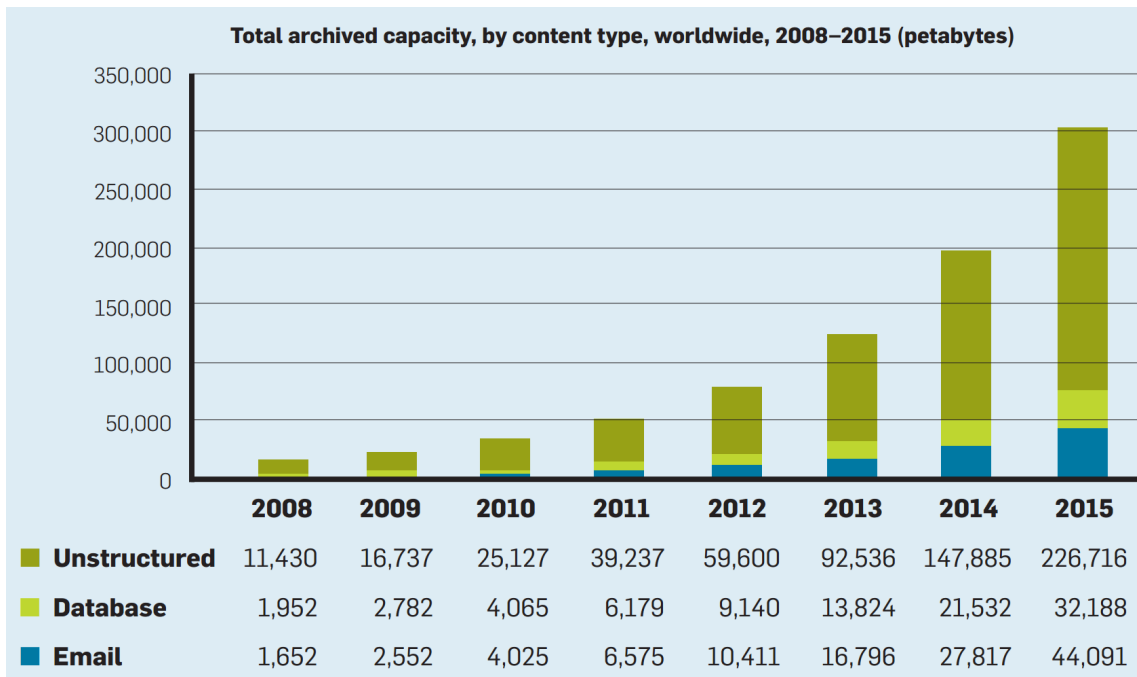


Abbildung A.14: Entwicklung der weltweiten Kapazität für Datenspeicherung 1986-2007, gegliedert nach Medientyp. Quelle: [195], <http://martinhilbert.net> (12.04.2015), modifiziert.



**Abbildung A.15:** Wachstum der Datenmenge weltweit, gegliedert nach inhaltlicher Struktur.  
Quelle: IDC [198].

**Tabelle A.3:** Beispiele für Big Data und deren Wachstum in unterschiedlichen Bereichen (Quellen: [109], [136]).

<i>Daten/Bereich</i>	<i>Beschreibung</i>
<i>Large Hadron Collider (LHC), CERN</i>	50-100 PB jährlich, 20 PB davon wird weltweit verteilt gespeichert und analysiert, jährlich zunehmend
<i>Pan-STARRS Projekt</i>	2.5 PB jährlich
<i>Internetkommunikation (Cisco)</i>	667 000 PB im Jahr 2013
<i>Soziale Netze</i>	über 12 TB an Twitter-Nachrichten täglich (4,38 PB jährlich), zunehmend
<i>Digitale Gesamtdatenmenge</i>	1 700 000 PB im Jahr 2011 7 900 000 PB im Jahr 2015
<i>Bioinformatik</i>	1 GB DNA-Daten pro Genom, 2 TB Mikroskopiedaten pro Tag pro Gerät etc.
<i>British Library UK Website Crawl</i>	~ 110 TB pro Internet-Domain Abbild
<i>Weitere</i>	Millionen von RFIDS, Smart Metern, 4,6 Milliarden GPS-fähige Smartphones, einige Milliarden Digitale Foto- und Videokameras (Flickr, Youtube, etc.), Open Street Maps, Wikipedia ...

#### A.3.1.4 Archivierung digitaler Daten

Durch die digitale Technik ist es heute möglich, immer mehr und immer größere Daten zu speichern, was zu dem beschriebenen rasanten Zuwachs der internationalen Gesamtdatenmenge führt. Doch während die speicherbare Datenmenge ständig zunimmt, ist dies leider nicht der Fall für deren Verarbeitbarkeit und auch nicht für die Maximaldauer der Speicherung. Festplatten, die hauptsächlich Speichermedien für digitale Daten sind für eine Betriebsdauer von etwa fünf Jahren vorgesehen, Magnetbänder, die einzige Alternative, welche momentan für die Massenspeicherung großer Datenmengen zur Verfügung steht, haben eine etwaige Lebensdauer von zehn Jahren. Optische Medien wie DVDs und CDs werden nach nur ca. 20 Jahren unlesbar und haben nicht die erforderliche Speicherkapazität. Zusammenfassend kann man sagen, dass die gesamte Speicherkapazität zunimmt, während die Haltbarkeit der Speichermedien abnimmt. Momentan wird dies dadurch kompensiert, dass zu archivierende Daten alle zehn bis zwanzig Jahre auf neue Speichermedien kopiert werden, ein Vorgang, der mit zunehmender Datenmenge ebenfalls einer Skalierungsproblematik unterliegt. Macht man sich im Vergleich dazu die Haltbarkeit von Büchern oder gar gravierter Steintafeln etc. bewusst, wird klar wie flüchtig und unsicher digital gespeicherte Daten sind und wie unwahrscheinlich es ist, dass diese ein extremes Ereignis wie etwa eine weltweite Katastrophe überleben [245].

#### A.3.1.5 Langzeitarchivierung in unterschiedlichen Bereichen

Sicherlich ist es nicht für alle Bereiche gleichermaßen notwendig, sämtliche Daten über lange Zeiträume zu archivieren und verfügbar zu halten. Im unternehmerischen Bereich reicht es oftmals aus, Zusammenfassungen historischer Daten vorzuhalten, solange eine revisions sichere Buchführung möglich ist, wie sie im Handelsgesetzbuch geregelt ist. Andererseits gibt es auch Einschränkungen für die Datenspeicherung, z. B. durch § 28 des Bundesdatenschutzgesetzes. Insbesondere in den Wissenschaften ist jedoch eine Langzeitarchivierung häufig notwendig und sogar teilweise vorgeschrieben<sup>65</sup>, um eine Nachvollziehbarkeit und Überprüfbarkeit von Folgerungen und Analysen aufgrund der Ursprungsdaten im Sinne guter wissenschaftlicher Praxis zu gewährleisten. Dies ist natürlich für große Datenmengen extrem aufwendig und teuer und wird daher häufig vernachlässigt.

---

<sup>65</sup> Die Vorschriften zur Langzeitarchivierung wissenschaftlicher Daten sind national unterschiedlich. Für Deutschland gibt es bisher keine gesetzlichen Vorgaben, lediglich Handlungsempfehlungen. Z. B. hat die DFG bereits 1997 „Vorschläge zur Sicherung guter wissenschaftlicher Praxis“ veröffentlicht, worin sich u. a. folgende Empfehlung findet: „Primärdaten als Grundlagen für Veröffentlichungen sollen auf haltbaren und gesicherten Trägern in der Institution, wo sie entstanden sind, für zehn Jahre aufbewahrt werden“ [70].



#### A.3.1.6 Einordnung der Stromnetz-Messdaten des KIT

Auch die Analyse der am KIT gemessenen hochfrequenten Stromnetz-Messdaten fällt in das Big Data Problemfeld. Bei diesen Daten handelt es sich um große Sammlungen komplexer, teilweise unstrukturierter Zeitreihendaten und zugehöriger Metadaten. Diese Daten stammen jeweils aus örtlich verteilten Messungen und liegen in einzelnen Dateien unterschiedlichen Formats in einer flachen Ordnerhierarchie vor, wobei sich Formate und Ordnerstruktur über die Zeit verändert haben. Außerdem sind die Datenpunkte der multivariaten Zeitreihen nicht exakt äquidistant sondern besitzen jeweils eigene Zeitstempel. Eine Speicherung dieser Daten in einer Datenbank würde keine skalierbare Verarbeitung zulassen, die Daten sollen jedoch in großem Umfang analysiert werden.

Es wurden daher Möglichkeiten für die Verarbeitung großer Datenmengen untersucht, die eine effiziente Verarbeitung in akzeptabler Zeit versprechen und insbesondere auf Zeitreihendaten anwendbar sind.

#### A.3.2 Datenparallele verteilte Verarbeitung und Datenlokalität

Ohne eine Berücksichtigung der Datenlokalität, also des genauen Speicherorts, bei der Verteilung können große Datenmengen nicht effizient datenparallel verarbeitet werden, ohne diese zuvor auf lokale Speichermedien<sup>66</sup> der Verarbeitungsknoten durch sog. *Staging*<sup>67</sup> zu übertragen, was einen beträchtlichen zusätzlichen Zeitaufwand bedeutet. Jede parallel arbeitende Recheninstanz benötigt schnellen – und nach Möglichkeit exklusiven – Datenzugriff, damit sich die Zugriffe nicht gegenseitig ausbremsen. Dies kann optimal durch eine lokale Persistenz der Daten ermöglicht werden. Es existieren jedoch auch Ansätze, welche schnelle zentrale Speicher einsetzen, die bis zu einer kritischen Datenmenge skalierbar sind, jedoch nicht darüber hinaus.

**Skalierung:** In diesem Zusammenhang müssen zwei Arten der Skalierung von Ressourcen unterschieden werden:

---

<sup>66</sup> Meist werden aufgrund ihrer hohen Kapazitäten, Langlebigkeit und kostengünstigen Verfügbarkeit immer noch magnetische Speichermedien („Festplatten“) zur verteilten nichtflüchtigen Speicherung großer Datenmengen eingesetzt. Zunehmend kommen jedoch – v. a. für geschwindigkeitskritische Anwendungen – auch Halbleiterlaufwerke (engl. Solid-State-Disks, kurz SSDs) zum Einsatz, da diese schnellere Zugriffszeiten aufweisen.

<sup>67</sup> Man nennt den Vorgang des Ladens und der temporären Platzierung der Daten für (verteilte) Berechnungen *Staging*. Im Bereich des High-Performance-Computings, welches typischerweise eher für rechenintensive als für datenintensive Verarbeitung eingesetzt wird, ist ein solches Staging üblich und dauert oft ähnlich lange wie die eigentlichen Berechnungen.

*Vertikale Skalierung* bezeichnet eine Leistungssteigerung durch Hinzufügen von Ressourcen zu einer Recheneinheit, also etwa die Erhöhung der Taktfrequenz des Prozessors, die Erweiterung des Hauptspeichers etc. Im Englischen wird oft von *scale up* gesprochen.

*Horizontale Skalierung* bezeichnet hingegen eine Leistungssteigerung durch Hinzufügen zusätzlicher (gleichwertiger) Recheneinheiten und wird im Englischen oft *scale out* genannt.

Die vertikale Skalierung ist durch den Stand der Technik beschränkt und erfordert meist spezielle Hochleistungsrechner, während horizontaler Skalierung prinzipiell keine Grenzen gesetzt sind. Die mit der Anzahl eingesetzter Rechenknoten zunehmende Komplexität des Systems und die Problematik schlechter Parallelisierbarkeit vieler Verarbeitungsvorgänge sind jedoch Nachteile der horizontalen Skalierung und müssen durch eine intelligente Verarbeitungssoftware gesteuert werden.

#### A.3.2.1 *Abgrenzung zu datenparalleler Verarbeitung ohne Datenlokalität und Ad-hoc-Systemen für HPC-Cluster*

Es existieren zahlreiche Ansätze, die versuchen, zentrale, von den Rechenknoten gemeinsam verwendete Speichersysteme zu verwenden. Diese werden durch Mechanismen wie *Datenpufferung* (engl. *Caching*) auf Geschwindigkeit optimiert, so dass für kleinere bis mittelgroße Cluster ähnliche parallele Lese- und Schreibgeschwindigkeiten möglich sind wie bei Ansätzen, die mit Datenlokalitätsprinzip arbeiten.

Vergleicht man jedoch die Skalierbarkeit beider Ansätze, so wird die Datenmanipulation bei zentralistischen Ansätzen mit zunehmender Anzahl gleichzeitig zugreifender Verarbeitungsknoten immer langsamer. Das Datenlokalitätsprinzip erlaubt hingegen eine prinzipiell unbegrenzte horizontale Skalierung.

Es gibt jedoch auch Vorteile datenparalleler Verarbeitung mit zentralen Speichersystemen. Viele Rechenzentren beinhalten bereits Clustersysteme für *hochperformantes Rechnen* (engl. *High-Performance Computing*, kurz HPC), da diese und die entsprechenden rechenintensiven Anwendungen eine wesentlich längere Tradition aufweisen als die neueren datenintensiven Anwendungen (v. a. die Big Data Analyse). Auf diesen HPC-Clustern lässt sich meist ein Ad-hoc-System für verteilte datenintensive Verarbeitung einrichten, sofern das vorhandene zentrale Dateisystem genutzt wird. Hierdurch lassen sich zum einen die Kosten für die Einrichtung eines zusätzlichen Clustersystems vermeiden und zum anderen ist es möglich, rechenintensive und datenintensive Verarbeitung zu kombinieren. Im Einzelfall und für bestimmte Anwendungen mag deshalb eine solche Kombination sinnvoll sein. Eine echte Skalierbarkeit mit der zu verarbeitenden Datenmenge ist jedoch nicht gewährleistet und der Cluster muss, sobald eine Anwendung die Skalierungsgrenzen sprengt, komplett neu eingerichtet werden.

Obwohl die für diese Arbeit untersuchten Datenmengen aktuell noch keine Größe aufweisen, für welche derartige Ad-hoc-Systeme an ihre Grenzen stoßen, wird hier ein Fokus auf die Skalierbarkeit für große Zeitreihendaten gelegt. Eine datenparallele Verarbeitung unter Berücksichtigung der Datenlokalität ermöglicht auch zukünftige Anwendungen zur Zeitreihenanalyse, für welche wesentlich größere Datenmengen zu erwarten sind. Für die untersuchten EDR-Messdaten ist beispielsweise ein schneller Zuwachs der Anzahl an gleichzeitig verteilt messenden EDR-Geräten zu erwarten, da nur so die komplexe Dynamik in Stromnetzen vollständig erfasst werden kann. Aus diesen Gründen stehen die Möglichkeiten von Verbundsystemen nicht im Fokus der Arbeit.

#### A.3.2.2 *Abgrenzung zu datenparalleler Verarbeitung für mittelgroße Datenmengen*

Datenparallele Verarbeitung mittelgroßer Datenmengen, welche nicht zwingend eine verteilte datenparallele Verarbeitung auf einem Cluster erfordert, wird problematischer Weise vonseiten der Industrie oft ebenfalls als Big Data Problem bezeichnet. Aufgrund der fehlenden Berücksichtigung der Skalierungs- und Schemaeinschränkungen kommt es deshalb nicht selten zu Missverständnissen hinsichtlich der Einsetzbarkeit solcher Systeme für DIC.

Parallele In-Memory Verarbeitung erlaubt beispielsweise die schnelle Analyse größerer Datenmengen innerhalb einer Datenbank, ohne den laufenden Produktionsbetrieb zu stören. Ein weiteres Beispiel stellen parallele Datenbank-Management-Systeme (DBMS) dar, die es bereits seit den 1980er Jahren gibt. Hinsichtlich der Skalierbarkeit, der Anforderungen an Ressourcen und vor allem der Notwendigkeit eines strukturellen Datenschemas unterscheiden sich derartige Ansätze jedoch grundlegend von der verteilten datenparallelen Verarbeitung von Big Data im Bereich des DIC. Die zu verarbeitenden Daten liegen meist in strukturierter Form (v. a. als Zeilen und Spalten von Tabellen) innerhalb einer Datenbank vor. Damit fallen sie nicht in das Problemfeld der komplexen Analyse großer, teilweise unstrukturierter, heterogener Daten. In [232] zeigen die Autoren für einen Cluster aus 100 Rechenknoten, dass sich parallele DBMS für einige bestimmte Verarbeitungsoperationen auf strukturierten Daten performanter zeigen als eine vergleichbare Implementierung im MapReduce Programmiermodell (siehe Abs. 0). Dies ist jedoch erst nach der Überführung der Daten in ein relationales Schema der Fall, was für große, komplexe Daten einen erheblichen Aufwand darstellt. Die Verarbeitungszeit, die für eine solche Schemaüberführung aufgewendet werden muss übersteigt oftmals die Dauer der nachfolgenden Analysen. Deshalb ist die Einschätzung der Autoren nicht verallgemeinerbar auf große Daten beliebiger heterogener Struktur.

Die Verarbeitung in parallelen DBMS ist für große Zeitreihen im speziellen deshalb ungeeignet, da die Speicherung in Tabellenform je Zeile nur einen Datenpunkt mit entsprechendem Zeitstempel ermöglicht. Bei einem Lesevorgang einer Zeitreihe mit Millionen von Datenpunkten wären dementsprechend auch Millionen einzelner Leseoperationen notwendig, was schnell zu einer

Überlastung der Datenbank führen könnte. Häufig liegen Zeitreihendaten in kodierter Form als Einzeldateien für einen bestimmten Zeitraum vor und müssen als solche verarbeitet werden.

#### A.3.2.3 Datenparallele Verarbeitung mit Datenlokalitätsprinzip

Ein vielversprechender Ansatz für skalierbares DIC funktioniert so, dass sämtliche Daten verteilt auf den lokalen Speichermedien mehrerer Rechenknoten persistieren und so keine Eingangsdaten mehr zur auswertenden Recheneinheit transportiert werden müssen. Der auszuführende Programmcode wird stattdessen zu jedem Rechner transportiert, auf welchem ein relevanter Datenblock lokal gespeichert ist. Eine redundante Speicherung auf mehreren Rechnern ermöglicht über die verteilte Verarbeitung hinaus eine datenparallele Verarbeitung derselben Datenblöcke auf unterschiedlichen Rechenknoten.

So müssen lediglich die (Teil-)Ergebnisse der Berechnungen jedes beteiligten Rechenknotens zusammengeführt und übertragen werden. Ein großer Vorteil dieses Ansatzes ist, dass nicht zwangsweise teure Spezialhardware benötigt wird, sondern dass ein Verbund aus konventionellen Computern bereits als leistungsfähiges, skalierbares System eingesetzt werden kann.

Eine datenparallele, verteilte Verarbeitung großer Datenmengen bedarf besonderer Mechanismen zur Datenorganisation und zur Steuerung des Programmablaufs. Beispielsweise stellt das von Google Inc. eingeführte Programmiermodell *MapReduce* [66], [67] eine bereits etablierte Möglichkeit zur Umsetzung des Programmablaufs dar. Dieses Prinzip wird z. B. auch in der quelloffenen und weit verbreiteten Implementierung *Hadoop* [322] eingesetzt. Zusammen mit dem Hadoop Dateisystem HDFS (von engl. Hadoop file system) ermöglicht es die grundlegende Funktionalität verteilter datenparalleler Verarbeitung großer Datenmengen. Unzählige Softwareprojekte bauen auf dieser Basis und ihrer Weiterentwicklungen auf und stellen diverse Schnittstellen sowie Programmier- und Anfragesprachen auf hohem Abstraktionsniveau bereit. Diese erlauben eine vereinfachte Benutzung der komplizierten Architekturen durch Analysten, bei welcher die durch Datenlokalität, verteilte Verarbeitung und Datenparallelität hervorgerufene Komplexität weitgehend transparent bleibt.

### A.3.3 Hadoop

*Hadoop* ist ein auf Java basierendes quelloffenes Softwaresystem zur skalierbaren verteilten Verarbeitung großer Datenmengen auf Clustersystemen. Es besteht aus zwei grundlegenden Komponenten: Einem speziellen verteilten Dateisystem namens *HDFS* (von engl. *Hadoop File System*) und einem Programmiermodell für die verteilte datenparallele Verarbeitung namens *MapReduce*.

Auf dieser Basis ermöglicht Hadoop die Implementierung von Anwendungen, welche große Mengen an teilweise unstrukturierten Daten verteilt auf tausenden von Rechenknoten verarbeiten

können. Das System ist optimal für eine horizontale Skalierung über die Anzahl an Rechenknoten geeignet, erlaubt jedoch ebenso eine vertikale Skalierung über die Leistungsfähigkeit der einzelnen Knoten. Typische Datengrößen, für die Hadoop in der Praxis klare Vorteile gegenüber anderen Verarbeitungsansätzen bietet, reichen deshalb je nach Konfiguration des Clusters von einigen Gigabyte über mehrere tausend Terabyte. Theoretisch können jedoch durch die horizontalen Skalierungsmöglichkeiten unbegrenzt große Datenmengen verarbeitet werden.

#### A.3.3.1 *Entstehungsgeschichte*

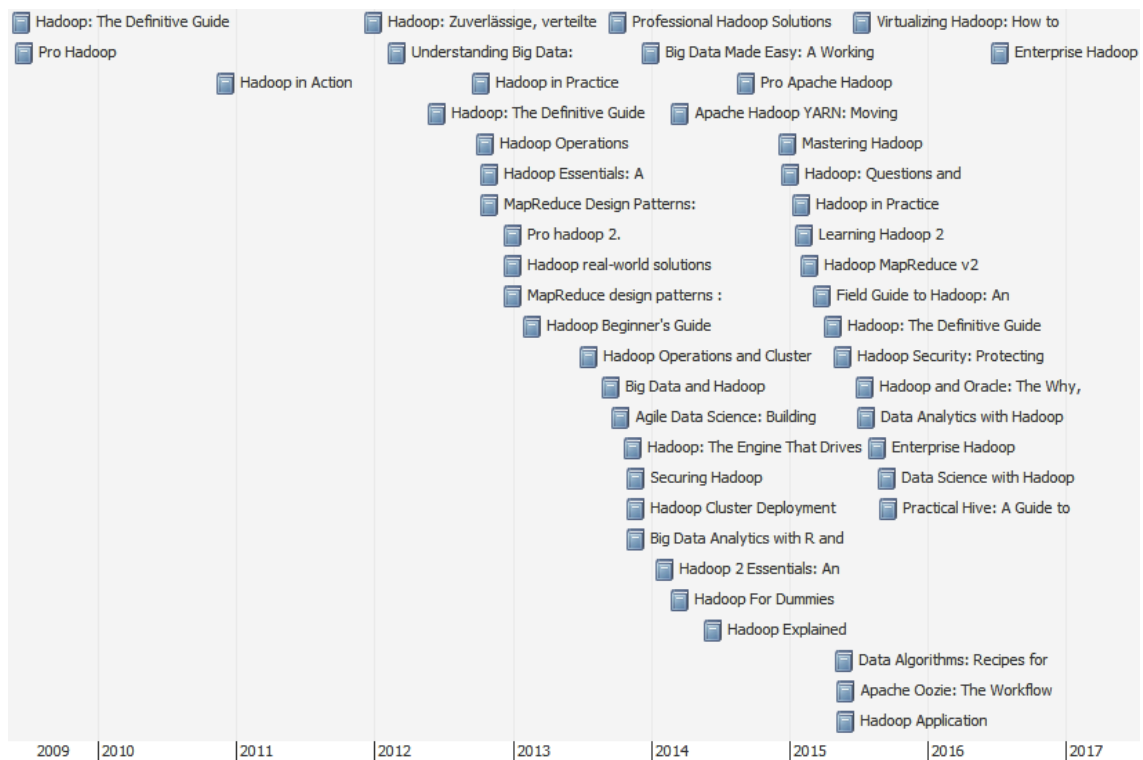
Die Ursprünge der Software reichen zurück in das Jahr 2

002, als Doug Cutting und Mike Cafarella die Open-Source-Software Nutch [156], [10] entwickelten, eine skalierbare Suchmaschine als Antwort auf das proprietäre System von Google Inc. Als Google 2003 das horizontal skalierbare verteilte Dateisystem GFS (von engl. Google File System) [91] und 2004 das dazu passende Programmiermodell für skalierbare verteilte Datenverarbeitung Google MapReduce [66] veröffentlichte, wurde die grundlegende Funktionalität aufgegriffen und bildete die Basis für ein generischeres Verarbeitungssystem – das in Java implementierte Hadoop. Nutch wurde schließlich als Hadoop-Anwendung neu implementiert [104].

Seit der ersten Implementierung wurde das Potential von Hadoop zur Lösung von Skalierbarkeitsproblemen von einer zunehmend wachsenden Nutzer- und Entwicklergemeinde entdeckt, was zur schnellen Verbreitung und zu zahlreichen Erweiterungen und Verbesserungen führte. Die Softwareentwicklung wird durch große kommerzielle Anwender, wie Intel, Facebook, Amazon und Yahoo, aber auch zahlreiche mittelständische Betriebe und öffentliche Einrichtungen wie Universitäten und Bibliotheken stark vorangetrieben ([240] enthält eine stets aktuell gehaltene Liste von Institutionen).

Eine schier unüberschaubare Anzahl an Softwareschichten baut mittlerweile auf den Kernkomponenten von Hadoop auf und definiert ein erweitertes „Software-Ökosystem“, welches die Entwicklung verteilter Anwendungen stark vereinfacht und die Komplexität der Verteilung und Parallelität für Entwickler weitgehend transparent macht. Hierdurch wird die Nützlichkeit und Einsetzbarkeit von Hadoop enorm gesteigert, da sich die Entwicklungszeit von Anwendungen durch die unterschiedlichen Schnittstellen deutlich verkürzt. Die bewährtesten Erweiterungen werden zunehmend in die Distributionen von Hadoop integriert, was eine Installation wesentlich vereinfacht. Die Wörterwolke in Abbildung A.16 zeigt einen Überblick über Begriffe, die häufig in





**Abbildung A.17:** Zeitstrahl zu Bucherscheinungen über Hadoop (Titel teilweise gekürzt, enthält eine angekündigte Erscheinung für 2016).

Quelle: Eigene Darstellung, generiert mit Zotero<sup>68</sup>, modifiziert.

Während zunächst verhältnismäßig wenig gute Dokumentation existierte – abgesehen von einigen Whitepapers und der Dokumentation des Hadoop-Quellcodes – nimmt die Zahl der Buch-Neuerscheinungen Hadoop-spezifischer Literatur noch immer zu, wie aus dem Zeitstrahl in Abbildung A.17 ersichtlich wird. Da keines der Bücher mehr auch nur annähernd alle Aspekte und Anwendungsmöglichkeiten des komplexen und ständig wachsenden Hadoop-Ökosystems beinhalten kann, sei des Weiteren auf die zahlreichen Online-Ressourcen der einzelnen Projekte verwiesen ([287] listet die meisten freien Open-Source Software-Erweiterungen und Schnittstellen in thematischer Gruppierung). Des Weiteren wurden unzählige wissenschaftliche Veröffentlichungen publiziert, die unterschiedliche Aspekte und Anwendungen von Hadoop beleuchten.

### A.3.3.3 Architektur

Hadoop benötigt nicht unbedingt eine spezielle Serverarchitektur sondern lässt sich auf einfacher Standardhardware einrichten. Hierdurch ist die Umsetzung eines Clusters für verteilte Datenverarbeitung sehr kostengünstig und erstmals auch für kleinere Forschergruppen, Unternehmen und Privatanwender möglich, was sicherlich zur Verbreitung von Hadoop beigetragen hat. Die Verwendung von Hochleistungsrechnern ist jedoch ebenso möglich und für manche spezielle rechenintensive Anwendungen wie die wissenschaftliche Analyse großer Mess- und Simulationsdaten sicherlich empfehlenswert.

Mehrere eigenständige Computer bilden einen *Hadoop-Cluster*, der weitgehend einer sog. *Shared Nothing*-Architektur folgt, in welcher jeder Rechenknoten seine eigenen Ressourcen besitzt und keine zentralen Ressourcen geteilt werden. Lediglich die Koordination wird von speziellen Instanzen übernommen, welche etwa einen Index vorhalten, der angibt, auf welchen Datenknoten welche Daten gespeichert sind oder den Ablauf und die Verteilung von Verarbeitungsprozessen koordinieren. Es kann prinzipiell zwischen solchen sog. *Namensknoten* und sog. *Datenknoten* unterschieden werden, welche Daten lokal speichern und verarbeiten. Alle eingesetzten Rechner werden typischerweise unter einem Linux-Betriebssystem betrieben, wobei über Umwege nahezu alle Betriebssysteme eingesetzt werden können.

**Der Namensknoten** (engl. Schreibweise: *NameNode*) verwaltet Metadaten über alle ihm bekannten Datenknoten und die dort gespeicherten Daten. Er verwaltet und aktualisiert den *Namensraum*, die Gesamtheit des Speicherplatzes, die verteilten Speicherorte der Datenblöcke von Dateien und Verzeichnissen und deren Eigenschaften wie Zugriffsrechte und weitere Dateiattribute in einer Liste, die als *Image-Datei* im RAM gehalten wird. Diese wird regelmäßig im lokalen Dateisystem abgelegt und als *Checkpoint* bezeichnet. Eine sog. *Journal-Datei* enthält außerdem alle Information über aktuell stattfindende Dateioperationen. Auf dem Namensknoten selbst werden meistens keine Nutzerdaten gespeichert. Er bearbeitet alle Anfragen entfernter Clients und initiiert Datentransfers, indem er Datenströme von und zu relevanten Datenknoten vermittelt.

**Ein zweiter Namensknoten** (engl. *Secondary NameNode*) unterstützt den Namensknoten, indem regelmäßig die Liste mit Dateiblöcken und die Journal-Datei des Namensknotens miteinander abgeglichen und bereinigt werden. Nach einem Komplettausfall des Namensknotens können die Metadaten des zweiten Namensknotens geladen werden, um die Konsistenz der Daten zu erhalten.

**Datenknoten** (engl. *DataNodes*) empfangen, speichern, senden und verarbeiten lokale Daten als Blöcke fixer Größe (siehe Abs. 0 zu HDFS). Diese Rechner werden auch manchmal „Arbeiter“ genannt. Möchte man den Cluster bei wachsender Datenmenge horizontal skalieren, installiert man weitere Datenknoten. Die Datenparallelität und somit meist auch die Verarbeitungsgeschwindigkeit kann erhöht werden, wenn die redundante Replikation von Datenblöcken erhöht wird, was auch zu einer verbesserten Ausfallsicherheit führt. Eine vertikale Skalierung ist ebenfalls möglich, indem die Kapazität lokaler Ressourcen ausgebaut wird, also beispielsweise Standardhardware durch Hochleistungsrechner ersetzt wird, um schnellere Rechenprozesse zu ermöglichen.

Entscheidend für die Leistungsfähigkeit des Gesamtsystems ist die Art der Vernetzung aller Knoten, welche stets einen schnellstmöglichen Datenverkehr über das Netzwerk ermöglichen sollte.



#### A.3.3.4 HDFS – das Hadoop Dateisystem

Das Hadoop-eigene verteilte Dateisystem HDFS ist darauf ausgelegt, auf Standardhardware ein skalierbares, fehlertolerantes und leistungsfähiges Dateisystem speziell für große Datenmengen bereitzustellen. Es ist auf hohen Datendurchsatz für große Datenmengen optimiert. Um eine schnelle Datenübertragung von und zu entfernten Clients zu ermöglichen, setzt HDFS auf lokal gepuffertes Streaming. Dies führte dazu, dass einige POSIX<sup>69</sup>-Anforderungen im ursprünglichen Entwurf ([38]) zugunsten einer Durchsatzoptimierung nicht vollständig umgesetzt wurden. HDFS wurde ursprünglich als Dateisystem der quelloffenen Web-Crawling-Software *Nutch*<sup>70</sup> entwickelt, welche zu Hadoop ähnliche Anforderungen an Skalierbarkeit und Ausfallsicherheit aufweist.

Wie die meisten Dateisysteme besitzt HDFS einen hierarchischen Namensraum mit Dateien und Ordnern. Jedoch findet die Ablage von Dateiinhalten blockweise statt, wobei die Blockgröße für jede Datei separat gewählt werden kann. Typische Blockgrößen sind 128 MB, 64 MB oder 32 MB, wodurch ersichtlich wird, dass HDFS auf Dateigrößen ausgelegt ist, die jenseits dieser Blockgrößen liegen. Datenblöcke werden durch Replikation stets redundant auf mehreren Datenknoten gespeichert. Dies erhöht die Ausfallsicherheit, jedoch auch die mögliche Datenparallelität bei der Verarbeitung. Meist wird der Standardreplikationsfaktor von drei verwendet. Die Verwaltung des Namensraums mit Ordnerhierarchie und Zuordnung der Dateipfade zu physischen Speicherorten der Blöcke auf den Datenknoten übernimmt der Namensknoten, während die Datenknoten die Nutzlast speichern.

Clients kontaktieren zum Lesen einer Datei den Namensknoten. Dieser liefert eine Blockliste zurück, welche nach Entfernung der Datenknoten vom Client geordnet ist, so dass der Client die Blockinhalte vom jeweils nächstgelegenen Datenknoten lesen kann. Beim Schreiben von Daten teilen Clients dem Namensknoten die gewünschte Blockgröße und den Replikationsfaktor mit. Dieser wählt geeignete Datenknoten zur Speicherung aus und verbindet den Client mit dem nächstgelegenen Knoten, auf welchen der Client die Daten überträgt. Die Replikation auf weitere Datenknoten findet automatisch statt. Eine detaillierte Betrachtung von HDFS findet sich z. B. in [38], [259], [265]. Zahlreiche Verbesserungsvorschläge, Vergleiche und Analysen der Performance finden sich in [73], [105], [127], [132], [133], [137], [142], [186], [191], [196], [225], [262], [285]. Die Grenzen der verwaltbaren Datenmenge (unter Verwendung eines einzelnen Namensknotens) werden in [266] ausgelotet.

---

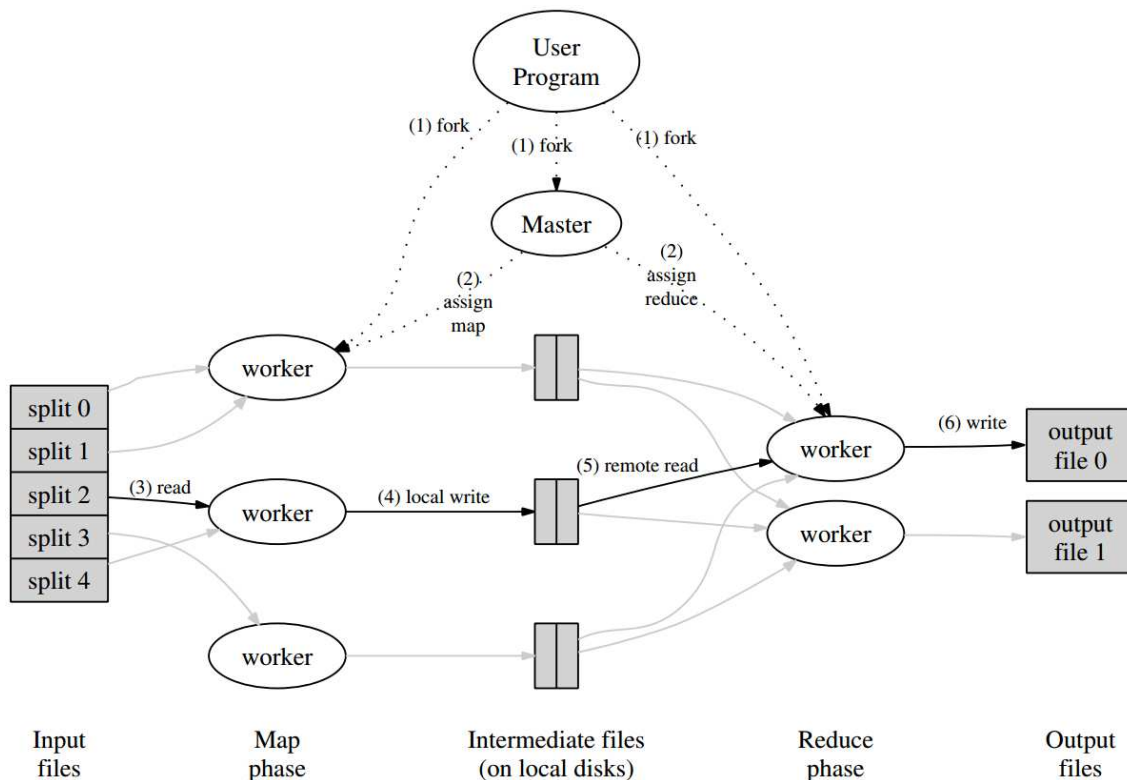
<sup>69</sup> POSIX steht kurz für engl. *Portable Operating System Interface*, eine durch die IEEE und die Open Group für Unix definierte und standardisierte (ISO/IEC/IEEE 9945) betriebssystemunabhängige Programmierschnittstelle zwischen Anwendungssoftware und Betriebssystemen. Die aktuelle Version IEEE Std 1003.1-2008 wird von den meisten Betriebssystemen vollständig oder teilweise umgesetzt.

<sup>70</sup> Projektseite: <http://nutch.apache.org>

Es ist zu betonen, dass zur Bewertung von HDFS als Dateisystem die typischen Maßstäbe und Performanz-Indikatoren für Dateisysteme zwar angelegt werden können, somit jedoch lediglich Lese- und Schreiboperationen bewertet werden und die Performanz der datenparallelen Verarbeitung mit MapReduce außer Acht gelassen wird, was in vielen Reviews leider der Fall ist.

#### A.3.3.5 MapReduce – das Hadoop Programmiermodell

Das *MapReduce-Programmiermodell* sieht eine Aufteilung durchzuführender Berechnungen in Map- und Reduce-Schritte vor. Beide operieren auf *Schlüssel/Wert-Paaren* und produzieren auch Schlüssel/Wert-Paare als Ausgangsdaten. Um eigene Berechnungen durchführen zu können, muss eine *Map*- und eine *Reduce*-Funktion implementiert werden. Die Map-Funktion hat die Aufgabe, aus Eingangsdaten in Form von Schlüssel/Wert-Paaren temporäre Schlüssel/Wert-Paare zu produzieren. Resultate mit dem gleichen temporären Schlüssel werden durch das Framework automatisch gruppiert und an die Reduce-Funktion weitergereicht. Die Reduce-Funktion erhält als Argumente also einen temporären Schlüssel und eine dazugehörige Sammlung von Werten in Form eines Iterators, so dass auch Datenmengen verarbeitet werden können, die nicht in den Hauptspeicher passen. Diese werden zu einer verkleinerten Sammlung von Werten zusammengefasst [66].



**Abbildung A.18:** Darstellung des Ablaufs einer MapReduce Operation. Quelle: [14].

Sowohl die Map- als auch die Reduce-Funktion werden während der Abarbeitung eines MapReduce-Programms mehrmals mit unterschiedlichen Teilmengen der Daten – und möglicherweise auf unterschiedlichen Datenknoten – aufgerufen (siehe Abbildung A.18). Das MapReduce-Programmiermodell gilt als sehr gut parallelisierbar. Insbesondere lässt sich damit eine datenparallele Verarbeitung realisieren. Auf die softwaretechnischen Besonderheiten bei der Implementierung von MapReduce-basierter Software wird z. B. in [110] eingegangen.

#### A.3.3.6 Verteilte Verarbeitung mittels Apache Pig

Um so weit wie irgend möglich unabhängig von der zugrundeliegenden Cluster-Architektur und der konkreten Version und Konfiguration von Hadoop entwickeln zu können und außerdem den Quellcode lesbar und übersichtlich zu halten, wurde von der umständlichen direkten Entwicklung von MapReduce-Code in Java für jede einzelne Verarbeitungsaufgabe abgesehen. Stattdessen wurde für jegliche datenparallele verteilte Verarbeitung die *Apache Pig* [321] API eingesetzt, welche automatisch MapReduce-Verarbeitungscode erzeugen und je nach Konfiguration auf eine Zielarchitektur optimieren und ausführen kann.

Apache Pig ist ein System zur Analyse großer Daten, das auf Hadoop und weiteren parallelen Verarbeitungsumgebungen lauffähig ist. Es beinhaltet eine abstrakte Datenfluss-Beschreibungssprache namens *Pig Latin*<sup>71</sup> (siehe [90]), welche von einem sog. *Pig Server* interpretiert und auf der Zielarchitektur ausgeführt wird. Für die verteilte Verarbeitung auf einem Hadoop-Cluster erzeugt der Pig Server somit optimierten MapReduce-Code. Optimiert wird dieser Code insofern, dass automatisch eine optimale Verarbeitungskette erzeugt wird, in der Map-, Combine- und Reduce-Tasks so angeordnet werden, dass möglichst wenig Datentransfer erfolgt und Teilergebnisse effizient wiederverwendet werden. Hierzu dienen in Pig Latin sog. *Relationen*, die auf den ersten Blick wie Variablen in modernen Programmiersprachen erscheinen, sich jedoch grundlegend anders verhalten. Pig Latin weist gemäß der Projekt-Homepage [321] drei wesentliche Eigenschaften auf, die Pig – gemeinsam mit der Flexibilität bei der Architekturwahl – zu einer optimalen Basis für die zu realisierende verteilte datenparallele Verarbeitung machen. Diese drei Eigenschaften<sup>72</sup> sind:

- **Einfache Programmierung:** Die parallele Ausführung einfacher „unerhört paralleler“ Datenanalyseaufgaben ist trivial. Komplexe Aufgaben, welche mehrere in Beziehung stehende bzw. voneinander abhängige Datentransformationen beinhalten, werden explizit als Sequenz von

---

<sup>71</sup> Die von Apache Pig verwendete Skriptsprache zur Datenflussbeschreibung namens *Pig Latin* sollte nicht mit dem gleichnamigen Pig Latin (engl.; wörtlich: „Schweine-Latein“) verwechselt werden, einer Spielsprache, die im englischen Sprachraum als einfache Gemeinsprache verwendet wird. Der Name stellt offensichtlich eine Anspielung auf ebendiese dar.

<sup>72</sup> Der englische Text wurde vom Autor frei übersetzt.

Datenflüssen beschrieben, wodurch diese einfach zu schreiben, zu verstehen und zu verwalten sind.

- **Optimierung:** Die Art der Kodierung der Aufgaben (in Pig Latin) erlaubt es dem System, die Ausführung automatisch zu optimieren. Dadurch kann sich der Nutzer mehr auf Semantik statt auf Effizienz konzentrieren.
- **Erweiterbarkeit:** Nutzer können das System durch eigene Funktionen für spezielle Verarbeitungsabläufe erweitern.

#### A.3.3.7 Erweiterung von Pig durch UDFs

Um eine datenparallele verteilte Verarbeitung mittels Hadoop MapReduce speziell für Zeitreihendaten zu ermöglichen, wurde spezifische Funktionalität benötigt, die nicht zum Funktionsumfang von Pig gehört, beispielsweise für die Vorverarbeitung, Transformation und Dimensionsreduktion von Zeitreihen. Um diese Verarbeitungsschritte zu ermöglichen, wurde der Funktionsumfang von Pig mit sog. UDFs (von engl. *User defined functions*) erweitert. Die entwickelten UDFs stellen in Java geschriebene Evaluations-Klassen dar. Diese müssen vor Verwendung am Pig Server registriert werden und können anschließend in Pig-Skripten als Operatoren eingesetzt werden (siehe [90]).

Dies hat den Vorteil, dass einzelne Verarbeitungsschritte modular für verschiedene Verarbeitungsketten (sog. *Datenverarbeitungs-Pipelines*) wiederverwendet werden können. Der Datenfluss kann somit weiterhin durch Pig Latin beschrieben und für die konkrete Ausführung auf dem Hadoop-Cluster optimiert werden, während die zusätzlichen Funktionen in Form ausführbarer Java Jar-Dateien automatisch auf die entsprechenden Zieldatenknoten verteilt werden, auf denen die zu verarbeitenden Daten gespeichert sind.

## A.4 Zusammenfassung

In diesem Kapitel wurden Methoden aus den Bereichen *Zeitreihenanalyse*, *Zeitreihenexploration* und *-visualisierung* sowie *Data-Intensive Computing* vorgestellt, die für das in der vorliegenden Arbeit entwickelte Zeitreihenanalyzesystem benötigt werden. Dabei wurde der Entwicklungsstand des jeweiligen Gebiets dargestellt und unter dem Aspekt der Skalierbarkeit untersucht. Die Eignung einzelner Methoden zur Umsetzung des vorgestellten Konzepts für ein skalierbares System zur explorativen Analyse großer Zeitreihendaten wurde herausgearbeitet und es wurden geeignete Methoden und Verfahren identifiziert und genauer beleuchtet.

Als Zeitreihenrepräsentation für die zu extrahierenden Metadaten wurde die symbolische SAX-Darstellung gewählt, welche die Umsetzung zahlreicher Analyseaufgaben bei einer niedrigen Laufzeitkomplexität erlaubt und die Dimensionalität der Originalzeitreihe stark zu reduzieren

vermag, wobei charakteristische Merkmale des Zeitreihenverlaufs erhalten bleiben. Des Weiteren wurden Visualisierungsmethoden vorgestellt und auf die Skalierungsproblematik bezogen, sowie Ansätze zur Darstellung der Zeitreihencharakteristik skizziert. Als geeignete Architektur zur datenparallelen verteilten Verarbeitung, welche sich auch für größte Datenmengen eignet, wurde *Hadoop* ausgewählt, das massiv horizontal skalierbar ist. *Hadoop* bietet effektive Methoden zur Entwicklung skalierbarer Datenanalysen und zur Ableitung datenreduzierter Metadaten, die den Zugriff auf die großen Originaldaten erleichtern können.



# Anhang B - Stromnetzanalyse

In diesem Abschnitt wird ein Anwendungsfeld näher beschrieben, in dem das vorgestellte Framework für skalierbare Verwaltung und Analyse großer Zeitreihendaten (*FraScaTi*) gewinnbringend angewandt werden kann und wurde. Es handelt sich um den Bereich der Stromnetzanalyse, die eine kritische Voraussetzung für geplante Anpassungen der Netzinfrastruktur an zukünftige Herausforderungen darstellt. In den kommenden Jahren werden durch großflächige Stromnetzanalysen große Zeitreihendaten aus synchronisierten Strom- und Spannungsmessungen anfallen, die mit neuen, skalierbaren Systemen verwaltet und ausgewertet werden müssen. Bislang sind hierzu nur wenige Verfahren bekannt, die in skalierbaren Analysesystemen einsetzbar sind, weshalb das *FraScaTi*-System einen wichtigen Beitrag dazu leisten kann, Stromdaten bisher nicht verarbeitbaren Umfangs explorierbar und analysierbar zu machen.

## B.1 Motivation der Stromnetzanalyse

Ein detailliertes Verständnis der komplexen Dynamik in großen Stromnetzen ist von großer Bedeutung für die Aufrechterhaltung der Stabilität der Stromversorgung. Um dies näher zu beleuchten, werden im Folgenden Gründe für eine zukünftige Zunahme dieser Komplexität aufgezeigt und Aufbau und Funktionsweise insbesondere des deutschen Stromnetzes kurz erklärt. Anschließend werden Aspekte der Netzqualität dargestellt. Im Rahmen der Anwendung des *FraScaTi*-Systems ist dabei die Stromqualität von besonderem Interesse, da hier große Messdaten exploriert und analysiert werden müssen.

### **B.1.2 Zunehmende Komplexität der Stromnetze**

Deutschland befindet sich wie zahlreiche andere Länder in der Frühphase einer Energiewende – hin zu vermehrter Integration erneuerbarer Energiequellen und einem dynamischen, modularen Verbund von Erzeugern und Verbrauchern. Das traditionelle System der Energieversorgung, in welchem Energie in großen Kraftwerken erzeugt, in elektrische Energie umgewandelt und direkt über weiträumige Verteilnetze zu den Verbrauchern transportiert wird, muss auf diese Veränderungen adäquat angepasst und modernisiert werden. Eine Zunahme der komplexen Dynamik innerhalb des Stromnetzes ist zu erwarten, ausgelöst etwa durch die zunehmende Integration direkt

## B.1 Motivation der Stromnetzanalyse

in Verteilnetze einspeisender Erzeuger erneuerbarer Energie, elektrischer Fahrzeuge, welche sowohl einspeisen als auch verbrauchen oder auch durch die Einführung des sog. Smart-Grid. Diese neu entstandene Komplexität muss genauer untersucht werden.

Bereits im Jahr 2012 wurden laut eines Fortschrittsberichts der Bundesregierung von 2013 [44] immerhin ca. 12,4 % der Elektrizität aus erneuerbaren Energien gewonnen. Der Anteil an sog. klimafreundlichen Energiequellen, also etwa Wind-, Solar- und Wasserkraft sowie aus Biomasse und Erdwärme gewonnener Energie lag zusammen sogar bei rund 20 %. Die Bundesregierung strebt laut [280] für 2020 eine Erhöhung dieses Anteils auf mindestens 35 % des Stroms an, die Branche der Erneuerbaren Energien rechnet dann schon mit 47 %. Die schnelle Verteilung einer solchen dezentralen dynamischen Energieeinspeisung bringt das bisherige statische Netz jedoch an seine Grenzen. Für die zukunftsfähige Planung des Stromnetzes bedarf es eines genauen Modells seiner bisher weitgehend unbekannt internen Dynamik, welches basierend auf Analysen groß angelegter, detaillierter Messungen und Simulationen auf allen Skalen aufgebaut werden muss.

### B.1.3 Das deutsche Stromnetz

Der Begriff *Stromnetz* bezeichnet die Summe an Komponenten des Netzwerks, in welchem Stromerzeuger und -verbraucher durch elektrische Leitungen miteinander verbunden sind. Das von den Stromversorgern betriebene deutsche Stromnetz ist historisch gewachsen, wobei sich seine Aufgaben und sein Betrieb mit der Zeit stark verändert haben. Es umfasst laut BDEW<sup>73</sup> ungefähr 1 700 000 Kilometer an verlegten Stromkabeln, ca. 350 000 Kilometer davon sind Freileitungen. Das Verbundnetz beinhaltet viele regionale Teilnetze mit unterschiedlichen Spannungsniveaus, welche durch über 550 000 Transformatoren gekoppelt sind. Der Großteil der Kabelstrecke ist der Niederspannungsebene und damit dem Verteilungsnetz zuzurechnen. Im europäischen Vergleich stellt das deutsche Stromnetz damit das dichteste Netz dar, aber auch eines der ältesten. Um die Problematik beim Übergang des Stromnetzes zu einem intelligenteren modularen Netz mit neuen Anforderungen zu verstehen, muss man seine heutige Struktur und Funktionsweise kennen, weshalb diese im Folgenden dargestellt wird.

#### B.1.3.1 Aufbau

Das deutsche Stromnetz ist ein Verbund an Teilnetzen mit unterschiedlicher Funktion. Es besteht aus dem Übertragungsnetz und dem Verteilungsnetz. Innerhalb des Verteilungsnetzes lassen sich

---

<sup>73</sup> BDEW steht für Bundesverband der Energie- und Wasserwirtschaft [119].



wiederum das Hochspannungsnetz, das Mittelspannungsnetz und das Niederspannungsnetz unterscheiden.

Der Gesamtaufbau ist in Abbildung B.1 dargestellt.

Laut einer vom *Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit* mitherausgegebenen Informationsbroschüre [228] der *Deutschen Umwelthilfe* dient das sog. *Übertragungsnetz* der Fernübertragung großer generierter Strommengen über große Distanzen, während über das sog. *Verteilungsnetz* kleine und mittelgroße Erzeuger und Endverbraucher auf lokaler Ebene verbunden werden.

Das Übertragungsnetz ist als Höchstspannungsnetz mit 220 Kilovolt (kV) oder 380 kV Betriebsspannung realisiert. Hierüber können große Strommengen über lange Transportstrecken ohne größere Verluste übertragen werden. Eingespeist wird der Strom bedeutender Großkraftwerke, v. a. aus Kernkraft-, Kohle- und Gaswerken, sowie aus großen Offshore-Windparks<sup>74</sup>. In sog. *Umspannwerken* wird der Strom dann auf das jeweils benötigte Spannungsniveau angeschlossener Verteilnetze abgesenkt. Hier existieren wiederum drei Teilnetze mit unterschiedlicher Betriebsspannung:

- An das *Hochspannungsnetz* sind sowohl mittelgroße Erzeuger, z. B. Kohle- und Gaskraftwerke oder Windparks, als auch große Abnehmer, wie etwa Industrieanlagen und Rechenzentren, direkt angeschlossen. Es kann mit Spannungen von 60 kV oder 110 kV betrieben werden und ist somit für größere und mittlere Transportwege geeignet.
- Das *Mittelspannungsnetz* unterstützt mit Betriebsspannungen zwischen 1 kV und 60 kV viele unterschiedliche Modi. Angeschlossen sind an dieses Verteilnetz mittelgroße Erzeuger wie Heizkraftwerke, Solarparks, Wasser- und Windkraftwerke und gewerbliche und industrielle Verbraucher.
- Das *Niederspannungsnetz* ist schließlich für den Anschluss kleinerer Endverbraucher zuständig. Dies sind neben Privathaushalten kleinere Industrieanlagen. Auch eine direkte Einspeisung aus kleineren Erzeugern, wie z. B. Fotovoltaik-Anlagen, ist möglich. Das Niederspannungsnetz wird bei 230 V oder 400 V betrieben.

Die verschiedenen Netzformen und Schutzklassen sowie Bestimmungen zu Netzkomponenten wie etwa maximal zulässige Längen von Kabeln und Leitungen werden in der *DIN VDE 0100* als Teil der VDE-Bestimmungen „*Gruppe 0+1 – Grundsätze + Energieanlagen*“ behandelt. Nähere Informationen hierzu finden sich z. B. in [157] und [247].

---

<sup>74</sup> Offshore-Windparks sind große Windkraftanlagen auf dem offenen Meer, in Deutschland v. a. in der Nordsee.

## B.1 Motivation der Stromnetzanalyse

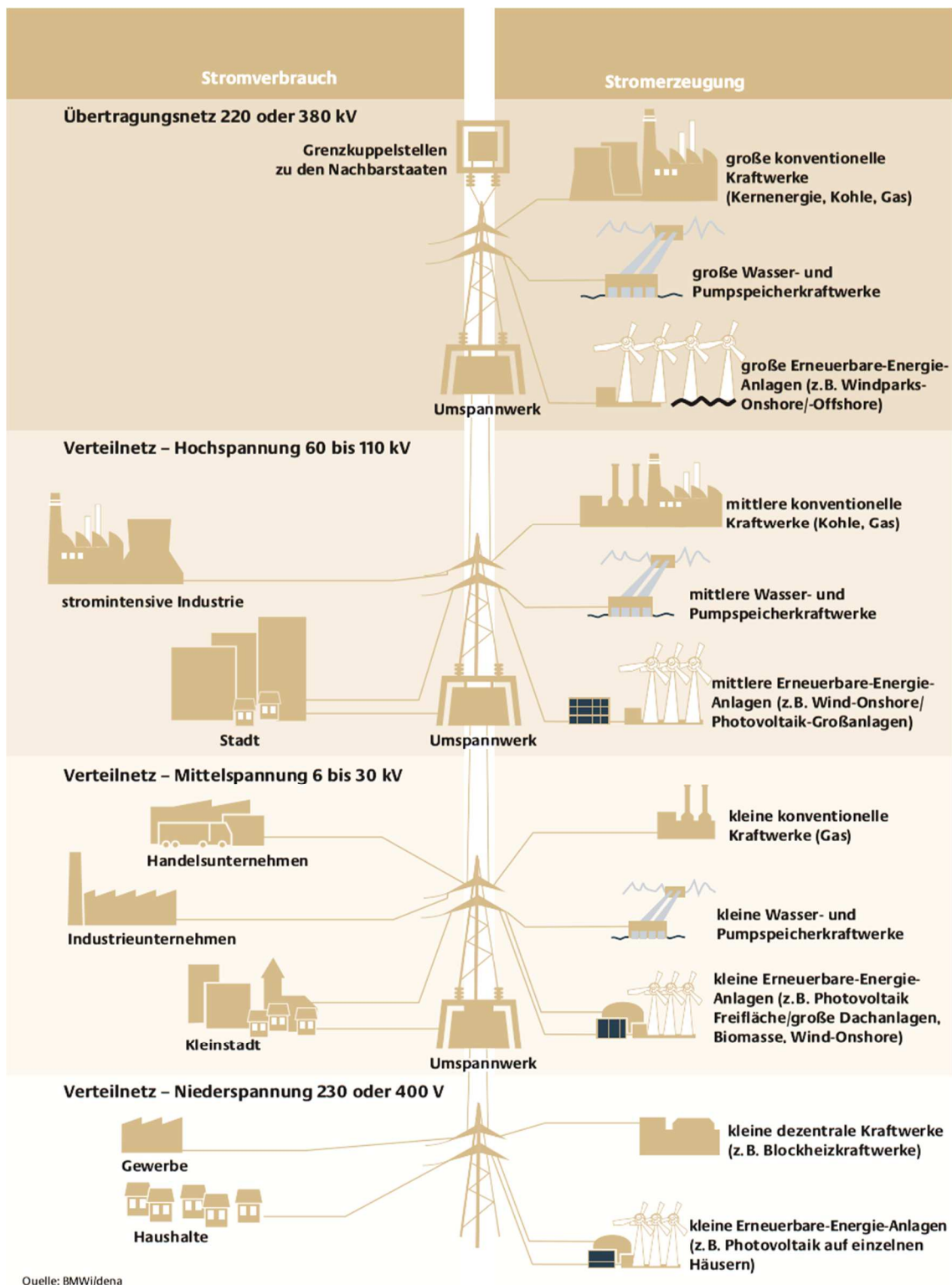


Abbildung B.1: Aufbau des deutschen Stromnetzes (Quelle: BMWi/dena, modifiziert).

### B.1.3.2 Lastregelung über Netzfrequenz

Im Jahr 2011 betrug der gesamte deutsche Bruttostromverbrauch<sup>75</sup> etwa 609 TWh [77]. Bei solchen Strommengen bedarf es einer geeigneten Lastregelung. Momentan findet ein Lastausgleich zwischen Erzeugern und Verbrauchern über die Regelung der Netzfrequenz statt. Für die Generatoren, welche die Spannungsstabilität durch die Trägheit großer Rotationsmassen aufrechterhalten, wird die Frequenz systemweit synchronisiert [167].

Übersteigt die Leistungsaufnahme die eingespeiste Leistung, wird auf die Rotationsenergie der Generatoren zurückgegriffen. Diese verlangsamen ihre Drehung und in Folge sinkt die Netzfrequenz. Ebenso steigt die Frequenz im Falle einer Überproduktion. Man bezeichnet diesen Mechanismus der zügigen Begrenzung von Frequenzschwankungen (im Sekundenbereich) als *Primärregelung* [31].

Des Weiteren können Generatoren, die aktuell nicht betrieben werden, zur Unterstützung der Kapazität zugeschaltet werden. Man bezeichnet die potentiell zuschaltbaren Erzeuger als *Kaltreserve*.

Um allzu große Abweichungen von der Sollfrequenz zu verhindern, existieren weitere Regelungsmechanismen, welche als *Sekundärregelung* bezeichnet werden. Diese wird nach 30 Sekunden automatisch aktiviert und soll Leistungsungleichgewichte automatisch innerhalb von fünf Minuten, durch Regelleistung aus thermischen Kraftwerken und Pumpspeicherkraftwerken, ausregeln [31].

Man kann jedoch davon ausgehen, dass die bisherigen Regelungsmechanismen nicht ausreichen werden, um kurzfristige transiente Last- und Spannungsänderungen auszugleichen, wie sie für zukünftige Szenarien zu erwarten sind. Auf plötzliche Veränderungen kann nicht schnell genug reagiert werden und es kann nahezu keine erzeugte Energie längerfristig gespeichert werden. Jede erzeugte Leistung muss also sofort durch eine gleich große Leistungsaufnahme entnommen werden. Die Transportstrecken können dabei durchaus lang sein, man denke etwa an die Weiterleitung von in Offshore-Windparks in der Nordsee generierter Energie zu wirtschaftlichen Ballungsräumen in Baden-Württemberg und Bayern. Der Trend geht aber hin zu einem eher modularen Verbund lokaler Erzeuger und Verbraucher, um besser dezentral einspeisen zu können und lange Transportwege zu vermeiden.

Eine schnellere und flexiblere dezentrale Verteilung soll zukünftig vor allem durch die flächendeckende Umstellung auf *intelligente Stromnetze*, sog. *Smart Grids* erfolgen. Dies sind neuartige dy-

---

<sup>75</sup> Der Bruttostromverbrauch eines Landes stellt die Summe des importierten und inländisch produzierten Stroms unter Einbeziehung aller Erzeuger nach Abzug des exportierten Stroms dar [43].

## B.2 Netzqualität

namische Stromnetze mit modularem Aufbau, die Sensorik, Kommunikationsnetze und operationale Steuerungslogik integrieren. Das Ziel dabei ist es, die komplexe Dynamik der zukünftigen Energieversorgung in Echtzeit überwachen und optimieren zu können.

### B.1.3.3 Spannungsregelung

Zusätzlich zur Last bedarf auch die Spannung an vielen Stellen einer Regelung. Gründe hierfür sind verbrauchsbedingte Spannungsschwankungen aber neuerdings auch Schwankungen aufgrund der ungleichmäßigen Einspeisung dezentraler Erzeuger direkt in die Verteilnetze. Die Regelung ist zumeist an Übergangs- und Verbundpunkten von Niederspannungsnetzen notwendig, seltener aber auch in den Übertragungsnetzen, wenn die Spannung etwa schnell stark abnimmt, was für den Stromtransport kritisch sein kann.

Die Versorgungsspannung erfolgt i. d. R. direkt über Hoch- und Mittelspannungs-Transformatoren mit manuell einstellbaren elektromechanischen Stufenschaltern, vor allem an Anschlusspunkten von Ortsnetzen, oder indirekt über eine Einspeisung oder Entnahme von Blindleistung. Letztere Methode erweist sich als flexibler, da schnell und stufenlos geregelt werden kann und somit – zumindest theoretisch – auch dynamische Spannungsschwankungen ausgeglichen werden können. Häufig kommt es dabei jedoch zu verstärktem Blindleistungsfluss, was zu unerwünschten Effekten, etwa zu verringerter Einspeisungskapazität von Wirkleistung, führen kann [248].

Die indirekte Spannungsregelung wird durch parallele Spulen mit unterschiedlicher Wicklungszahl realisiert, wobei sich das Verhältnis der Wicklungszahlen durch elektromagnetische Induktion auf die Ausgangsspannung auswirkt.

## B.2 Netzqualität

Der Begriff Netz- bzw. Stromqualität bezieht sich auf die elektrische Interaktion zwischen dem Elektrizitätsnetz und den daran angebotenen Teilnehmern bzw. Gerätschaften [35]. Die Qualität von Stromnetzen beinhaltet folgende Aspekte (teilweise aus [116]):

- Versorgungszuverlässigkeit (Unterbrechungen pro Zeit),
- Spannungsqualität (Oberschwingungen, Flicker, Einbrüche etc.),
- Stromqualität (Oberschwingungen, Flicker, Einbrüche etc.) und
- Servicequalität (Interaktion zwischen Netzbetreiber und Kunde).

Beschränkt man die Betrachtung auf die rein technischen Aspekte spricht man in der Regel von der sog. *Netzqualität*, womit v. a. die Qualität der Versorgungsspannung und des Stroms gemeint

ist. Der Begriff der Stromqualität, unter dem diese beiden Aspekte zusammengefasst werden können, ist in der Literatur nicht eindeutig definiert. So besteht Uneinigkeit darüber, ob hierunter nur Effekte eines fehlerhaften Gerätebetriebs zu verstehen sind oder auch Versorgungsunterbrechungen oder gar auch Auswirkungen harmonischer Schwingungen bzw. Abweichungen von der idealen Wellenform für Strom und Spannung [35].

In der vorliegenden Arbeit wird das Hauptaugenmerk auf die Untersuchung der Wellenform gelegt, so dass die erarbeiteten Ansätze und Methoden durchaus auch auf andere Anwendungsgebiete übertragbar sind. Als Strom-Anomalitäten werden alle Abweichungen von der normalen und idealen Wellenform für Strom und Spannung angesehen. Die ideale Wellenform der Spannung ist eine unverzerrte Sinusschwingung mit konstanter Amplitude und Frequenz entsprechend der Nennwerte. Für die Wellenform des Stroms ist die Übereinstimmung der Phase mit der Sinusschwingung der Spannung entscheidend.

Die Standards und Empfehlungen zur Mess- und Auswertemethodik sind teilweise widersprüchlich [165] und sollten deshalb nicht unkritisch übernommen und angewandt werden (Differenzen zwischen IEEE 1459-2010 [124] und CPC [63] sind in [165] zu finden). Zu den Standards, die Empfehlungen zur Methodik der Stromqualitätsanalyse geben, gehören IEEE 1459-2010 [124] und [63], die beide auf einer Komponentenzerlegung basierend auf der Fourier-Transformation basieren. Mindestanforderungen an die Spannungsqualität werden z. B. in der europäischen Norm EN 50160 definiert, die zu verwendenden Messmethoden hierzu und die Interpretation der Ergebnisse in IEC 61000-4-30. Laut letzterer sind für die Netzqualität die folgenden Attribute und Effekte relevant [138]:

- Versorgungsunterbrechungen (Dauer)
- Spannungseinbrüche (Häufung und Ausprägung)
- Spannungshöhe, langsame Spannungsänderungen
- Schnelle Spannungsänderungen, Flicker (Häufung und Ausprägung)
- Spannungsunsymmetrie
- transiente und netzfrequente Überspannungen
- Frequenz der Spannung
- Wellenform der Spannung (Oberschwingungen, Zwischenharmonische, Signalspannungen).

### **B.2.1 Stromausfälle und Versorgungsengpässe**

Ein *Stromausfall* bedeutet den kompletten Ausfall der Stromversorgung innerhalb eines Teilnetzes. Der Begriff ist zu unterscheiden von der sog. *Versorgungsunterbrechung* eines oder mehrerer Endkunden für mehr als eine Sekunde [31].

Da heute ein entscheidender Teil der kritischen Infrastruktur elektronisch gesteuert wird, bedeutet ein Stromausfall gleichzeitig auch die Nichtverfügbarkeit jener Infrastruktur – etwa für Internet, Börse, Finanzdienstleistungen, Krankenhäuser und Gesundheitswesen, Kraftwerke, Kommunikation von Behörden, Wasserversorgung und -aufbereitung sowie Abwasser, Kühlanlagen, Lebensmittelversorgung, Rechenzentren, Schienenverkehr, Verkehrs-, Park- und Ampelsysteme, Straßenbeleuchtung, Rolltreppen und Fahrstühle, etc. Eine detaillierte Analyse der Folgen eines großräumigen, langandauernden Stromausfalls für die Gesellschaft wurde z. B. 2011 in einem Bericht [16] des Ausschusses für Bildung, Forschung und Technikfolgenabschätzung des Deutschen Bundestags veröffentlicht. Darüber hinaus kann erheblicher Schaden für die Industrie entstehen: Produktions- und Verarbeitungsprozesse stehen still und empfindliche Maschinen und Anlagen können dauerhaft geschädigt werden, weshalb in kritischen Bereichen meist Notstromaggregate verbaut werden. Mit den typischen dieselbetriebenen Notstromsystemen lassen sich für gewöhnlich ca. drei Tage Versorgungsunterbrechung überbrücken. Danach werden neue Treibstoffvorräte benötigt, die jedoch bei einem langfristigen, großflächigen Ausfall logistisch schwer zu beschaffen sind. Man kann also sagen, dass die Kosten schneller als linear mit der Dauer und der Größe der betroffenen Region des Ausfalls steigen, da die Wiederaufnahme des Betriebs vieler Anlagen mit der Ausfallzeit immer schwieriger wird. Einige Anlagen – etwa Kernkraftwerke – können überhaupt nicht kurzfristig komplett stillgelegt werden und benötigen eine unterbrechungsfreie Stromversorgung. Die bisherige zentralisierte hierarchische Struktur des Stromnetzes führte in der Vergangenheit in Europa bereits zu vielen großen Stromausfällen und damit zu enormen finanziellen Schäden. In Tabelle B.1 werden einige konkrete Beispiele für typische Verluste aus unterschiedlichen industriellen Sparten gemäß [255] gezeigt.

**Tabelle B.1:** Finanzielle Verluste durch Stromausfälle.

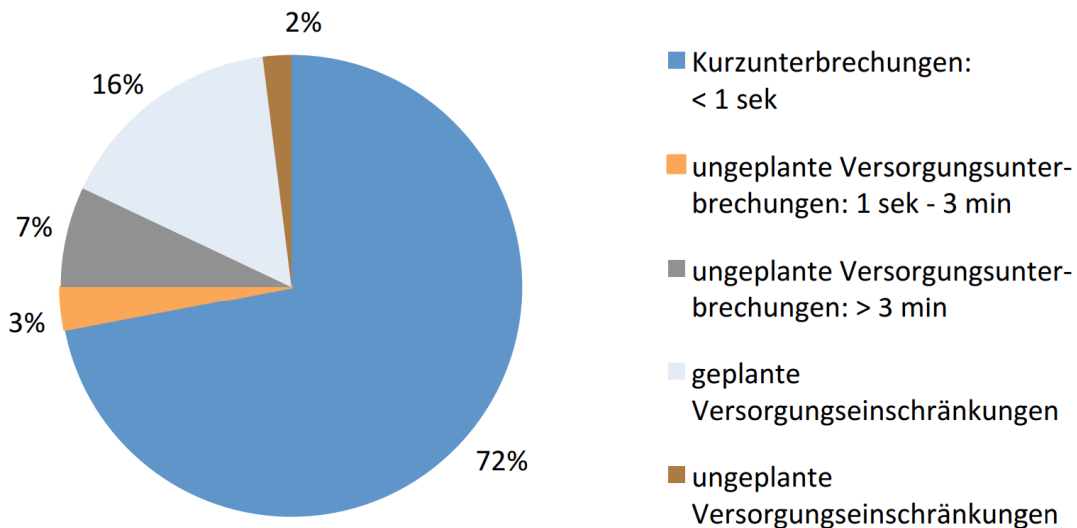
<i>Sparte</i>	<b>Monetärer Verlust</b>	<b>Zeitraum</b>
<i>Finanzverkehr</i>	6 000 000 €	pro Stunde
<i>Halbleiter-Produktion</i>	3 800 000 €	pro Vorfall
<i>Rechenzentrum</i>	750 000 €	pro Vorfall
<i>Telekommunikation</i>	30 000 €	pro Minute
<i>Stahlindustrie</i>	350 000 €	pro Vorfall
<i>Glasindustrie</i>	250 000 €	pro Vorfall
<i>Offshore-Plattformen</i>	250 000 – 750 000 €	pro Tag
<i>Baugewerbe &amp; Landgewinnung</i>	50 000 – 250 000 €	pro Tag

Gründe für langfristige Stromausfälle können unterschiedlicher Natur sein und sind oft nur schwierig zu identifizieren. Der unvorhergesehene Ausfall großer Kraftwerke, beispielsweise durch Beschädigung von Generatoren ohne sofortigen Ersatz, führt zu Einspeisungsengpässen. Im Einzelfall kann ein solcher durch vermehrte Stromproduktion und -einspeisung durch andere

Kraftwerke meist kompensiert werden. Das Versorgungssystem ist jedoch überlastet, sobald mehrere Produzenten ausfallen, was zum Stromausfall für ganze Regionen führen kann. Des Weiteren können Ausfälle durch beschädigte Stromnetzkomponenten wie Hochspannungsleitungen und Transformatoren entstehen, wobei auch diese meist redundant ausgelegt und robust gegenüber Einzelausfällen sind. Insbesondere in den Wintermonaten oder bei größeren Baumaßnahmen können jedoch mehrere Komponenten gleichzeitig ausfallen, was ohne sofortige Kompensation kleinere lokale Stromausfälle zur Folge hat. Oftmals müssen in solchen Fällen starker Netzbelastung aber auch größere Teilnetze stillgelegt werden, um die Gesamtlast zu verringern, da es ansonsten kurzfristig zu Überlastungssituationen kommen kann, welche Stromnetze ganzer Regionen zusammenbrechen lassen könnten.

Die Bundesnetzagentur beziffert in [42] 206 000 Unterbrechungen im Stromnetz und die Gesamtzeit unterbrochener Stromversorgung im Jahre 2013 auf 15,32 Minuten. Der Wert ist im Vergleich zu 2012 (15,91 Minuten) geringfügig gesunken und liegt unter dem Mittelwert für die Jahre 2006 bis 2012 von 16,92 Minuten. Die Ausfallzeiten belegen, dass in Deutschland die Versorgungszuverlässigkeit im europäischen Vergleich noch immer außerordentlich gut ist. Die Zahlen basieren auf rund 179 000 rückgemeldeten Unterbrechungen in den 878 Netzen seitens der 868 deutschen Betreiber.

Von der Bundesnetzagentur werden jedoch nur Unterbrechungen von mehr als 3 Minuten Dauer einbezogen, die außerdem im Voraus nicht bekannt bzw. geplant waren und auf technisches oder menschliches Versagen, nicht jedoch auf sog. höhere Gewalt zurückzuführen sind. Es zeigt sich



©FfE IHK-01 Leitfaden Energiewende\_00124

**Abbildung B.2:** Aufteilung der Versorgungsstörungen nach Vorfalltyp (Quelle: [94])

jedoch, dass der größte Anteil an Versorgungsstörungen mit rund 72 % kurzfristige Unterbrechungen sind und nur 7 % zum Typ der erfassten Störungen gehören (siehe Abbildung B.2). Die Erfassungs-Methodik der Bundesnetzagentur ist somit für eine Qualitätsbewertung als unzureichend zu bewerten, da der Großteil der Störungen nicht erfasst wird.

### **B.2.2 Kurzfristige Spannungseinbrüche und Überspannungen**

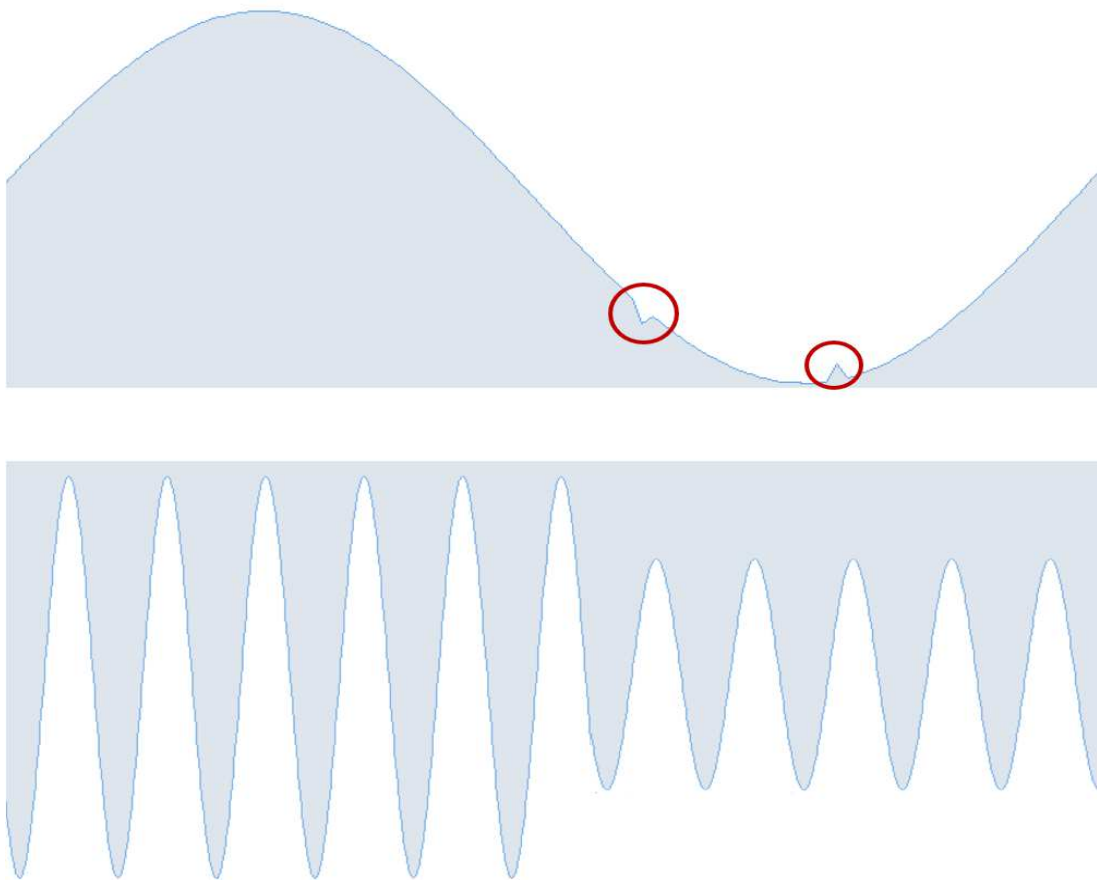
Nicht nur langandauernde Ausfälle der Stromversorgung führen zu Problemen im Stromnetz, sondern auch kurzfristige Änderungen der Spannung. Laut der Norm EN 50160 sind Änderungen der Bezugsspannung um bis zu 10 % zulässig, trotzdem können auch bereits geringfügige Abweichungen empfindliche Anlagen stören. Verursacht werden diese Schwankungen meist durch plötzliche Änderungen von Last oder Leistungsaufnahme und Schaltvorgängen oder kurzzeitigen Kurzschlüssen im Versorgungsnetz [77].

Gerade *transiente Spannungseinbrüche* und *-spitzen* können elektrischen Geräten langfristig auch irreversibel schaden. *Transiente Über-* bzw. *Unterspannungen* bezeichnen kurzzeitige Über- bzw. Unterspannungen mit einer Dauer von einigen Millisekunden, z. B. aufgrund von Schalthandlungen, Auslösen von Sicherungen oder durch Blitzeinschläge [77].

Als *Transienten* bezeichnet man allgemein (auch bezogen auf den Strom) höherfrequente instationäre Schwingungen, die innerhalb einer Periode einer sinusförmigen Schwingung auftreten und sich nicht periodisch wiederholen. Unterschiedliche Transienten können in verschiedenem Maß Einfluss haben auf die Nulldurchgänge, die Mittelwerte und Extremwerte der Schwingung. In diesen Fällen sind sie verhältnismäßig einfach detektierbar. Es treten jedoch häufig auch Transienten wie die in Abbildung B.3 (oben) dargestellten auf, die keinen oder nur geringen Einfluss auf die genannten Merkmale haben. Diese sind mit den Auswertemethoden gemäß der Norm *DIN EN 61000-4-30* bzw. *VDE 0847-4-30* nicht zuverlässig detektierbar, da die hierin abgeleiteten Merkmale eine Mittelung über mehrere Perioden vorsehen.

Laut des Marktforschungsunternehmens IDC sind beispielsweise rund ein Drittel aller Ausfälle von Computeranlagen auf kurzfristige Stromstörungen zurückzuführen (vergl. [256]).

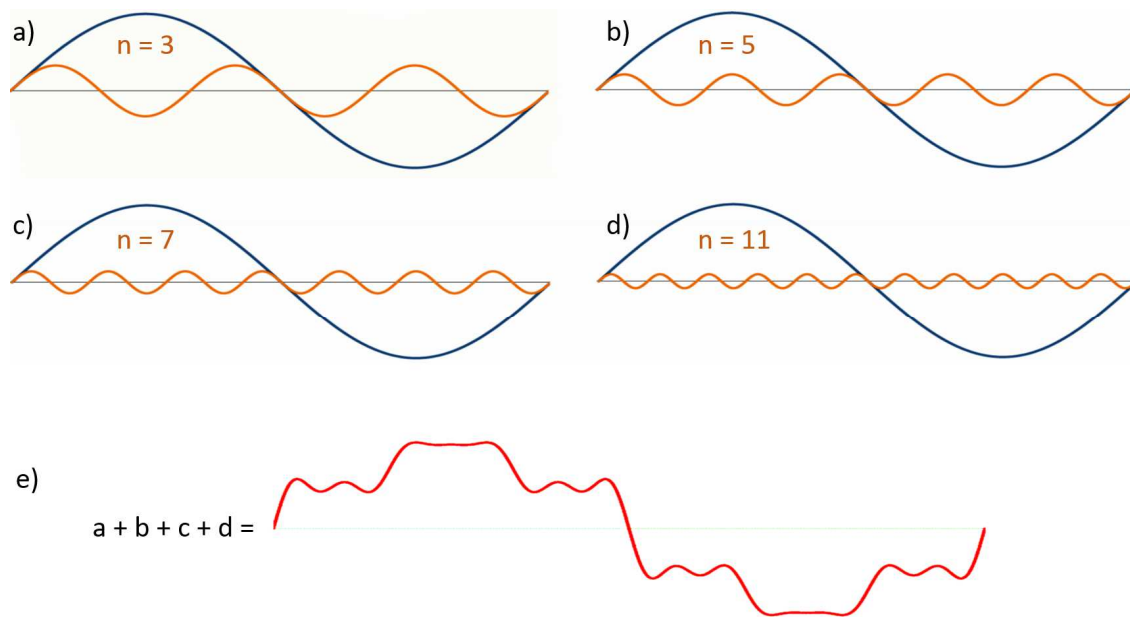




**Abbildung B.3:** Typische kurze Spannungseinbrüche, oben: Sinus-Periode mit Transienten (rot markiert), unten: plötzlicher Spannungseinbruch über mehrere Perioden.

### B.2.3 Netzzrückwirkungen und Oberschwingungen

Idealerweise besitzt der Spannungsverlauf in einem Verteilnetz eine unverfälschte Sinusform. Kommt es zu Abweichungen hiervon, stellt dies eine weitere Belastung des Stromnetzes dar. Sowohl Verbraucher als auch Erzeuger mit besonderer Stromaufnahmecharakteristik können zu sog. *Netzzrückwirkungen* führen, wobei sie bei einer anliegenden sinusförmigen Spannung einen charakteristischen nicht-sinusförmigen Strom aufnehmen bzw. einspeisen. Es kann dabei zu Oberschwingungen sowohl in der Spannung als auch im Strom kommen. In früheren Zeiten wurden diese Effekte meist durch Verbraucher wie z. B. große Frequenzumrichter-Motoren ausgelöst. Heute sind es auf Verbraucherseite hauptsächlich einphasige Wechselstromverbraucher – häufig Netzteile und Energiesparbirnen – und auf Erzeugerseite kleinere Solar- und Windkraftanlagen. Neuerdings verbreiten sich mehr und mehr auch elektronische Autos und Fahrräder, deren Ladevorgänge ebenfalls zu Rückwirkungen führen können [281].



**Abbildung B.4:** a) bis d): Ideale Sinusschwingungen (blau) mit harmonischen Schwingungen mehrfacher Grundfrequenz (orange), e): Überlagerungssignal (rot).  
Quelle: [138], modifiziert.

Derartige Komponenten produzieren *Oberschwingungsströme*. Diese wirken wiederum auf die *Netzimpedanz* zurück und lösen einen entsprechenden Spannungsabfall aus, welcher dann auch an anderen elektrischen Geräten anliegt, die bei gleicher Netzimpedanz angeschlossen sind. Auf diese Weise breitet sich die Störung im Teilnetz aus und kann dort zu Schäden führen [21]. Der Umfang und die Dynamik der Ausbreitung sind noch weitgehend unklar, da hierzu hochfrequente, synchronisierte Messungen an vielen Positionen im Netz notwendig wären. Abbildung B.4 zeigt vier Beispiele (a) – d)) harmonischer Schwingungen (orange) mit ganzzahlig mehrfacher Grundfrequenz einer Sinusschwingung (blau). Der ganzzahlige Faktor  $n$  gibt die Anzahl an Perioden an, welche die Oberschwingung während einer Periode der Grundschwingung durchläuft und bezeichnet die *Ordnung der Oberschwingung*<sup>76</sup>.

Die Frequenz einer harmonischen Oberschwingung ist also definiert als

$$f_h = n * \text{Grundfrequenz.} \quad (\text{B.1})$$

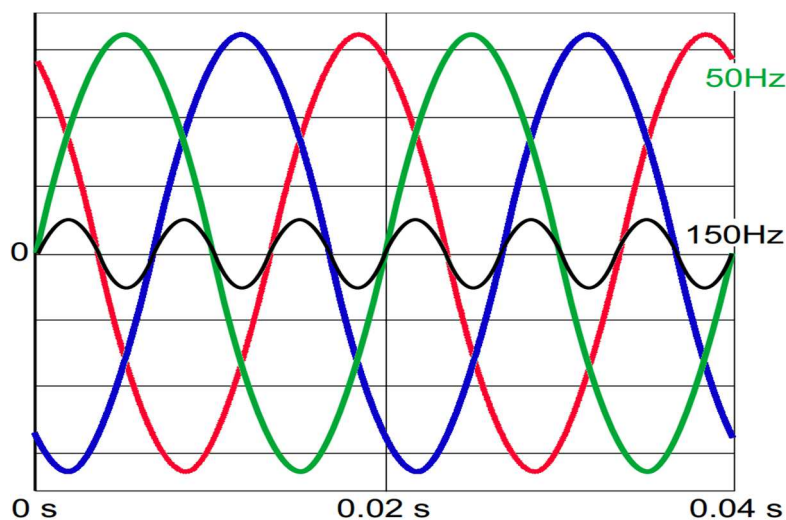
<sup>76</sup> In Fachliteratur wird häufig auch von einer *Harmonischen n-ter Ordnung* oder der *n-ten Harmonischen* gesprochen, wobei hier seltsamerweise die n. Harmonische die n-fache Grundfrequenz besitzt und der (n-1). Oberschwingung entspricht [30], [216].

Ein Oberschwingungsstrom 3. Ordnung hat bei einer Netzspannung von 50 Hz demnach eine Frequenz von  $3 * 50$  Hz, also 150 Hz. Gemäß [138] können drei Klassen harmonischer Oberschwingungen unterschieden werden:

**Tabelle B.2:** Klassen ganzzahlig vielfacher Oberschwingungen.

Ordnungsklasse	Ungerade	Gerade	Vielfache von 3
Ordnungszahlen (Beispiele)	3, 5, 7, 9, 11, 13, ...	2, 4, 6, 8, 10, 12, ...	3, 6, 9, 12, 15, ...

Ungerade Oberschwingungen, welche bezogen auf die Zeitachse symmetrisch sind, treten in heutigen Stromnetzen gehäuft auf. Aufgrund der meist dreiphasig ausgelegten Symmetrie aktueller Netze sind nahezu alle vorkommenden Signale symmetrisch. Gerade Oberschwingungen hingegen basieren auf unsymmetrischen Wellenformen und sind daher in Netzen mit sinusartigen Verläufen eher selten [138]. Anders als die meisten Oberschwingungsströme addieren sich in dreiphasigen Netzen Oberschwingungen 3. Ordnung (siehe Abbildung B.5) im sog. Generatorsternpunkt<sup>77</sup> arithmetisch auf [116].



**Abbildung B.5:** Kurvendarstellung von Strom im 3-Phasensystem: je zwei Perioden pro Phase mit 50 Hz (r, g, b) und sechs Perioden einer überlagerten Schwingung mit 150 Hz (schwarz), der *dritten Harmonischen*.

<sup>77</sup> In *Sternschaltungen* sind mehrere Leitungen über einen Widerstand mit einem Leiter verbunden, der als Sternpunkt bezeichnet wird.

Dies lässt sich aus ihrer besonderen Phasenfolge erklären. Oberschwingungen lassen sich gemäß ihrer Positiv-, Negativ- und Null-Sequenzen in sog. *Mit-*, *Gegen-* und Nullsysteme einteilen [138] (siehe Tabelle B.3).

**Tabelle B.3:** Symmetrische Phasenkomponenten von Oberschwingungen nach [138].

Symmetrie:	Mitsystem +	Gegensystem –	Nullsystem unipolar
<b>Ordnungszahl</b>	1	2	3
	4	5	6
	7	8	9
	10	11	12
	...	...	...
	$3k + 1$	$3k + 2$	$3k + 3$

Mitsysteme folgen der Phasenfolge der Grundschwingung und „zirkulieren“ zwischen den drei Phasen. Gegensysteme haben zur Grundkomponente entgegengesetzte Phasenfolgen und zirkulieren ebenfalls zwischen den Phasen. Nullsysteme jedoch sind unipolar und zirkulieren zwischen Phase und Neutralleiter [138]. Dadurch kann es zu Spannungsunsymmetrie kommen, die sich negativ auf die Spannungsqualität auswirkt.